TALK CODE-Y TO ME:AN ANALYSIS OF SPEECH TO TEXT SYSTEMS FOR CONSIDERATION OF USE IN WRITING SOFTWARE

by

Isaac Tijerina, BSA

DISSERTATION

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Software Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

MAY, 2022

TALK CODE-Y TO ME: AN ANALYSIS OF SPEECH TO TEXT SYSTEMS FOR CONSIDERATION OF USE IN WRITING SOFTWARE

by

Isaac Tijerina

APPROVED BY

____
Soma Datta, PhD, Chair

____
James Carlton Helm, PhD, Committee Member

____
Lisa Louise Lacher, PhD, Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF <COLLEGE NAME>:

____
David Garrison, PhD, Interim Associate Dean

____
Miguel A. Gonzalez, PhD, Dean

Acknowledgements

Thank you to God and my parents.

ABSTRACT

TALK CODE-Y TO ME:AN ANALYSIS OF SPEECH TO TEXT SYSTEMS FOR CONSIDERATION OF USE IN WRITING SOFTWARE

Isaac Tijerina
University of Houston-Clear Lake, 2022

Dissertation Chair: Soma Datta, PhD

This study proposes to create an application to allow ease of Speech to Text (STT) conversion specifically for programmers to make programming more accessible to those with disabilities. Recently there is being a movement of pairing STT with other disciplines now that STT is readily available and reliable. The main questions are how well Apple's STT performs, is Apple's STT ready to be integrated with coding, how do programmers interpret and speak code aloud, and how well does a formatting application created for this study to format transcriptions into executable code perform. The study concludes that Apple's STT transcribes text at an average success rate of 50.1% and in correctly transcribing and interpreting words at an average success rate of 13.12%, whether it is ready to be used in coding is up to the reader, programmers interpret and speak code in wide range of ways, and the application had a success rate of 0% but the data collected will help it to improve in the future.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:

INTRODUCTION

1.1) Introduction

Speech To Text (STT) or Speech to Text Recognition (STR) [5] is the ability of a computer to receive audio input from a person and translate it to text. This has several uses from automatic typing and display [9] to creating subtitles [7] to automatic translation from one language to another [5]. The process that allows STT to occur is first the speaker states something aloud, then it is converted from analog to digital sound, it is then sent to an algorithm that will transcribe the speech, and the algorithm then returns the text of what was said [3].

Speech To Text is not a new concept. It began in the 1950s and has picked up momentum in the 2000s as technology grows and is more capable of handling large vocabularies [6]. As it progresses many applications for it are being researched and applied in fields like psychology, doctor offices, and education [1][9][5]. What has been found is that as the field grows the applications for STT are growing. It began with attempting to get speech to be captured and converted to text on a computer, which is still occurring such as Iancu’s study to expand the language ability to include Romanian or the study to expand it to the language Yorùbá and is being improved upon performancewise such as adding background noise which shows a higher accuracy [7][2][9]. It has also been shown that STT is something people use and performs well on all sizes of devices [10][3].

Now as the original languages that STT was developed for are being found to be reliable the field of STT is being paired with other disciplines and technologies to expand its use. One example of this is it is being tested with psychological experiments [1].

Through this testing, it is being found that STT can keep up with speech at such a reliable rate that it is safe to use in experiments. Another example is it being applied to create a more accessible environment for people who are deaf or struggle to speak. STT has been tested in doctor’s offices to pick up what the doctor is saying on their smartphone and display it on the screen so deaf patients can know what they are saying [9]. This relieves doctors of having to write down or type what they would like to communicate and decreases the amount of time it requires to communicate with the patient. STT is also being used to help people with dysarthric speech [4]. “Dysarthria is a motor speech disorder resulting from damaged peripheral or central nervous system and causes slow speaking rate, pronunciation deviations, and prolonged pause interval between words and syllables.” [4] By using STT the user can speak, and the program can remove the parts of speech that are not needed and combine all the speech from throughout speaking to aid in faster communication to the other party. STT is also being paired with machine translation (MT) to be able to give instant translations of the speech provided [7]. In a study, it is being used to automatically convert the speech in YouTube videos to text and then using MT to translate it with proper punctuation. In another study, it is being used in classes to translate English lectures to Spanish, and then using eye-tracking technology, they monitor if the student is paying attention to the auto-translated text or the video [5].

1.2) Research Motivation and Objectives

Currently, if someone were to want to write programming code using a Speech to Text service it would not format the text correctly to be understood by the compiler. For example, if someone were to want to write the beginning of a for loop, typed out it would look like “for(int i = 0; i < n; i++)”. If a person were to read this naturally it would be stated as “For int I equal to zero while I is less than N I plus plus”. This does not match how it should be written and for it to be written as such it would have to be stated as “For open parenthesis int I equal sign zero semicolon I less than n semicolon I plus plus”. This still does not guarantee spacing, or capitalization would be correct. The natural way to read code does not exactly match what is written down. Currently using a STT application would also require having the code typed in a third-party application. This causes the burden of both having to then transfer the text to a text editor or Integrated Development Environment (IDE) and allows for the security issue of having code in an environment that it should not be in. Thus, the focus of this study is to research into what has already been accomplished in the field of Speech to Text in relation to usage in programming and the general application of Speech to Text in other fields, how people speak code aloud, and how a formatting application created for this study performs. What is found can then be applied to furthering the application of Speech to Text in relation to coding.

The rest of this paper is as follows; section 2 will be over researching Speech to Text. Section 3 is the methodology to be used in testing how people speak code, Apple’s Speech to Text, and the formatting application. Section 4 will be the results of the testing. Section 5 will be conclusion and future work.

CHAPTER II:

LITERATURE REVIEW

2.1) Research Design

Research was done by first understanding what the general goal of the search would be. Then by starting with a broad term for the desired theme of articles, a search was conducted through literary databases to gather all articles that had an initial impression of relevancy. From there, the articles were reviewed in a team effort by the authors to remove the articles not relevant and keep the articles that were relevant. This led to the final step of detailed review of the articles to pull the necessary data that would give an understanding of the field in question and help guide the methodology used to conduct an experiment.

2.1.1) Research Design

To guide the research done in this study, three research questions were created. They were created to be general to the point that many diverse themed articles could be found while remaining specific to the point that it remained on the topic. The research questions are as follows:

RQ1) What is the state of modern Speech to Text technologies?

RQ2) How is Speech to Text technology being used?

RQ3) How does Speech to Text handle a user's accent?

The first question was written to give a basis for the capabilities of modern Speech-To-Text. It would guide the research to see how Speech-To-Text is viewed, what work is currently being done to improve it, and how accessible is it to the public. This information is necessary to give an understanding to the research community of how far

Speech-To-Text has been developed to thus be able to give new ideas as to where development can go forwards.

The second question was written to give an understanding of how Speech-ToText functionality is being viewed. It would guide the research to see the different fields that Speech-To-Text is being applied. This allows the research community to know how far Speech-To-Text has spread in application. Then with this information the research community can find ways of deepening the research done in these fields and find ways of applying Speech-To-Text in fields not yet considered.

The third question was written to give a basis on one of the potential issues of Speech-To-Text. Accents vary across the globe and even though a system is built and can easily understand the creator’s voice, it may not be able to understand a voice of someone from a different region who speaks the same language with an accent. This information is useful to the research community to be able to build applications that are inclusive to all speakers and can be used worldwide.

2.1.2) Research Steps

To ensure a structured systematic review was conducted the following steps were followed:

1. Planning Phase
    a. Understand which topic was to be the focus of the research
    b. Note the key phrases associated with the topic
    c. Locate the databases to be used in finding articles
2. Research Phase
    a. Gather articles that had the possibility of being relevant to the topic

   b. Review the articles in an iterative process that started general and became detailed through iterations
   c. Read the chosen articles in depth
3. Writing Phase
   a. Understand the general structure of the paper to be written
   b. Compile notes from the chosen articles
   c. Write the systematic review

2.1.3) Literature Search

During step b of the planning phase of research a list of phrases and terms were compiled to be used in searching for articles relevant to Speech-To-Text. The following terms were compiled and used in the search:

“Speech to text”

“computer speech recognition”

“spoken word recognition”

”speech to text accent”

“Speech to text” was chosen given it is the overall topic and so this will yield a broad range of articles. “computer speech recognition” was chosen given it is a critical piece of the Speech-To-Text workflow. Thus, this could potentially bring in articles about Speech-To-Text along with articles over this specific part of the workflow which is still good information to understand. “spoken word recognition” was chosen as finding a term with a different perspective of what speech is to potentially yield a greater number of articles. These three terms were chosen to specifically answer research questions RQ1 and RQ2.

“speech to text accent” was chosen to find articles that were of the topic SpeechTo-Text and specifically mentioned accents. In this study accent is referring to the way a person pronounces a language, not to be confused with accent as in the symbol above a letter to indicate a different pronunciation. This information could help researchers understand how accents are currently being viewed and handled in the research community. This term was chosen to specifically answer research question RQ3.

2.1.4) Study Selection

When beginning the research phase three restrictive pieces of search criteria were set as a basis for research to restrict articles to a narrow selection that were most relevant. This first piece of search criteria was that the article must have been peer reviewed. This was decided to ensure the information in the article was accurate and high quality. The second was the article must be available online. This was decided for research ease due to all articles would be available at a moment’s notice. The third search criteria were that the article must have been written between 2015 and 2021. This was decided to ensure all information would be current in the field.

2.1.5) Paper Selection and Filtering Process

2.1.5.1) Article Collection

Collecting articles to be reviewed was done through “OneSearch” database search engine. “OneSeach” allows users to search all the databases the university has access to of all types of media. This was chosen versus going through individual databases as it would yield more results in a shorter amount of time. Once an article from the search was selected the user is directed to the article on its home database.

The chart below illustrates the search terms used and how many articles were collected from each of the terms. Note that along with the search term, the other search criteria were that it must have been a Peer Review article, available online, and published between 2015 and 2021.

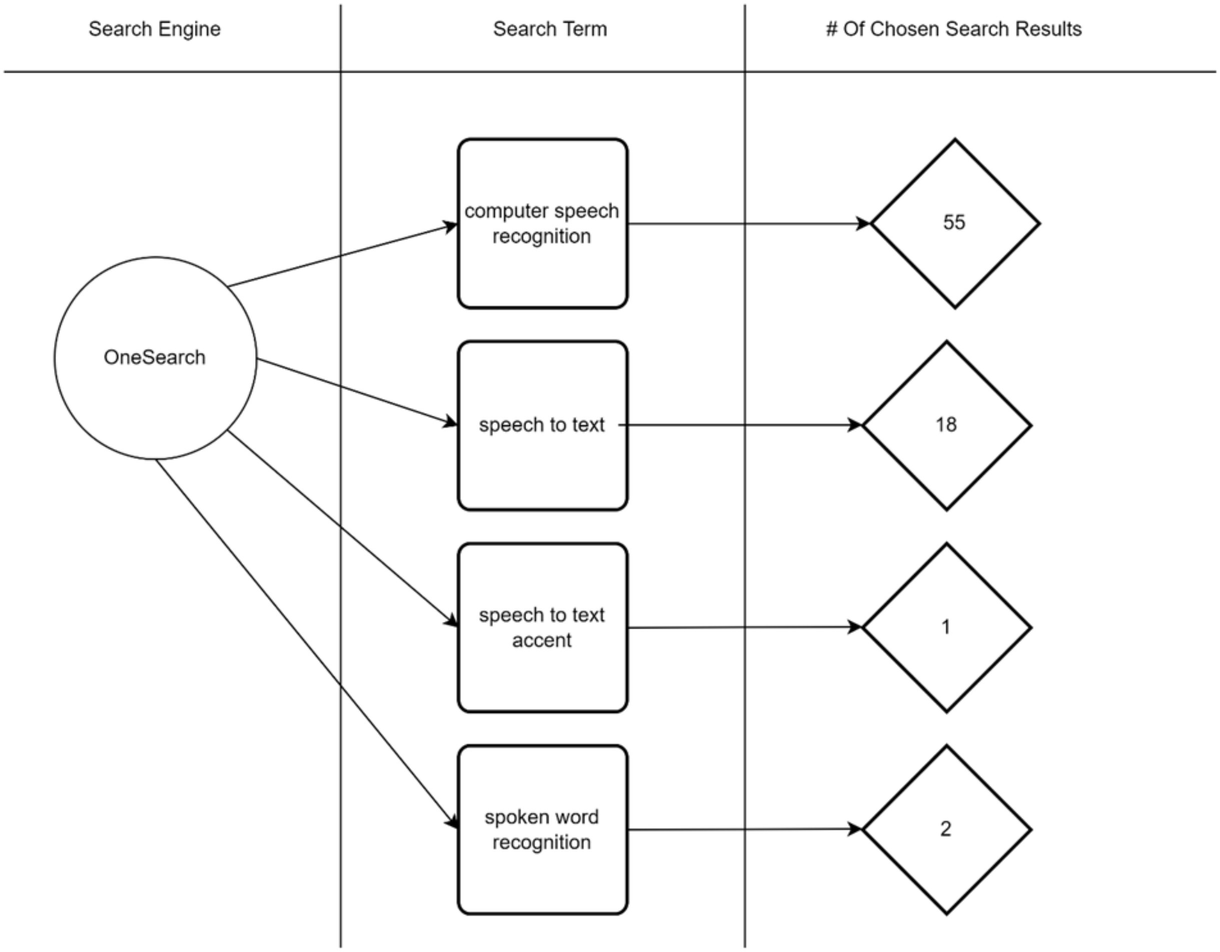


Figure 1.
Article Search

2.1.5.2) Title Review

Once several articles that seem suitable were gathered the next step was conducted. This step was to review the titles of all the articles in detail to ensure it seemed to reflect a consensus with the theme of speech to text applications. This was done by downloading all article's information into an excel spreadsheet. Then a column was added for each author and one column for a consensus decision. At this point one author reviewed every article and noted whether they thought the article was ‘Relevant’, ‘Irrelevant’, or if a decision could not be made then ‘Maybe’. Once completed the second author did the same. Afterwards, both authors met to discuss their decisions. If both were

‘Relevant’ or ‘Irrelevant’ then the final decision was the same and in the consensus column the same decision was noted. If the decisions did not match each other or both were ‘Maybe’ then the authors discussed why they made the decision and the strength and weaknesses of including or excluding it. Once a decision was made it was noted in the consensus column. At the conclusion of this review, four articles were deemed to be irrelevant and seventy-four were deemed to be relevant and were moved to the next stage of review.

2.1.5.3) Abstract Review

This step was to review the abstracts of all the articles in detail to ensure it seemed to reflect a consensus with the theme of speech to text applications. This was done by creating a new excel document with the information of all the articles that made it past the title review stage. Then in the same manner as before, a column was added for each author and one column for a consensus decision. One author would then review the abstract of each article and note whether they thought the article was ‘Relevant’,

‘Irrelevant’, or if a decision could not be made then ‘Maybe’. Once completed the second author did the same. Afterwards, both authors met to discuss their decisions. If both were

‘Relevant’ or ‘Irrelevant’ then the final decision was the same and in the consensus column the same decision was noted. If the decisions did not match each other or both were ‘Maybe’ then the authors discussed why they made the decision and the strength and weaknesses of including or excluding it. Once a decision was made it was noted in the consensus column. At the conclusion of this review, twenty-eight articles were deemed to be irrelevant, forty-four were deemed to be relevant, and two were found to be duplicates. The articles deemed relevant were moved to the next stage of review.

2.1.5.4) Article Review

The final step of the review process was to review the entire articles in detail to ensure that they did reflect in their entirety the consensus of speech to text applications. This was done by creating a new excel document with the information of all the articles that made it past the abstract review stage. Then in the same manner as before, a column was added for each author and on column for a consensus decision. In addition to this, two other columns were added, one for each author to leave notes on why they felt the article deserved the decision made. One author would then review the article and note whether they thought the article was ‘Relevant’, ‘Irrelevant’, or if a decision could not be made then ‘Maybe’ as well as leave a note explaining their decision if they chose to do so. Afterwards, both authors met to discuss their decisions. If both were ‘Relevant’ or ‘Irrelevant’ then the final decision was the same and in the consensus column the same decision was noted. If the decisions did not match each other or both were ‘Maybe’ then the authors discussed why they made the decision and the strength and weaknesses of including or excluding it. Once a decision was made it was noted in the consensus column. At the conclusion of this review, thirteen articles were found to be irrelevant and thirty-one were found to be relevant. Thus, there are thirty-one articles that will be analyzed in this paper.

2.1.6) Data Extraction and Synthesis

After all articles were reviewed, it was found that many of the studies used similar technologies for their applications. The rest of this section will go in depth into these similarities. It will first go over the focus of the studies in section 3.6.1. Then it will review the technologies used in section 3.6.2. And finally, it will review their findings in section 4.

2.1.6.1) Content Type

After reviewing all articles if was found that many had similarities in the focus of their studies and could be grouped into three categories:

- Performance: The focus of these articles was to design a new Speech to Text system to improve performance. Some were focused on the performance for certain kinds of users while others were just attempting to improve the general performance.
- Application: The focus of these articles was to take Speech to Text and use it. It was applied to many different fields in various methods using various technologies which will be discussed in section 3.6.2
- Neither: These articles had a focus outside of either technology or application/performance in general. While these are still good for reference for other parts of this paper, they will not give us information into the specific topic.

These categories of articles will allow us to further investigate the specifics of what the articles have to offer. The initial sorting is general, but we will dive further into categorization in the following section.

2.1.6.2) Application Types

The application types are further categorized into the following set:

- Web: These applications were created to be run in a browser over the internet.

- Android: These applications were designed to be run on an Android device (Smart phone, tablet…). This is to allow the application to be on the user's mobile device
- iOS: These applications were designed to be run on an Apple device (iPhone, iPad…). This was also to allow the applications to be on the user's mobile device.

There were also some applications that were created to be run on both Android and iOS. From these categories we can see the availability of the applications. The mobile applications were created to be on the user's device that would go with them wherever they went while the web applications were a mix of designed to be accessed on a mobile device or meant to be on a computer that had to be used while stationary.

2.1.6.3) Speech to Text Dictation Types

The Application category can also be broken down into the Speech to Text dictation tools used in the project. The categories are as follows:

- Google API Products: There are multiple studies that used various Speech to Text APIs created by Google. Many may be the same with just a different naming convention used in the article.
- Nuance SpeechAnywhere API: There is one study that uses this API. While alone in this collection of articles it does show diversity in API's.
- Dragon Dictation: There is one study that uses this tool. While alone in this collection of articles it does show diversity in API's.

- Siri: There is one study that uses this dictation tool. This does come as a surprise given it is built into all iOS mobile devices and is therefore widely available to the public.

This gives insight into the choice's studies are making for their technologies. It shows what is most popular in usage and displays the diversity of dictation applications available.

2.2) Results

The following section will analyze the categories found in section 3.6 and pull the data from them to answer the research questions.

2.2.1) Quantitative Analysis

This section will give the numbers found relating to each of the found categories.

2.2.1.1) Final Set

Below is a list of all the articles deemed relevant for the research that was found during sections 3.3, 3.4, and 3.5. This section corresponds to the articles listed in Appendix A, but here we give more detail about each article in relation to the data found in them.

Table 1.
Articles Mapped to Content, Application, and STT Dictation Types

Article ID	**Content Type**	**Application Type**	**STT Dictation Type**
S1	Neither	N/A	N/A
S2	Performance	N/A	N/A
S3	Application	Web	Nuance SpeechAnywhere
S4	Performance	N/A	N/A

S5	Application	Web	Google Translate
S6	Application	Web	Google Speech To Text
S7	Performance	N/A	N/A
S8	Application		
S9	Performance	N/A	N/A

Article ID	**Content Type**	**Application Type**	**STT Dictation Type**
S10	Performance	N/A	N/A
S11	Application	Android	Google Speech API
S12	Neither	N/A	N/A
S13	Performance	N/A	N/A
S14	Performance	N/A	N/A
S15	Performance	N/A	N/A
S16	Performance	N/A	N/A
S17	Performance	N/A	N/A
S18	Application		
S19	Performance	N/A	N/A
S20	Neither	N/A	N/A
S21	Application	Web	Google Cloud Speech API
S22	Performance	N/A	N/A
S23	Performance	N/A	N/A

S24	Performance	N/A	N/A
S25	Application	iOS	Dragon Dictation
S26	Application	Android	Google Cloud Speech API
S27	Neither	N/A	N/A
S28	Application	Android	Google Online Speech Recognition
Article ID	**Content Type**	**Application Type**	**STT Dictation Type**
S29	Application	iOS and Android	Siri and Google
S30	Performance	N/A	N/A
S31	Application	Web	Google Cloud Speech API

2.2.1.2) Categorized Results

2.2.1.2.1) Content Type

The term “Content Type” refers to the focus of the article whether it was about performance, application, or neither. Performance in this case is about creating their own Speech to Text system to further improve the performance of Speech to Text. Application in this case is taking Speech to Text and applying it to a field to be used. Neither means that the article was not about the previously listed focuses but still had valuable information. In total thirteen papers were about performance, fourteen were about application, and four were about neither.

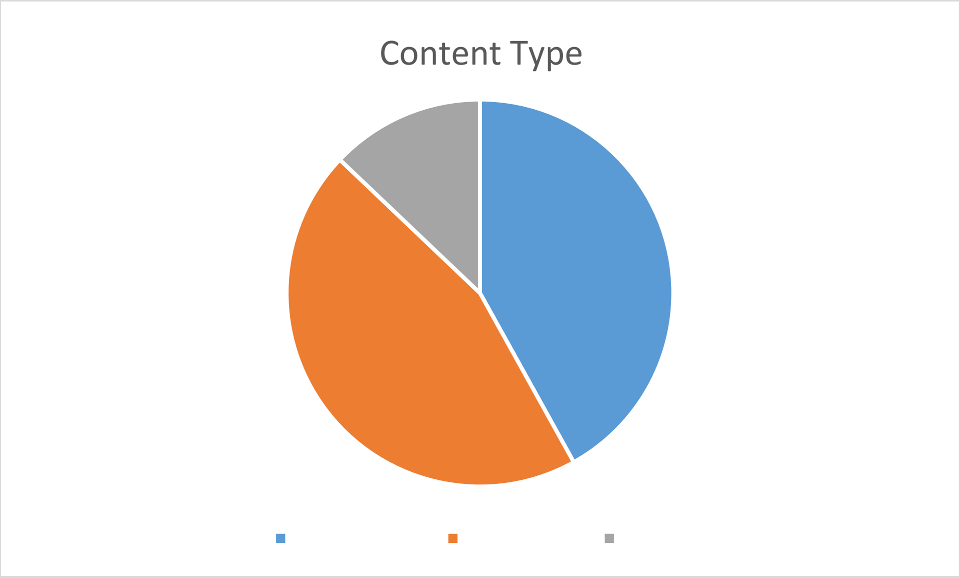


Figure 2.
Content Type

2.2.1.2.2) Application Type

“Application Type” is what type of application was created that uses Speech to Text, either Web, Android, iOS, Raspberry Pi, or iOS and Android. This was noted to understand what was most popular in usage, what was most accessible, and what users preferred. It was found that six used web, three used android, one used iOS, one used a Raspberry Pi, and one used iOS and Android.

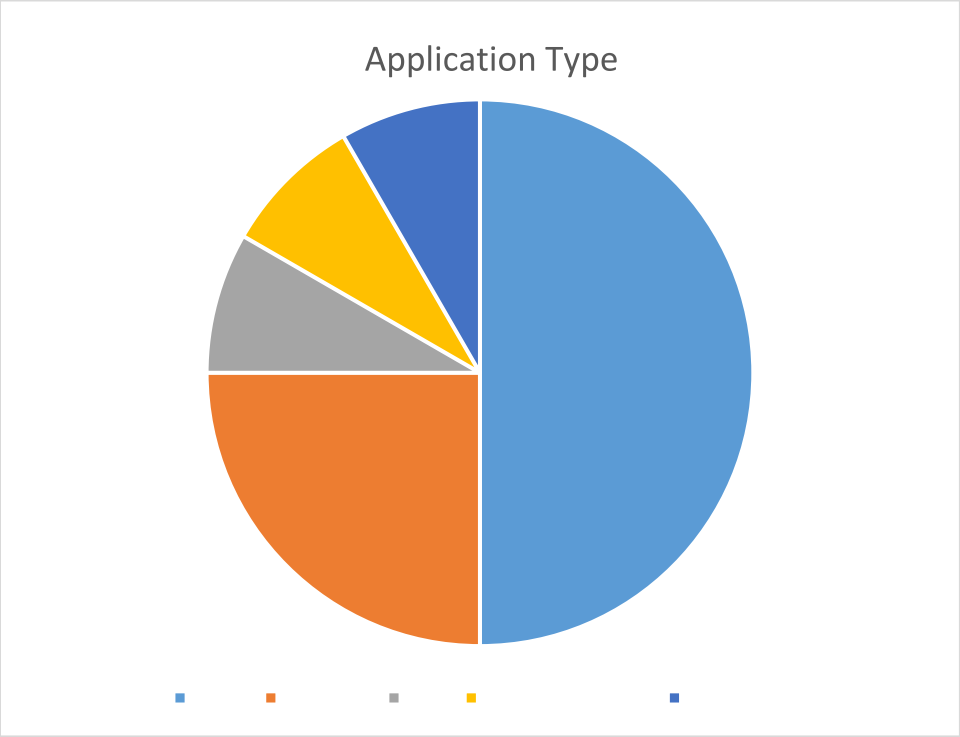


Figure 4.
Application Type

2.2.1.2.3) Speech To Text Dictation Types

“Speech To Text Dictation Types” refers to the Speech to Text dictation tool used that took an audio input and returned a transcribed text output. An interesting finding in this was the diversity of Google tools used. It seems like these were most likely the same product listed as different names. Thus, all the Google products will be categorized as on entity. It was found that one study used Nuance SpeechAnywhere, eight used Google, one used Dragon Dictation, one used Siri and Google, and one used Google, Watson, and Azure.

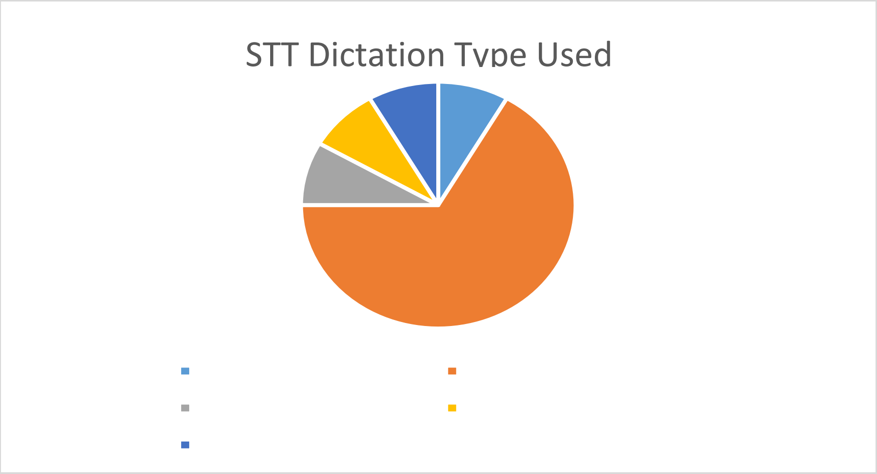


Figure 5.
Dictation Type Used

2.2.2) Article Notes

This section will give notes deemed important over the articles found in the literature search.

Table 2
Article Notes

Article ID	Notes
S1	This focus of this study is to find if people use accents to know the definition of a word. To accomplish this, five experiments were held. In each experiment a word with an ambiguous definition would be stated in a certain accent and the participant had to give the definition. In general, the study only focused on American and British accents and had only British or American participants. Experiment one was a mix of British and American participants while experiments two, three, four, and five were focused only on British participants. Testing was done in multiple experiments as they used the results

Article ID	Notes
	of the previous experiment to determine the direction they should focus on in the following experiment. At the conclusion of the study, it was found that listeners do use accents to understand the definition of a word that the speaker is stating.
S2	In this study, they research interacting with computers using vocal commands. The steps for how their system works are as follows: 1) First they create a database of all the required words. Then they have the user record their voice saying these words. This is done so that the system compares the command given directly to the users recording. The two ways they attempt to match inputs to the database (also known as feature extraction) are using double thresholds voice activity detection (VAD) with Mel-frequency cepstral coefficients (MFCC). To set up the database they had the user record a command word ten times. Then they applied a preprocessing to the recordings to clean them up and then saved them to the database. Another interesting note is that they created an avatar for the users to speak their commands to that has a human voice. This was done to provide a relaxing environment for the student, like a real teacher. Commands were given based on a question answer approach where the avatar would ask questions about what the student wanted to do until the desired outcome was reached.
S3	Using front-end automatic speech recognition system (ASR), speech can be automatically converted to text. Without a front-end ASR system either someone must manually type the text, or a slow backend system would have to convert it. In this study, they attempt to create a web-based ASR system for

Article ID	**Notes**
	clinical documentation for German speakers. They method used to get participants for the study was: • Physicians were asked to participate in morning meetings • Two were asked to participate via personal communication • Enrollment was open for 30 days • The requirement was that the physicians had to have clinical activity and document at least two reports during the study In the study to begin, the participants had to document texts first by speech and then by typing. Participants conducted documentation through a browser. Every time a participant created documentation the information recorded was: • Length of the text • Length of time taken to document • Number of corrections made using the keyboard • The mood of the participant The decision of whether the participant was allowed to use speech and type or just speech was randomized every time the webpage was loaded. The mood of the participants was collected by asking each participant to select one of three smiley faces indicating a mood. The speed of input was calculated by characters per minute. The data excluded were documents with more than 1000 characters per minute, document input of more than one hour documentation time, and documents of less than ten characters. The speech to text system used was Nuance SpeechAnywhere services. There were 28

Article ID	Notes
	participants total, all of which were German speakers. Findings were 79% of the participants were faster using speech to text, the number of corrections made were less for participants using speech to text, and more characters were recorded when using speech to text.
S4	This study aimed to create their own speech to text system to be used in a power grid dispatch system controlled by speech.
S5	What dives this study is the thought that people need to understand the differences in cultures so that when cross cultural interactions occur there are not any accidental issues. The language barrier is the biggest issue preventing cross-cultural learning. This study's research is to demonstrate that speech to text recognition (STR) and computer-aided translation (CAT) are accurate enough now to be able to aid in cross cultural interactions and to bring attention to them since there has been a lack of attention in the past. The study consisted of 21 participants from 13 countries who spoke their first language resulting in 10 languages that were included. Recorded in the study was the accuracy rate of STR and CAT, issues with STR and CAT and their possible solutions, and if STR and CAT are developed enough at this time to aid in cross-cultural learning. The study was done in four one-week steps. The first week the participants introduced themselves, hobbies, and interests. The second week they each presented one tradition. The third week they took part in someone else's tradition that was presented and the presented their experience with it. These first three weeks were done asynchronously in a chat. The fourth week they met face to face online to discuss the previous

Article ID	Notes
	weeks. The participants used Google Translate STR to talk and then CAT to translate their language into English. All participants were able to understand English and so that is why it was chosen as the median language. Data was collected through online communication and interviews. The lowest STR accuracy rate was for English at 93.94%, but this was attributed to the strong accent of the participant. The highest STR accuracy rate was for French and Hindi at 98.51% which are commonly spoken languages. The lowest CAT accuracy rate was for Mongolian at 94.37% and Filipino at 94.60%. The highest CAT accuracy rate was for Spanish at 98.15%, Russian at 98.02%, and French at 97.95%. This study considered 85% accuracy to be the minimum for STR and CAT to be considered useful and this was met which was also backed by the participants stating that both were useful. The STR and CAT accuracy rates fluctuated from step to step in the study which is thought to be caused by easier words being used in the beginning and as the study progressed the participants had to use more specific terms not found in the CAT database to describe their cultures. The most reliable browser for Google Translate was Google Chrome. The following were the issues and their solutions that were found by participants: STR did not add punctuation – The participants added them manually or stated the punctuation STR changed some Traditional Chinese characters to Simplified Chinese – participant corrected it manually

Article ID	Notes
	STR did not correctly transcribe some names or specific terminology – Participants attempted to state the name or term again and if it did not work then they manually changed it If the input had multiple languages, then the STR system would not correctly recognize the entire input, only part of it – They would either manually change the output or manually change the expected language when they were about to use a word from a different language If the participant paused for a long period and then resumed speaking, then the STR system would remove what was initially transcribed and replace it with what was stated after the pause – The participants would ensure there were no pauses or prepare a script before speaking The STR system would not recognize some words – The participant corrected the output manually The STR system would not correctly recognize some words that sounded similar – participants corrected the output manually
S6	The study aimed on using speech to text to measure how intelligible the speech of a patient with Parkinson's Disease (PD) is. PD is a movement disorder where voice problems tend to be the first symptom to present itself due to the complex nature of speaking. In the study a software called Voxtester was created which used Google Speech to Text for its speech to text (STT) system. Steps for their STT system were to first record input, use STT to create text, and then put text into their software for word matching. Their reasoning for choosing Google Could Speech API is it is easy to use in

Article ID	Notes
	development and handles any preprocessing to the audio. The study chose to ignore words with four or less characters and words with accents. The first round of testing was done on a group of fifteen young and healthy people to have as a basis for comparison. The second round of testing was done with twenty-two healthy elderly people. The third round of testing was done with twenty-eight people with PD. All participants were asked to conduct reading exercises in a quiet room while 15-25cm away from a microphone. The study concluded STT was shown to help with recognition.
S7	This study attempted to create their own speech to text system specifically for recognizing Japanese vowel length to be used in Computer Assisted Language Learning. Japanese vowels have long and short versions. This creates a complexity when attempting to understand the meaning of words as it is not just the length of the vowel that determines the meaning but also the vowels around it, frequency, and intensity. Currently speech to text systems do not take into consideration the time length of phenoms which is what the paper attempts to do. In their study they had one hundred and four participants.
S8	This study was conducted to create a speech to text (STT) application to be used by students for practicing speaking new languages. Through this the study will understand the strengths and weaknesses of using STT. The study used the ADDIE model which stands for: analyze, design, develop, implement, and evaluate. Initially questionnaires were sent out to ask about student's difficulties with practicing speaking a new language. The issues in

Article ID	Notes
	implementation were that the area the application was being tested was too loud and caused the STT system to not recognize the words spoken. A notable distinction to make is voice recognition focuses on differences between two people's voices where speech recognition focuses on recognizing words given. The conclusion of the study was STT can be used for practicing speech in a new language.
S9	Code-switching is when a person speaks multiple languages during a single conversation. Intersentential code-switching is when this happens between sentences and intrasentential code-switching is when this happens during sentences. This is common in South Africa where most residents know more than one language. This study is to create a speech to text (STT) system for code witching in five South African languages. The material used for study were South African soap operas. In the study, bilingual is considered knowing English and another language. First, they created four bilingual STT systems for isiZulu, isXhosa, Sesotho, and Setswana. Then they created on system to handle all five languages. It was found that the more training data inputted into the system the better it was able to recognize words. Word error rates were still found to be high. The best system had a word error rate of 26.3% for English and between 52%-63% for the South African language.
S10	The focus of this study is to compare how well humans can fill in missing information from partial audio versus how well a speech to text (STT) system is able to fill in the missing information from partial audio. Through creating their own STT system it was found that adding background noise aids

Article ID	Notes
	the STT system in understanding partial audio when the noise is not at an overwhelming level.
S11	The objective of this study was to create an Android app to help people memorize the Qur'an using speech to text. The method to determine the similarity between an input and the intended string was calculated using the Jaro Winkler Distance Algorithm. The Fisher-Yates Shuffle algorithm was used to randomize text given to the user. Google Speech Recognition API was used as the speech to text system. It was found that this API was not able to recognize all the letters from the Al-Quran. Although, the API did have a 91% accuracy, so it was concluded that it performed well.
S12	The focus of this study was to create an application to aid second language learners in word recognition from speech. The participants listened to a monologue and attempted to translate it by hand into a web application. Each pre and post test consisted of 60 words- 32 common words and 28 were from the Academic Word List. The odd words appeared in the monologues and the even ones did not.
S13	Study focused on creating their own speech to text system for Lithuanian.
S14	This study focuses on creating a speech to text application to understand Standard Yoruba using a syllable-based approach. Yourba is one of the three major languages of Nigeria. This application was created using Hidden Markov Model Toolkit on Windows using Java. Participants spoke 25 bi-syllable words and 25 tri-syllable words. Preprocessing was done on the

Article ID	Notes
	input audio files. They system had an accuracy of between 76% and 84%. The study concluded that the system was promising.
S15	This study focused on creating a speech to text system for hearing impaired users.
S16	Automatic Speech Recognition (ASR) research began in the 50’s at AT&T. There have been multiple approaches to ASR in the past including template-based, knowledge-based, neural network-based, dynamic time warping-based, and statistical-based. This study focuses on ASR systems already created. The five major ASR systems being evaluated here are Google Assistant from Google, Alexa from Amazon, Siri from Apple, Cortana from Microsoft, and Watson from IBM. The major Software as a service ASR systems are Amazon Transcribe from Amazon, Azure Cloud Service from Microsoft, Watson Cloud Services from IBM, and Cloud Speech-to-Text from Google. Google released the first ASR system. This study focuses on using ASR for Romanian. In the end the decision was made to test Google Speech Recognition. To do so 20 YouTube videos to test the API were chosen using the first minute of each video. The word error rate was calculated using: WER = (Substitutions + deletions + insertions) / total number of words. The study took background noise and clarity of speech into consideration when understanding the WER. They found the WER to be 30.96% with the lowest being 9.96%.

Article ID	Notes
S17	This study focuses on finding the best option for speech to text (STT) in Spanish that can run on small computers such as a Raspberry Pi. Choices were made to use a STT API to reduce time on project and second was to use Google Web Speech API, Watson, and Azure Speech Service. Python has “SpeechRecognition” library to support the STT API’s. Google’s API are accurate due to their growing neural networks. It was found that Google was the fastest and IBM was the slowest, with an interesting note that female voices were converted faster than male voices.
S18	In this study, an application was created that detects difficult words in captions and displays them while not displaying easy word to aid in computer assisted language learning. The system determined difficult words based on their “speech rate of the words, their frequency, and specificity”. It also tested users to see what information they retained and the words that were incorrect (ASR errors) were thus the difficult words.

S19	The process YouTube uses to create translated captions is first it uses a Speech to Text (STT) system to convert it to text. It then uses Machine Translation (MT) to translate the text into the requested language. The issue found with this is that it adds punctuation by pauses in speech which can make multiple sentences be seen as one sentence which then causes issues with translation. This study attempts to create an application to add punctuation to the STT created by YouTube using neural networks. For testing they used 27, 826 subtitles. In preprocessing, they set all text to lowercase, changed all ending punctuation to periods, and removed all other punctuation. The outcome of their application was 70.84% accuracy.

Article ID	**Notes**
S20	This study was to see if students paid attention to the automatically generated speech to text captions for lectures. This was done by watching where their eyes were focused on to see if they preferred the captions or slides on screen.

S21	This study focuses on using speech to text (STT) to transcribe what is stated in psychological experiments where speech is what is being analyzed. In the study Google Cloud Speech API was used. The steps of the experiment were: 1) had participants study a list of random words, 2) had participants verbally recall the words, 3) had human transcribers write what was said, 4) had Google Cloud Speech API transcribe what was said, 5) compared the transcripts. The outcome is that the transcripts matched to a high degree and that STT would work in psychological experiments. The criteria for the transcription were it had a high hit rate, low false alarm rate, and speech onset times should match. The issue would with STT was that the human transcriber could disregard eh spoken errors, but the STT system would still attempt to transcribe them. Another issue found was that people could purposely create audio that would be transcribed incorrectly to falsify the accuracy of the system.
S22	The focus of this study was to expand automatic speech recognition to have better results for children due to the differences in the way children speak such as the higher pitch and errors in speaking. This was done using a sequence-to-sequence model.

Article ID	**Notes**
S23	This is good for showing the different types of applications that have been done.

S24	This study focuses on the difference in speech patterns between the elderly and younger people. They then take what they learned and create their own automatic speech recognition system to create better accuracy for elderly users.
S25	This study aims to improve communication for deaf people, especially recently deaf people that do not know sign language and prefer to have written communication. The participants were medical students and doctors. They were given six sentences that were common in medical communication. They were then told to first write down the sentences, then type them, and then to use Speech to Text (STT) to have it transcribed. Typing was done in Word 2003 and STT was done using Dragon Dictation on iPhone 4. STT was found to be faster than writing or typing, but it was found to be less accurate than writing or typing. The accuracy did improve as the user talked more.
S26	This study focuses on creating an application to aid communication between speaking and deaf people. This is due to their though that writing messages down for deaf people is inefficient and ineffective in practice. They used Google Cloud Speech API as their speech to text system, waterfall model as their project creation model, and created an android app. If was found that speech impaired voice recognition was 80% and non-impaired was 100%.
S27	This is a good reference for history of speech to text.

Article ID	Notes
S28	This study focused on creating an Android app to help deafblind people in performing prayer, specifically Namaz which is a required prayer in Islam. Activities of daily living (ADL) devices have been created but disabled and elderly people still struggle to use them. This app will use automatic speech recognition to recognize stages of Namaz and then use vibration to communicate to the user what stage is being performed. There were six participants in the first experiment and fifteen in the second. They gathered base data by three participants recording five occurrences of actual prayer. The app knows when to being monitoring for prayer based on GPS location and time. The study was done in Arabic and used Google online speech recognition. The recorded noise level in the testing room was 23db found by using Sound Meter App. The automatic speech recognition was found to be 93.75% accurate.
S29	This study tests built in Voice Input (VI) features of smartphones and how they can be used in web surveys. They give statistics on Siri and Google usage. They had participants take web surveys only on iOS or Android devices. There were 1205 participants who each answered a maximum of 37 questions. A control group of people were allowed to use their keyboard and a control group could use VI. There was a failure rate of 3.3% in the IOS VI versus the typing groups of 3.0%. It was also found that iOS had shorter answers and less usage of VI.

Article ID	Notes
S30	This study focused on expanding automatic speech recognition to have better results for children due to the differences in the way children speak such as the pitch and pronunciation.
S31	This study focused on creating a Speech to Text (STT) system for giving commands to drones. They used Google Cloud Speech and had a dictionary of 48 commands which were a mix of English and Spanish. The dictionary mapped to 9 actions for the drone done by inputting the audio to Google, receiving the text output, and then mapping it to a command in the dictionary. Spanish performed better than English, but this is attributed to the participants main language being Spanish. High accuracy was found with issues lying in the network connection.

2.2.3) Qualitative Analysis

In this section, the research questions in section 3.1 will be answered.

2.2.3.1) RQ1 – What is the state of modern Speech to Text technologies?

What has been found is that Speech to Text is still very much a work in progress but is making great strides in progress since the very beginnings of transcription as shown in S27. The specific area that is being worked on the most is not sending recorded audio to the Speech to Text converter or sending text from the converter back to the requesting program but rather converting the audio to text.

This is being analyzed and revised in many ways each depending on the proposed solution in each study. In S2 the proposed methods are double thresholds VAD with MFCCs or Wavelet-based MFCCs for this “feature extraction” process. Study S10 attempted to improve Speech to Text by adding background noise to the input which was successful when the two were balanced properly. In study S19, it was attempted to improve Speech to Text punctuation by using Recurrent Neural Network. Study S22 attempted to improve Speech to Text specifically for children and did so by using a sequence-to-sequence model while S30 focused on children as well but used a preprocessing system. S22 was not the only one that focused on age group specific vocal patterns. S24 focused on elderly speech recognition and used a preprocessing system on the input audio as a way of improvement.

The way these new methods of Speech to Text are being tested vary widely too. In S4, their system is tested by creating a power grid dispatch system that is controlled by speech. Study S15 tested their new system by using to aid hearing impaired users.

There was also a great diversity in the languages focused on for specific Speech to Text systems. The study in S9, they created their STT system to handle five South African languages (English, isiZulu, isXhosa, Sesotho, and Setswana). They accomplished this by first creating four Speech to Text systems that handled two languages (English paired with one of the other four) and then creating a Speech to Text system for all five of them. Lithuanian was the focus of article S13's Speech to Text system. Study S14 focused on Yorùbá which is one of the three major languages of

Nigeria. Study S23 tested their new Machine Translation system by translating English to German. Many of the studies specifically took diversity of languages a step further and focused on creating their Speech to Text specifically for usage in Computer Assisted Language Learning (CALL). The study in S7 created a new Speech to Text system specifically for recognizing Japanese vowel length.

The overall takeaway from these articles is the diversity of them. It was seen how many different methods are being used to attempt improvement of Speech to Text. It was also seen how diverse testing these new systems are that span controlling and aid all the way to language specific work and education.

2.2.3.2) RQ2 – How is Speech to Text technology being used?

What was found was just like in section 4.2.1 there is a great diversity of ways Speech to Text is being used. Multiple studies were found to be using Speech to Text in the medical field. S3 created a web-based application for German medical workers and found that it was successful increasing the speed of text entry by 79%. Study S6 used it in an application called Voxtester to measure how intelligible the speech patterns are of

German speaking patients with Parkinson's Disease. S21 used it in preparations to evaluate if Speech to Text is ready to be used for transcription in psychological experiments. It was found that the human transcription and Google Cloud Speech API transcription matched to a high degree where the main issue was that spoken errors were disregarded by human transcribers while the Speech to Text system would attempt to transcribe it. S25 used it to help communication for deaf people, especially recently deaf people that do not know sign language and prefer to have written communication. S26 also focused on aiding communication between deaf and speaking people by creating an Android app that uses Google Cloud Speech API.

It was also found that multiple studies focus on using Speech to Text for education. Beginning with a basis shown in S12 where computers are already being used for language learning through Computer-Assisted Language Learning to help second language learners we then see this further extended using Speech to Text. Study S5 used Speech to Text as a method of instant computer-aided translation to allow 21 participants from 13 countries learn about each other's cultures through multiple virtual gatherings. S8 used it for Computer-Assisted Language Learning for students to practice speaking new languages. Similarly, S18 used it for Computer-Assisted Language Learning by creating an application that detects difficult words in captions and displays them while not displaying easy words. Further work in this area is already being done such as in S20 where it is being studied how well students pay attention to the Speech to Text captions. A couple of studies were also found to use Speech to Text for religious purposes. Study S11 uses speech to text to create an Android app that will help users memorize the Qur'an through memorization and random reciting. S28 created an Android app to help deafblind people in performing prayer, specifically Namaz which is a required prayer in Islam and found it to be 93.75% accurate.

S16 used it to test how well Google's Speech Recognition API works for Romanian by comparing it to YouTube videos and found its Word Error Rate to be at lowest 9.96%. S17 attempted to find the best option of running it on a small computer like Raspberry Pi and found that Google's API was the fastest while IBM's Watson was the slowest. S29 used it to help raise the number of responses for web surveys which was only successful on Android devices and not iOS. Study S31 used it to give commands to drones using Google Cloud Speech.

The overall takeaway from these articles is like in the last section, diversity of application but also the current success rate of Speech to Text. In these settings it was found that Speech to Text could be used successfully for these fields and thus can be applied to even more fields.

2.2.3.3) RQ3 – How does Speech to Text handle a user's accent?

What was found to answer is that understanding a user's accent that is not native to the language being spoken is still receiving high error rates and requires further work. Article S1 gives a good background on the way words spoken with an accent can be perceived. The outcome of their experiment proves that listeners do use accents to understand the definition that the speaker is conveying. Study S5 demonstrated a wide variety of languages being tested with one of the results being the lowest Speech to Text Recognition accuracy rate was for English, 93.94%, due to the strong accent of the participant. S31 demonstrated that Spanish performed better than English which is attributed to the participants main language being Spanish.

These examples show that even though there is high performance for the languages being tested, it is lower than average due to the accent of the user. Thus, more work needs to be done to help lessen the performance gap between native and non-native accents.

CHAPTER III:
METHODOLOGY

3.1) Test Goals

The goals for the tests conducted is to investigate if Apple's Speech to Text is ready to be used for coding and develop an app to format text to how an IDE would expect it to be. The reasoning to investigate if Apple's Speech To text is ready to be used for coding is to ensure the STT system would capture what is being stated by the user. If the STT system were to not capture what is being stated by the user in a consistent and reliable manner, then the amount of work necessary to make corrections would outweigh the amount of work saved by using the STT system.

3.2) Research Questions

To guide the tests to be conducted the following research questions were created. These were written to be specific question to help produce answers that would aid in further research and have no room for mis interpretation.

RQ1) How well does Apple's Speech to Text transcribe speech?

RQ2) Is Apple's STT ready to be integrated with coding?

RQ3) How do programmers read code aloud versus how it is written?

RQ4) How well does the formatting application perform based on transcription input?

The first question was created to measure Apple's STT performance. This question has a data-based answer. It will be tested by comparing what was spoken to the STT system to what was transcribed by it. This question is necessary to provide information for further research questions. It will also help to show what is the current state of it now for future comparison in the research field.

The second question was created to give an analysis of the answer from RQ1. This question has an opinion-based answer, but instead of giving a definitive answer I will only present the data to allow the reader to form their own answer. Many variables can affect how a person perceives if STT is ready to be integrated with code and so I will present the data to back up a reader’s decision.

The third question was created to understand how programmers speak code aloud. In what ways do they feel that pieces of code should be read aloud. This question has a data-based answer. This question is necessary to understand how to create a formatting program that will be successful by considering the ways that different programmers speak pieces of code.

The fourth question was created to give an analysis of how well the application created for the study performed. This question has a data-based answer. The application was created to be a starting point for formatting text outputted by a STT system into the format expected by a typical IDE. It was created based on how the creator spoke code aloud and thus needs to be analyzed based on the results of RQ3 to understand where its strengths and weaknesses lie to create a better version that would suit the programming community better.

3.3) Formatting Application

The following application was used in the research testing. It was developed to collect the text outputted by Apple’s keyboard’s speech to text function and then format the text into what an integrated development environment (IDE) would expect when programming.

3.3.1) Application Layout

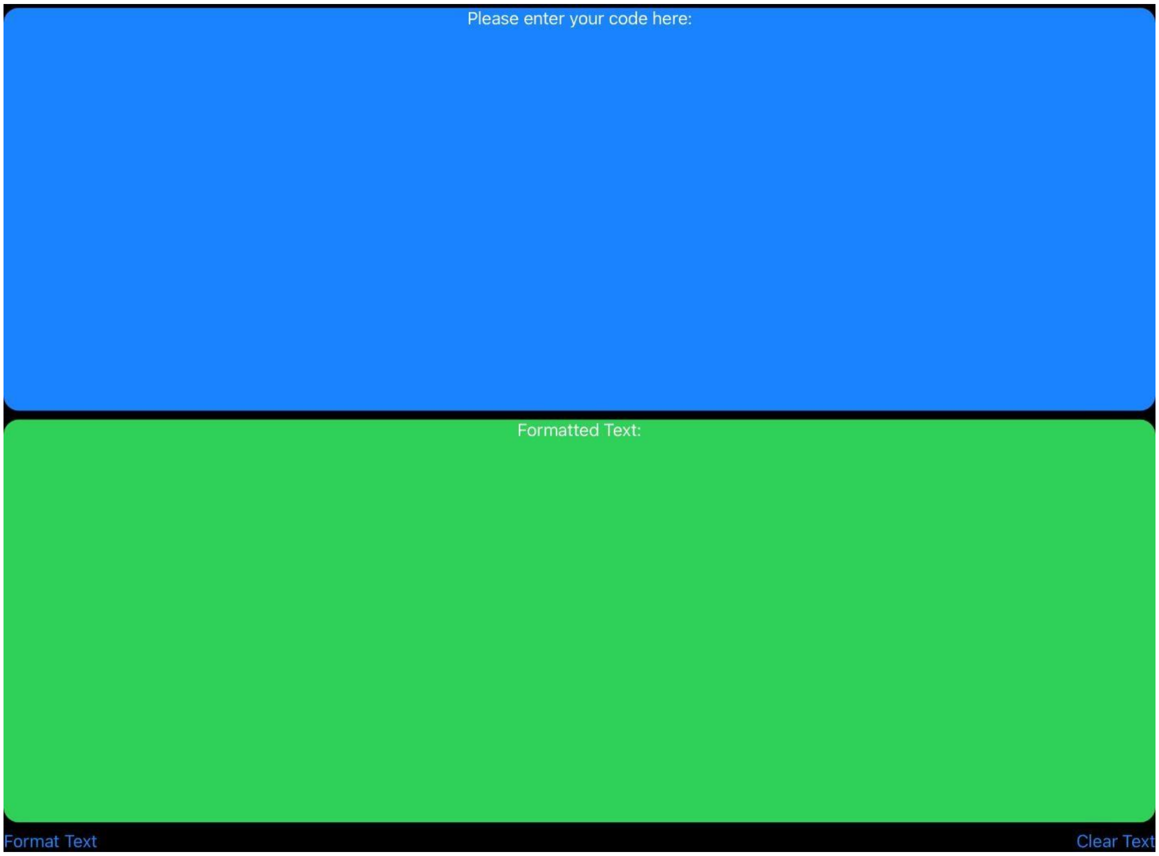


Figure 6.
Application Screenshot

The application has three main sections: text entry section, formatted text section, and button section.

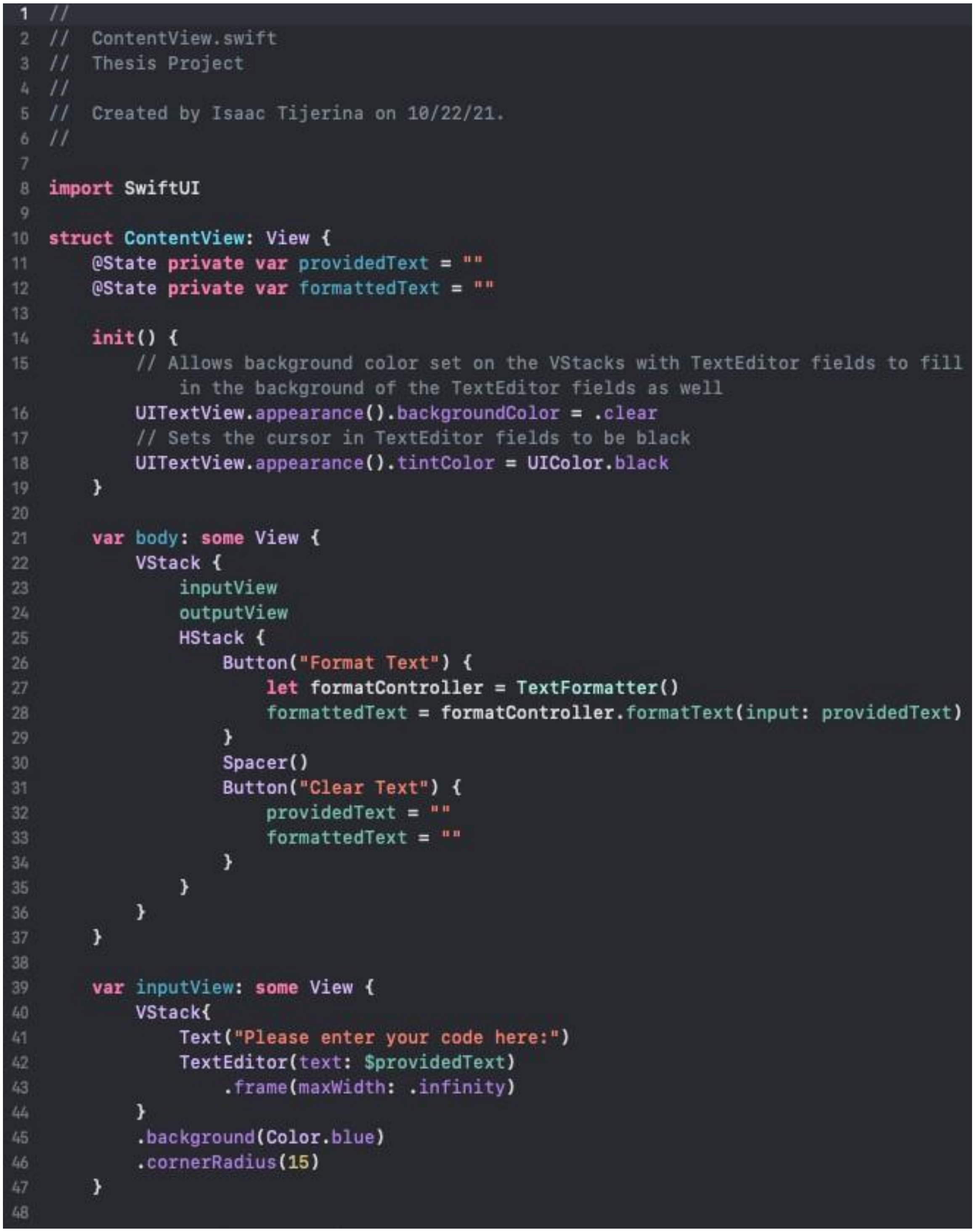
The text entry section is on the top of the application. It is denoted by its blue background and a title of “Please enter your code here”. The user can interact with it by entering text and editing text through the keyboard.

The formatted text section is below the text entry section. It is denoted by its green background and a title of “Formatted Text”. The user is not able to interact with it. It is populated with text when the “Format Text” button was selected.

The button section is on the bottom of the application. It contains a “Format Text” button and “Clear Text” button. When the “Format Text” button is selected the text in the text entry section is formatted into what an integrated development environment (IDE) would expect when programming. It is then displayed in the formatted text section. When the “Clear Text” section is selected all text in both the text entry and formatted text section are removed.

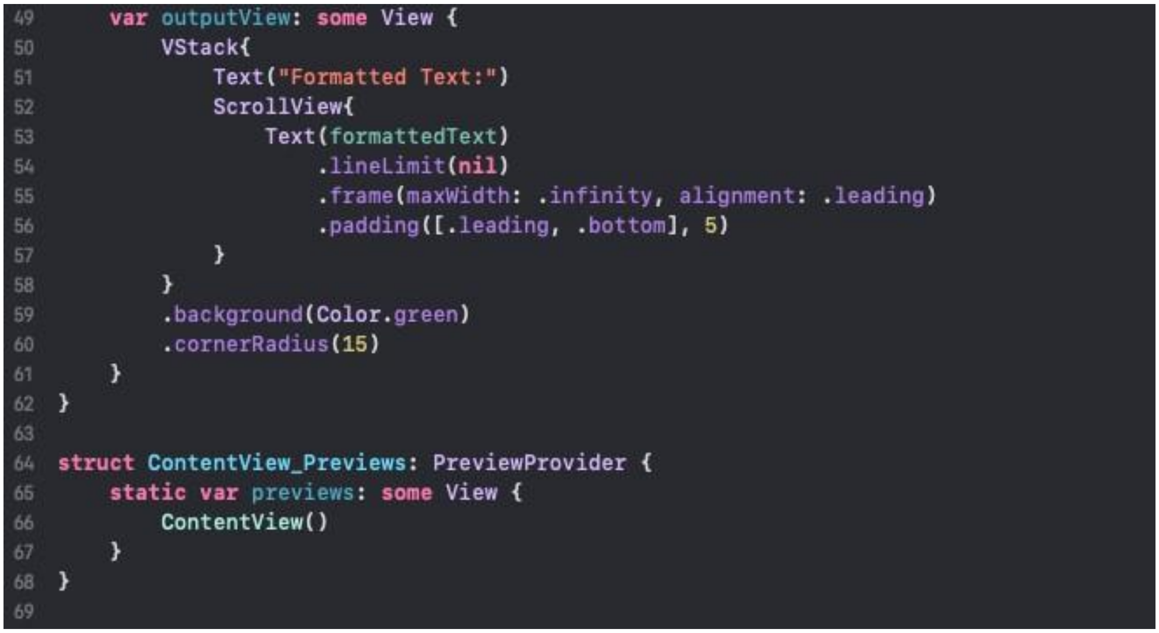
3.3.2) Application Code

3.3.2.1) ContentView.swift



```swift
//
//  ContentView.swift
//  Thesis Project
//
//  Created by Isaac Tijerina on 10/22/21.
//

import SwiftUI

struct ContentView: View {
    @State private var providedText = ""
    @State private var formattedText = ""

    init() {
        // Allows background color set on the VStacks with TextEditor fields to fill in the background of the TextEditor fields as well
        UITextView.appearance().backgroundColor = .clear
        // Sets the cursor in TextEditor fields to be black
        UITextView.appearance().tintColor = UIColor.black
    }

    var body: some View {
        VStack {
            inputView
            outputView
            HStack {
                Button("Format Text") {
                    let formatController = TextFormatter()
                    formattedText = formatController.formatText(input: providedText)
                }
                Spacer()
                Button("Clear Text") {
                    providedText = ""
                    formattedText = ""
                }
            }
        }
    }

    var inputView: some View {
        VStack{
            Text("Please enter your code here:")
            TextEditor(text: $providedText)
                .frame(maxWidth: .infinity)
        }
        .background(Color.blue)
        .cornerRadius(15)
    }

```

Figure 7.
ContentView.swift Screenshot 1

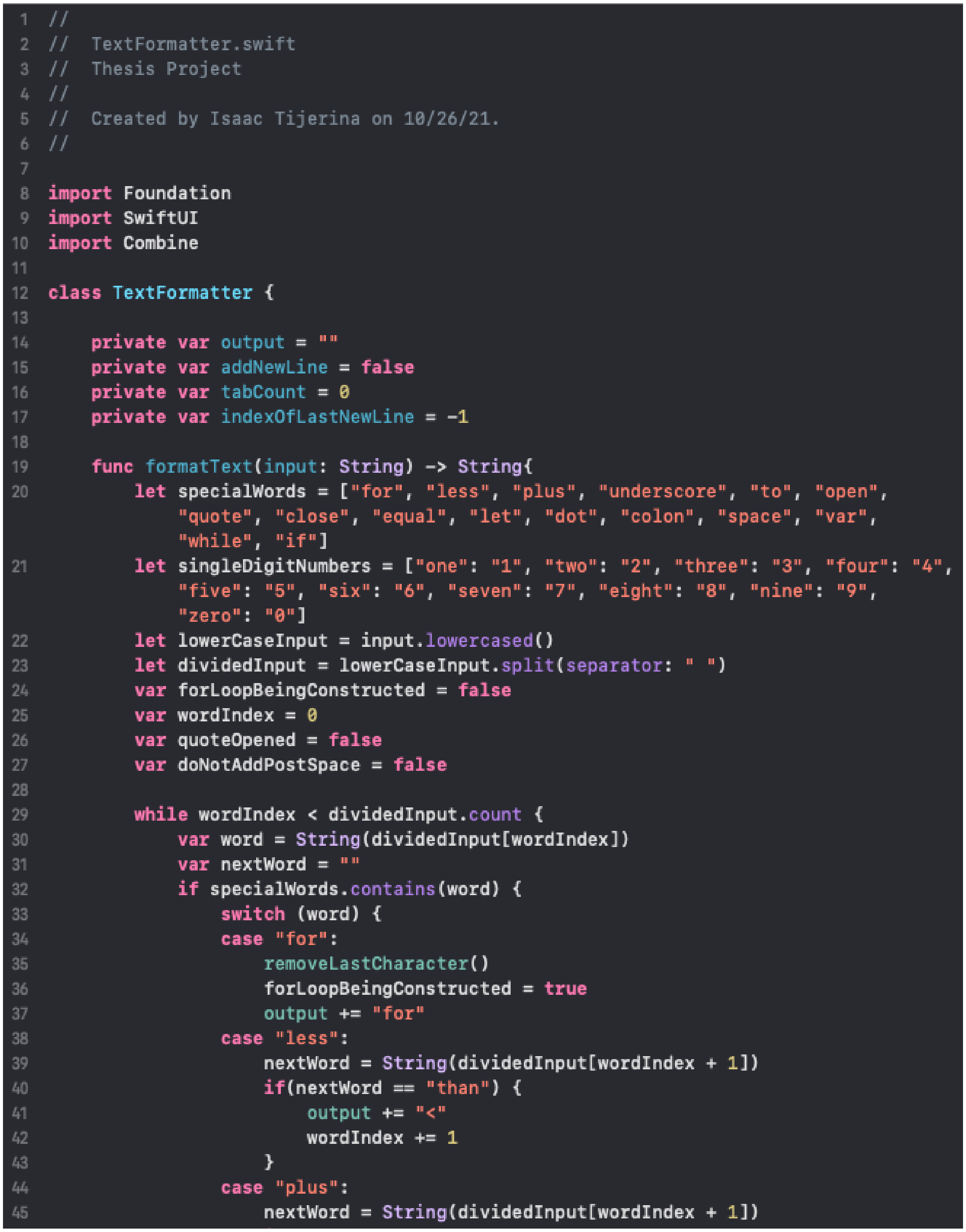


```
    var outputView: some View {
        VStack{
            Text("Formatted Text:")
            ScrollView{
                Text(formattedText)
                    .lineLimit(nil)
                    .frame(maxWidth: .infinity, alignment: .leading)
                    .padding([.leading, .bottom], 5)
            }
        }
        .background(Color.green)
        .cornerRadius(15)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Figure 8.
ContentView.swift Screenshot 2

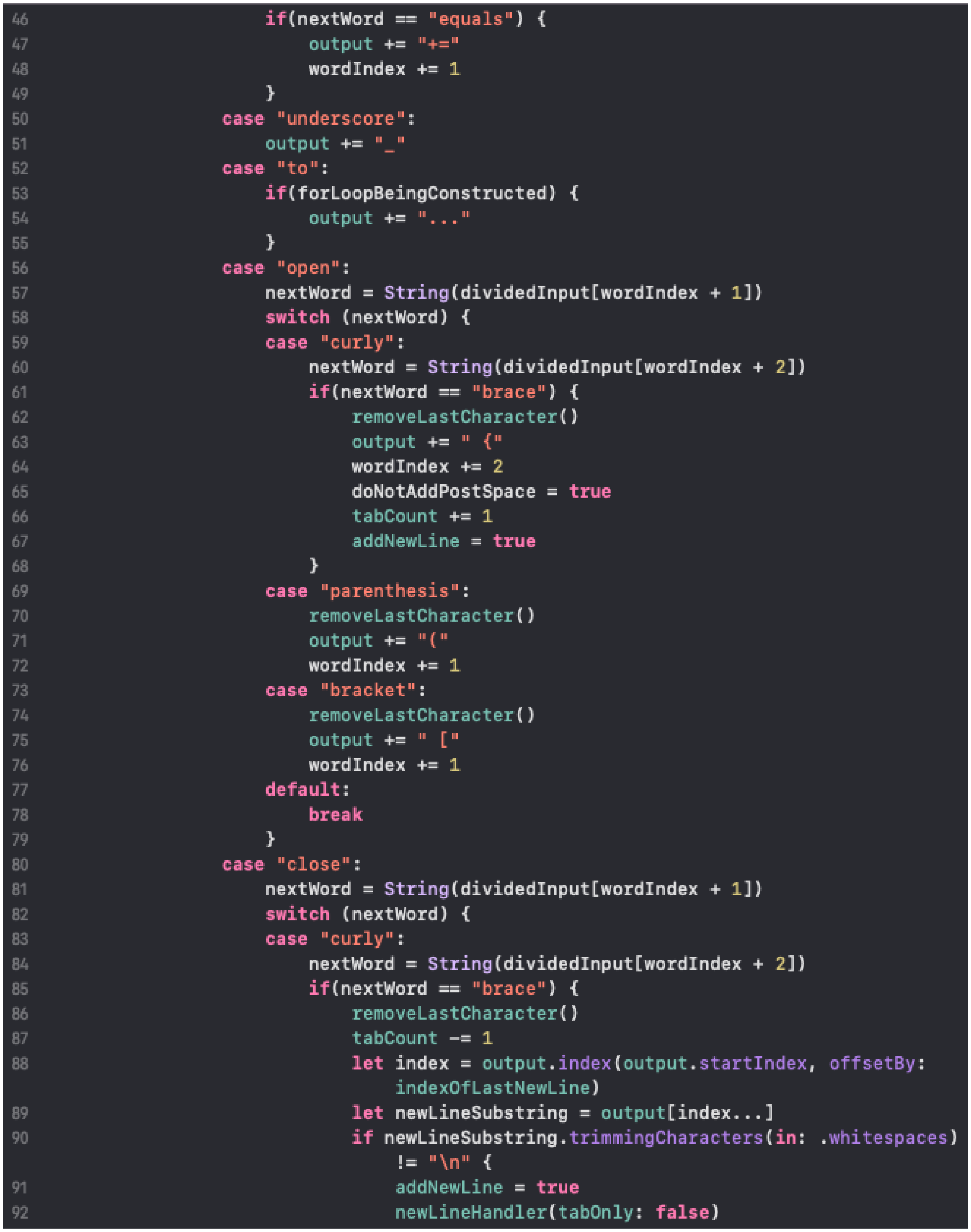
ContentView.swift was created to handle the UI elements of the application.

3.3.2.2) TextFormatter.swift



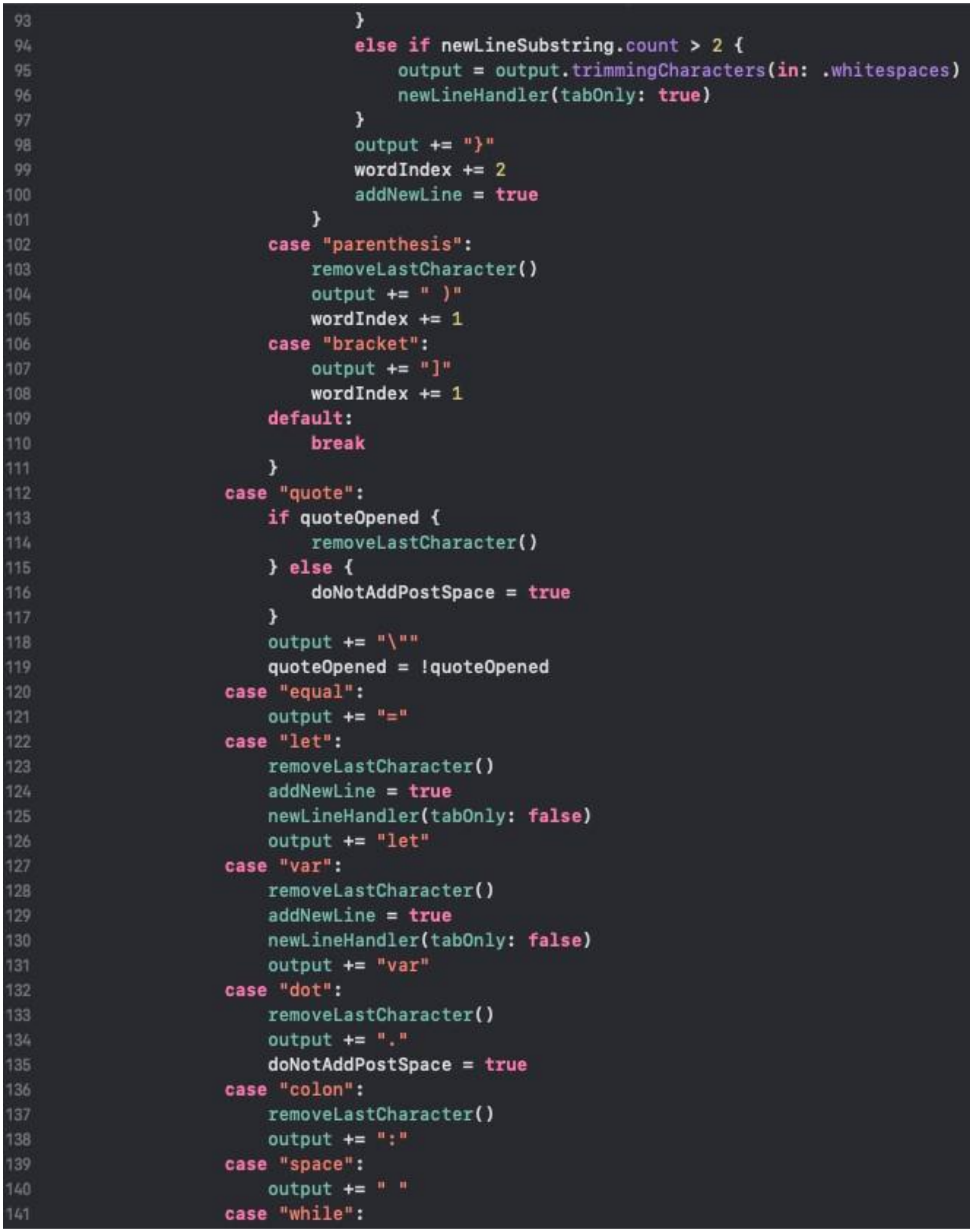
```
//
//  TextFormatter.swift
//  Thesis Project
//
//  Created by Isaac Tijerina on 10/26/21.
//

import Foundation
import SwiftUI
import Combine

class TextFormatter {

    private var output = ""
    private var addNewLine = false
    private var tabCount = 0
    private var indexOfLastNewLine = -1

    func formatText(input: String) -> String{
        let specialWords = ["for", "less", "plus", "underscore", "to", "open",
            "quote", "close", "equal", "let", "dot", "colon", "space", "var",
            "while", "if"]
        let singleDigitNumbers = ["one": "1", "two": "2", "three": "3", "four": "4",
            "five": "5", "six": "6", "seven": "7", "eight": "8", "nine": "9",
            "zero": "0"]
        let lowerCaseInput = input.lowercased()
        let dividedInput = lowerCaseInput.split(separator: " ")
        var forLoopBeingConstructed = false
        var wordIndex = 0
        var quoteOpened = false
        var doNotAddPostSpace = false

        while wordIndex < dividedInput.count {
            var word = String(dividedInput[wordIndex])
            var nextWord = ""
            if specialWords.contains(word) {
                switch (word) {
                case "for":
                    removeLastCharacter()
                    forLoopBeingConstructed = true
                    output += "for"
                case "less":
                    nextWord = String(dividedInput[wordIndex + 1])
                    if(nextWord == "than") {
                        output += "<"
                        wordIndex += 1
                    }
                case "plus":
                    nextWord = String(dividedInput[wordIndex + 1])
```

Figure 9.
TextFormatter.swift Screenshot 1



```
    if(nextWord == "equals") {
        output += "+="
        wordIndex += 1
    }
case "underscore":
    output += "_"
case "to":
    if(forLoopBeingConstructed) {
        output += "..."
    }
case "open":
    nextWord = String(dividedInput[wordIndex + 1])
    switch (nextWord) {
    case "curly":
        nextWord = String(dividedInput[wordIndex + 2])
        if(nextWord == "brace") {
            removeLastCharacter()
            output += " {"
            wordIndex += 2
            doNotAddPostSpace = true
            tabCount += 1
            addNewLine = true
        }
    case "parenthesis":
        removeLastCharacter()
        output += "("
        wordIndex += 1
    case "bracket":
        removeLastCharacter()
        output += " ["
        wordIndex += 1
    default:
        break
    }
case "close":
    nextWord = String(dividedInput[wordIndex + 1])
    switch (nextWord) {
    case "curly":
        nextWord = String(dividedInput[wordIndex + 2])
        if(nextWord == "brace") {
            removeLastCharacter()
            tabCount -= 1
            let index = output.index(output.startIndex, offsetBy:
                indexOfLastNewLine)
            let newLineSubstring = output[index...]
            if newLineSubstring.trimmingCharacters(in: .whitespaces)
                != "\n" {
                addNewLine = true
                newLineHandler(tabOnly: false)
```

Figure 10.
TextFormatter.swift Screenshot 2



```
                                }
                                else if newLineSubstring.count > 2 {
                                    output = output.trimmingCharacters(in: .whitespaces)
                                    newLineHandler(tabOnly: true)
                                }
                                output += "}"
                                wordIndex += 2
                                addNewLine = true
                            }
                        case "parenthesis":
                            removeLastCharacter()
                            output += " )"
                            wordIndex += 1
                        case "bracket":
                            output += "]"
                            wordIndex += 1
                        default:
                            break
                        }
                    case "quote":
                        if quoteOpened {
                            removeLastCharacter()
                        } else {
                            doNotAddPostSpace = true
                        }
                        output += "\""
                        quoteOpened = !quoteOpened
                    case "equal":
                        output += "="
                    case "let":
                        removeLastCharacter()
                        addNewLine = true
                        newLineHandler(tabOnly: false)
                        output += "let"
                    case "var":
                        removeLastCharacter()
                        addNewLine = true
                        newLineHandler(tabOnly: false)
                        output += "var"
                    case "dot":
                        removeLastCharacter()
                        output += "."
                        doNotAddPostSpace = true
                    case "colon":
                        removeLastCharacter()
                        output += ":"
                    case "space":
                        output += " "
                    case "while":
```

Figure 11.
TextFormatter.swift Screenshot 3

TextFormatter.swift was created to handle the text formatting. It takes in the raw text as a String and outputs a String with the formatted text. The main processing of the input is done using switches and cases which was done to have a clear way to process words and phrases.

3.4) Test Procedure

The testing procedure was outlined in a document to create a standard ensuring the same testing criteria was followed for each participant. It was broken into two sections (Equipment and Steps) both of which will be expanded upon in the following sections

3.3.1) Equipment

The pieces of equipment used in the experiment were an iPad Pro and a Mac [version type]. The iPad was used to run the formatting application being tested and to use the Speech to Text function built into the keyboard. The Mac was used to build and run the formatting app in XCode, record the audio of the participant speaking, and to save screenshots of the outputs of both the STT function and the formatting app.

3.3.2) Steps

1. Run `Thesis Project` app on the iPad.

This is necessary to use the formatting app which is titled ‘Thesis Project’.

2. Select the `Clear Data` button.

This ensures the input area off the app is clear in the case anything was accidentally typed into it

3. Begin audio recording using QuickTime

This is to begin collecting audio recordings of what is stated by the participant. It is expected that after this step only what is necessary for the study will be said aloud.

4. Select the Apple keyboard dictation button on the iPad and have the participant speak the following piece of code in their own words:

```
for _ 0…1 {

   print(“hello world”)

   for x 0…1 {

      var test = 10

   }

}


switch (food) {
case “taco”:
good = true
case “rice”:
good = true
case “ketchup”
good = false
default:
break

}



while participating > skipping {

   if job == good {

      var response = “thank you very much”

   }

   compensation += 10

}
```

This step will begin the testing of Apple’s Speech to Text functionality (RQ1) as well as collect data to understand how programmers speak code aloud (RQ3). The piece of code was written to have basic code that most programmers will understand as well as being code that the formatting will understand.

5. End audio recording.

The audio recording is ended and will be labeled at this point. It was decided to not have one long audio recording of both parts of the study to for one keep each audio recording easy to review in the analysis portion of the study and for two to limit the data collected to only what is necessary for the study for participant information protection.

6. Select the `Format Text` button.

The `Format Text` button is selected to use the app's functionality to format the transcribed text.

7. Screenshot the app.

A screenshot is taken and labeled for later analysis of what was transcribed and how it was formatted.

8. Select the `Clear Text` button.

The input and formatted text areas are cleared to begin the next test.

9. Begin another audio recording.

This is to begin collecting an audio recording of what the participant states to the STT system for the second test.

10. Select the Apple keyboard dictation button on the iPad and have the participant speak the following paragraph in their own words:

let variable equal 0 let boolean equal false let lower case input equal input dot lowercased open parenthesis close parenthesis let divided input equal lower-case input dot split open parenthesis separator colon quote space quote close parenthesis var word. Index equal zero while word index less than divided input dot count open curly brace var word equal string open parenthesis divided input open bracket word index close bracket close parenthesis var next word equal quote space quote if special words dot contains open parenthesis word close parenthesis open curly brace switch open parenthesis word close parenthesis open curly brace case quote woah quote colon remove last character open parenthesis close parenthesis close curly brace close curly brace close curly brace

This is to test RQ1 and RQ4. It is a paragraph of code that has been written out how the authors think a programmer may read code aloud. By telling the participant exactly what to say it will solely focus on testing the STT functionality and formatting functionality versus having to take into consideration how the participant wants to state code like in the first test.

11. End audio recording.

The audio recording is ended and labeled.

12. Select the `Format Text` button.

The `Format Text` button is selected to use the app's functionality to format the transcribed text.

13. Screenshot the app.

A screenshot is taken and labeled for later analysis of what was transcribed and how it was formatted.

CHAPTER IV:

RESULTS

4.1) Data Collection

Data collection took place over two days. It consisted of a four hour block each day in which a participant could sign up for a ten-minute segment. Each participant received a ten-dollar Amazon gift card and a bag of snacks worth up to five dollars at the end of their participation. Both in the first and second day of the collection had nineteen participants for a total of thirty-eight participants.

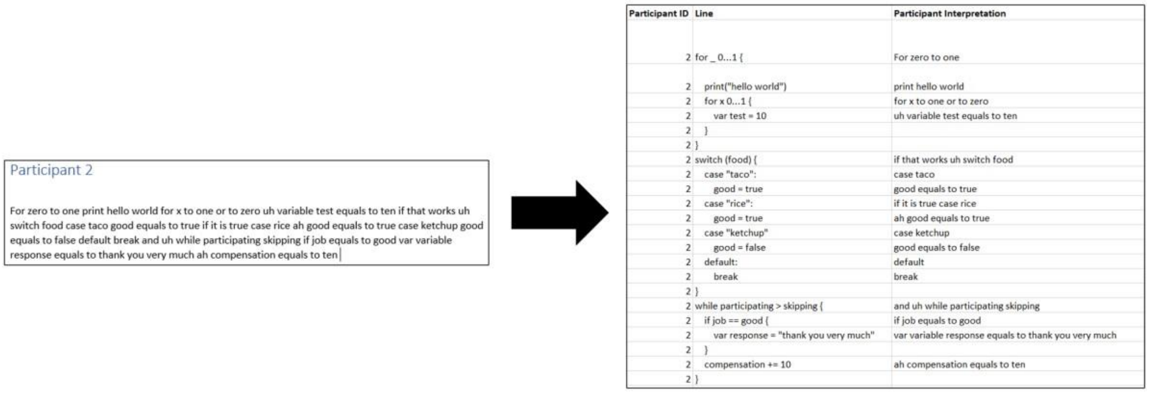
The collection took place in a study room that had minimal sound proofing and so conversations from the room next door could be heard. The participants sat approximately two feet away from the testing equipment and sat across the table from the tester. The testing equipment was an iPad for collection speech to text into the formatting app and a mac mini with a pair of earbuds connected to it to record what the participants stated for later manual transcription.

A notable occurrence discovered during testing was that after one minute of using the Apple keyboard speech to text it would automatically turn itself off. This effected results due to this issue required the tester to restart the speech to text functionality. It was decided that when the participant approached a time of one minute while reading, the tester would not interrupt them to then restart the STT functionality, but rather let the participant continue reading. This would result in the loss of audio transcribed by the STT system, but it is seen as a fault in the system. If a user would be speaking code in a real-world scenario and surpassed a minute of continuous talking while the STT system automatically turned itself off this would be a fault in the system that would pose a burden on the user. Hence the data collection was done to mimic this.

Another note was that of the thirty-eight participants, data for two participants were removed from the results. This was done because once the audio for these participants were manually transcribed it was determined that Participant 17 did not have a clear understanding of programming and Participant 4 did not understand what was being asked of in the collection and so explained the code instead of reading it.

4.2) Results For How Participants Read Code

The first step to analyze how the participants read the code is to first manually transcribe the audio recordings collected of what the participants said aloud. These thirtysix audio recordings were manually transcribed by having the researcher listen to the recordings and type what was stated as it was being said. The researcher was allowed to listen to the recordings multiple times to ensure the transcription was correct. These manual transcriptions can be found in Appendix D. Once the transcriptions were written the next step was to separate each paragraph into separate lines where each line corresponded to a line of code that the participant read. An example of this can be found in Figure 12.



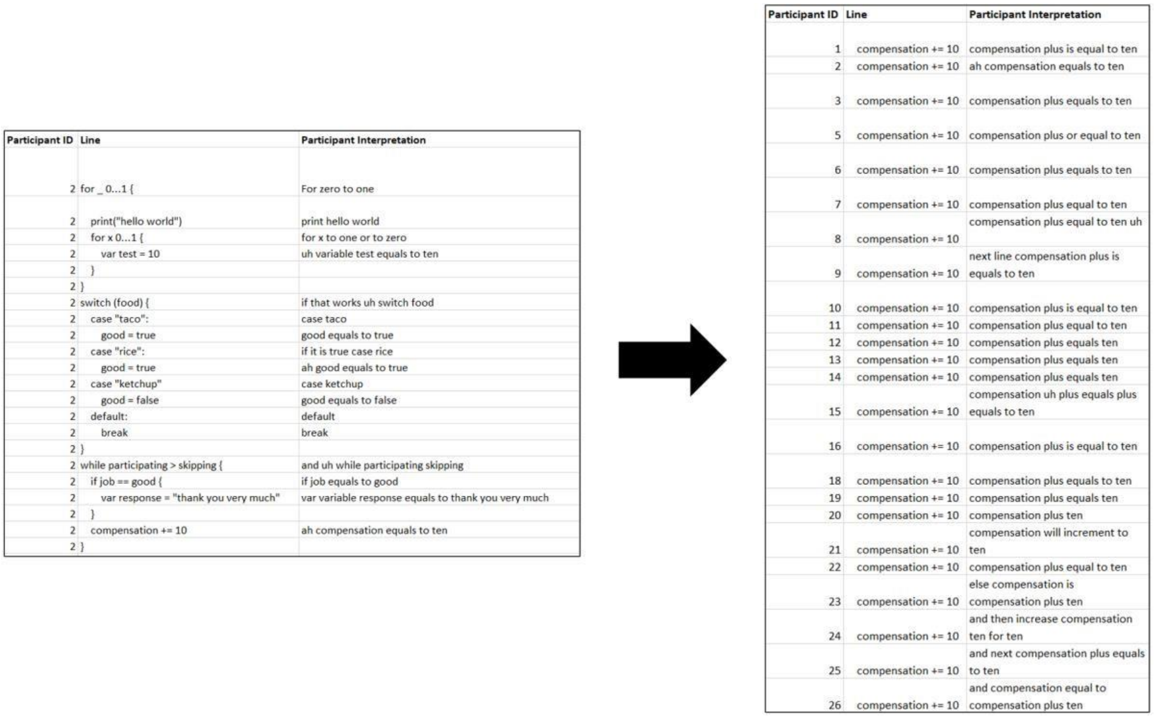
Participant 2

For zero to one print hello world for x to one or to zero uh variable test equals to ten if that works uh switch food case taco good equals to true if it is true case rice ah good equals to true case ketchup good equals to false default break and uh while participating skipping if job equals to good var variable response equals to thank you very much ah compensation equals to ten

Participant ID	Line	Participant Interpretation
2	for _ 0...1 {	For zero to one
2	print("hello world")	print hello world
2	for x 0...1 {	for x to one or to zero
2	var test = 10	uh variable test equals to ten
2	}	
2	}	
2	switch (food) {	if that works uh switch food
2	case "taco":	case taco
2	good = true	good equals to true
2	case "rice":	if it is true case rice
2	good = true	ah good equals to true
2	case "ketchup"	case ketchup
2	good = false	good equals to false
2	default:	default
2	break	break
2	}	
2	while participating > skipping {	and uh while participating skipping
2	if job == good {	if job equals to good
2	var response = "thank you very much"	var variable response equals to thank you very much
2	}	
2	compensation += 10	ah compensation equals to ten
2	}	

Figure 12.
Example of separating transcribed text into corresponding lines of code

Once this was complete all the lines of transcribed speech were then grouped based on which words or characters were meant to be stated during the reading of the line. An example of this can be seen in figure 13 where all the lines of transcribed text that were supposed to contain a transcription for what the participant stated for ‘+=’ were written.



Participant ID	Line	Participant Interpretation
2	for _ 0...1 {	For zero to one
2	print("hello world")	print hello world
2	for x 0...1 {	for x to one or to zero
2	var test = 10	uh variable test equals to ten
2	}	
2	}	
2	switch (food) {	if that works uh switch food
2	case "taco":	case taco
2	good = true	good equals to true
2	case "rice":	if it is true case rice
2	good = true	ah good equals to true
2	case "ketchup"	case ketchup
2	good = false	good equals to false
2	default:	default
2	break	break
2	}	
2	while participating > skipping {	and uh while participating skipping
2	if job == good {	if job equals to good
2	var response = "thank you very much"	var variable response equals to thank you very much
2	}	
2	compensation += 10	ah compensation equals to ten
2	}	

Participant ID	Line	Participant Interpretation
1	compensation += 10	compensation plus is equal to ten
2	compensation += 10	ah compensation equals to ten
3	compensation += 10	compensation plus equals to ten
5	compensation += 10	compensation plus or equal to ten
6	compensation += 10	compensation plus equals to ten
7	compensation += 10	compensation plus equal to ten
8	compensation += 10	compensation plus equal to ten uh
9	compensation += 10	next line compensation plus is equals to ten
10	compensation += 10	compensation plus is equal to ten
11	compensation += 10	compensation plus equal to ten
12	compensation += 10	compensation plus equals ten
13	compensation += 10	compensation plus equals ten
14	compensation += 10	compensation plus equals ten
15	compensation += 10	compensation uh plus equals plus equals to ten
16	compensation += 10	compensation plus is equal to ten
18	compensation += 10	compensation plus equals to ten
19	compensation += 10	compensation plus equals ten
20	compensation += 10	compensation plus ten
21	compensation += 10	compensation will increment to ten
22	compensation += 10	compensation plus equal to ten
23	compensation += 10	else compensation is compensation plus ten
24	compensation += 10	and then increase compensation ten for ten
25	compensation += 10	and next compensation plus equals to ten
26	compensation += 10	and compensation equal to compensation plus ten

Figure 13.
Example of gathering all the lines of transcribed text that correspond to the same word or character

Then each word or character from the original code was analyzed to see how each participant stated it in comparison with all the others. The following chart presents the top ways that participants stated the words/characters:

Table 3.
Top ways that participants stated the words/characters

Word/Character	**Top Participant Interpretation**	**Percentage of Occurrences**
For	“for”	84.72%
_	“underscore”	50%
X	“x”	72.22%
0	“zero”	88.89%
…	“to”	73.61%

Word/Character	**Top Participant Interpretation**	**Percentage of Occurrences**
1	“one”	98.61%
{	Did not state anything	45.56%
(	Did not state anything	65.28%
First “	Did not state	66.67%

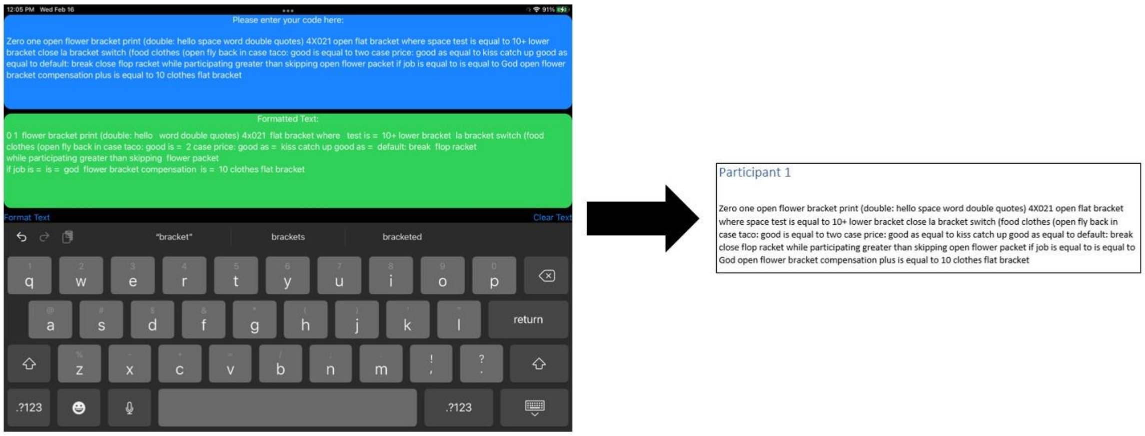
	anything	
Second “	Did not state anything	77.22%
)	Did not state anything	66.67%
var	“var”	59.7%
=	“equals to”	26.67%
10	“ten”	94.44%
}	Did not state anything	63%
switch	“switch”	75%
case	“case”	77.78%
:	Did not state anything	60.19%
default	“default”	83.33%
break	“break”	86.11%

while	“while”	91.67%
Word/Character	**Top Participant Interpretation**	**Percentage of Occurrences**
>	“greater than”	55.56%
if	“if”	97.22%
==	“equals to equals to” and “equal to equal to” and “equals equals”	11.11% Each
+=	“plus equal to” and “plus equals”	13.89 Each

From this data what can be seen is the characters that are for encapsulating sections were not stated aloud in any manner for most occurrences. The characters and words that defined statements were stated each time in most occurrences. The word/character with the most variation in how it was interpreted was } with 43 different interpretations. The word/character with the least number of variations of how it was interpreted was if and 1 both with two different interpretations each. It should be noted that all words/characters were stated in at least two occurrences, and all had at least two different interpretations.

4.3) Results For Apple's Speech To Text Transcriptions

The first step to analyze Apple's Speech to Text Transcriptions of what the participants stated was to transfer the text from the screenshots of the formatting application to a word document for easier manipulation and analysis. This can be seen below:



Please enter your code here:

Zero one open flower bracket print (double: hello space word double quotes) 4X021 open flat bracket where space test is equal to 10+ lower bracket close la bracket switch (food clothes (open fly back in case taco: good is equal to two case price: good as equal to kiss catch up good as equal to default: break close flop racket while participating greater than skipping open flower packet if job is equal to is equal to God open flower bracket compensation plus is equal to 10 clothes flat bracket

Formatted Text:

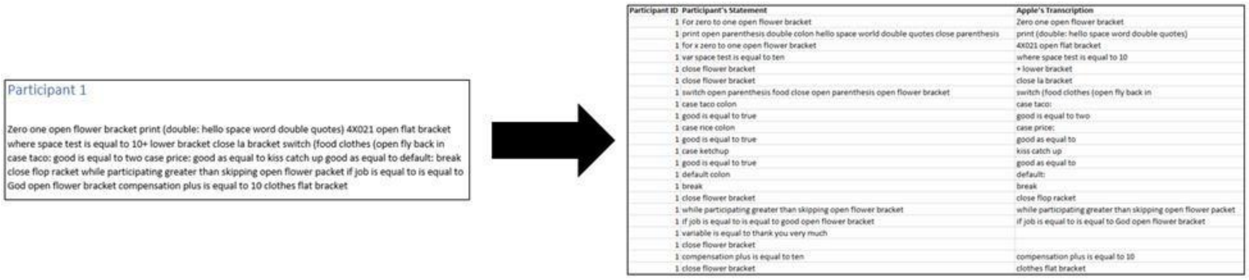
0 1 flower bracket print (double: hello word double quotes) 4x021 flat bracket where test is = 10+ lower bracket la bracket switch (food clothes (open fly back in case taco: good is = 2 case price: good as = kiss catch up good as = default: break flop racket
while participating greater than skipping flower packet
if job is = is = god flower bracket compensation is = 10 clothes flat bracket

Participant 1

Zero one open flower bracket print (double: hello space word double quotes) 4X021 open flat bracket where space test is equal to 10+ lower bracket close la bracket switch (food clothes (open fly back in case taco: good is equal to two case price: good as equal to kiss catch up good as equal to default: break close flop racket while participating greater than skipping open flower packet if job is equal to is equal to God open flower bracket compensation plus is equal to 10 clothes flat bracket

Figure 14.
Example of transferring Apple's STT transcription to a word document

The next step is to separate each paragraph of transcribed text into the spoken lines that they correspond to from section 4.2. An example of this is below:



Participant 1

Zero one open flower bracket print (double: hello space word double quotes) 4X021 open flat bracket where space test is equal to 10+ lower bracket close la bracket switch (food clothes (open fly back in case taco: good is equal to two case price: good as equal to kiss catch up good as equal to default: break close flop racket while participating greater than skipping open flower packet if job is equal to is equal to God open flower bracket compensation plus is equal to 10 clothes flat bracket

Participant ID	Participant's Statement	Apple's Transcription
1	For zero to one open flower bracket	Zero one open flower bracket
1	print open parenthesis double colon hello space world double quotes close parenthesis	print (double: hello space word double quotes)
1	for x zero to one open flower bracket	4X021 open flat bracket
1	var space test is equal to ten	where space test is equal to 10
1	close flower bracket	+ lower bracket
1	close flower bracket	close la bracket
1	switch open parenthesis food close open parenthesis open flower bracket	switch (food clothes (open fly back in
1	case taco colon	case taco:
1	good is equal to true	good is equal to two
1	case rice colon	case price:
1	good is equal to true	good as equal to
1	case ketchup	kiss catch up
1	good is equal to true	good as equal to
1	default colon	default:
1	break	break
1	close flower bracket	close flop racket
1	while participating greater than skipping open flower bracket	while participating greater than skipping open flower packet
1	if job is equal to is equal to good open flower bracket	if job is equal to is equal to God open flower bracket
1	variable is equal to thank you very much	
1	close flower bracket	
1	compensation plus is equal to ten	compensation plus is equal to 10
1	close flower bracket	clothes flat bracket

Figure 15.
Example of separating transcribed text into their corresponding spoken lines

The transcriptions were then analyzed. All results can be seen in Appendix G. Each transcription was analyzed to note the number of missing words, number of correctly transcribed words, number of correctly interpreted words (interpreted in this case meaning the word was understood to be a symbol or number and was transcribed as such), number of incorrectly interpreted words, and the number of incorrectly transcribed words. The measurements of number of correct words and number of interpreted words are measurements of the desired outcomes. The measurements of number of missing words, number of incorrectly interpreted words, and number of incorrectly transcribed words are measurements of not desired outcomes and are considered mistakes. Below is a summary of the average, highest, and lowest percentages found in a participant's transcription for each category.

Table 4.
Summary of the average, highest, and lowest percentages found in a participant's transcription for each category

	# Of Missing Words	**# Of Correct Words**	**# Of Interpreted Words**	**# Of Incorrectly Interpreted**	**# Of Incorrectly Transcribed**
Average	17.7999175%	50.10262324%	13.1196715%	2.580385295%	16.39740247%
Lowest	0%	16.21621622%	0.877192982%	0%	4.347826087%
Highest	59.45945946%	92.06349206%	31.19266055%	8.771929825%	34.48275862%

From this we can see that on average Apple transcribed the spoken words correctly half the time, interpreted words correctly 13.12% of the time, and the rest of the time transcribed errors. In addition, below is a table of notable interpretations that Apple transcribed:

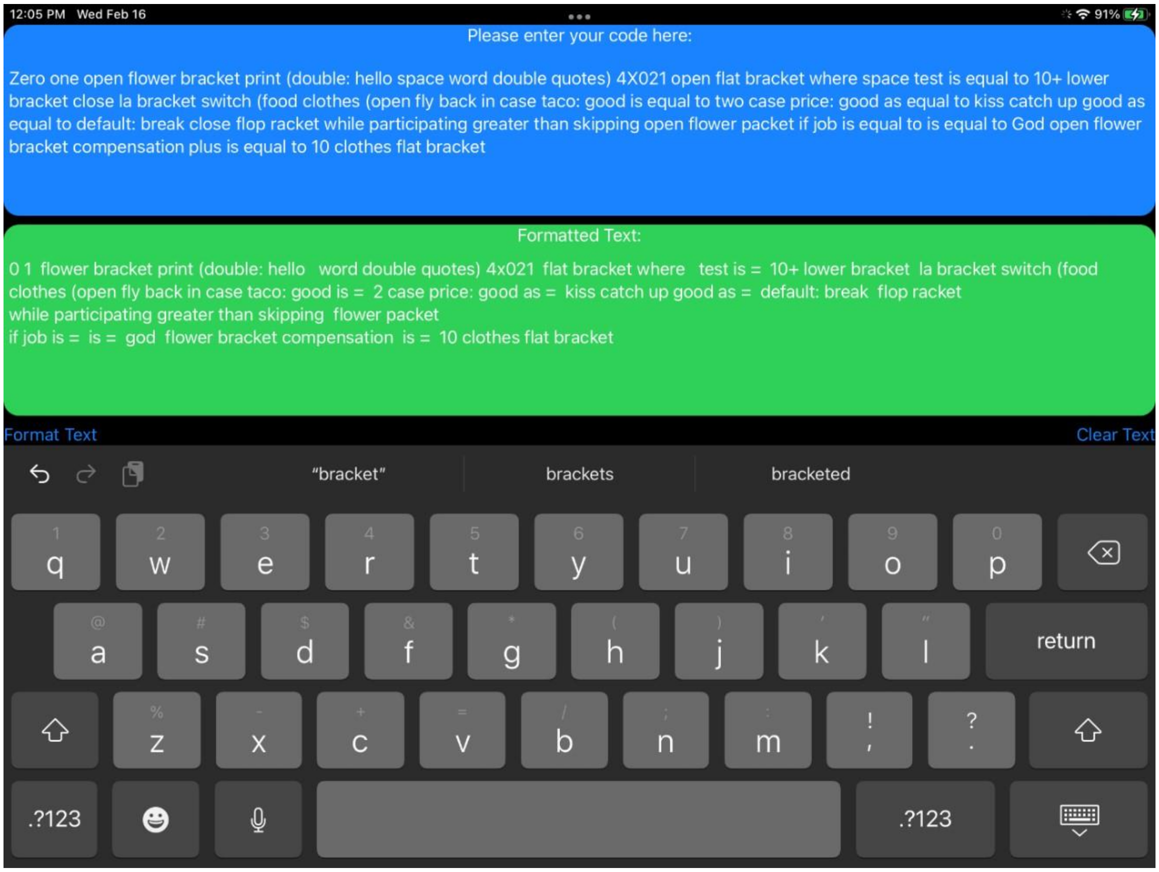
Table 5.
Notable interpretations that Apple transcribed

List of Notable Interpretations	
open parenthesis	(
colon	:
close parenthesis	)
zero	0
one	1
ten	10
ellipses	…
open bracket	[
quotations	"
close bracket	]
open quotation	"
quote	"
close quotation	""
underscore	_
dot	.
semi colon	;
new line	\n

three	3
periods	.
for	4
to	2
open curly brace	{
close curly brace	}
open brace	(
open brace	{
List of Notable Interpretations	
assign	+

4.4) Results For Formatting Application

The screenshots of the output of the formatting application to be analyzed can be found in Appendix H. The application performed at a 0% success rate in formatting regular text to code. An example of this can be seen here where the blue section is the transcribed text, and the green section is the text formatted for coding. As seen the area in green is not executable code. It does have remnants that represent code, but most of it is not.



Please enter your code here:

Zero one open flower bracket print (double: hello space word double quotes) 4X021 open flat bracket where space test is equal to 10+ lower bracket close la bracket switch (food clothes (open fly back in case taco: good is equal to two case price: good as equal to kiss catch up good as equal to default: break close flop racket while participating greater than skipping open flower packet if job is equal to is equal to God open flower bracket compensation plus is equal to 10 clothes flat bracket

Formatted Text:

0 1 flower bracket print (double: hello   word double quotes) 4x021 flat bracket where   test is = 10+ lower bracket la bracket switch (food clothes (open fly back in case taco: good is = 2 case price: good as = kiss catch up good as = default: break flop racket
while participating greater than skipping flower packet
if job is = is = god flower bracket compensation is = 10 clothes flat bracket

Figure 16.
Participant 1's transcribed and formatted text

While characters such as numbers and ellipses were properly translated from words to characters, this ended up being a minority in the larger error rate. While this was due to inputted text not being correct to be formatted, it is not an area to blame as a reason the app failed. Rather it is the improvement that can be built upon. Based on the data presented in sections 4.2 and 4.3, the app was simply not built to handle such manners that the public reads aloud code and how it is transcribed by Apple. Regardless of if you the reader feels that the results of 4.3 determine Apple's STT to be ready for use in coding or not, the application should still be built to handle all cases including the errors in speech and transcription to reach the goal of a 100% success rate.

CHAPTER V:
CONCLUSION AND FUTURE WORK

To determine our conclusions let us revisit the research questions. To answer RQ1, it transcribes text at an average success rate of 50.1% and in correctly transcribing and interpreting words at an average success rate of 13.12%. To answer RQ2, this is up to the reader. The argument can be made that any success rate can be considered ready for transcription; it is just up to the user for how many reviews they would like to make which is the real question. The success rate is right in the middle at 50% so on average half the text would need revisions. This is a far distance from a 100% success rate, but it is progress and can be made better. To answer RQ3, there is a great variety of how the public interprets and speaks code. There was not one word or character that all participants spoke in the same manner. Even the simple } has 43 variations of how it was spoken. This means that some participants had to have stated multiple ways they interpreted } at different stages of the code. This is progress though, for now we know a sample of the variety of ways code can be spoken and this can be used to further STT applications. To answer the last question RQ4, it did not perform well. There is a wide area of improvement for the application, but the improvements needed is now known based on the answers to RQ1 and RQ3. This means progress can be made to eventually reach the goal of 100% success. This leads into what future work is needed.

For work to be done, the results for RQ1 and RQ3 need to be integrated into the formatting application to allow it to be accessible to a greater population. As for

Apple's Speech to Text system, everything can always be improved upon and with this in mind the results from RQ3 could be integrated with it to help their system be even more successful for a greater population.

REFERENCES

References A: Included Studies

The following are the references for the articles included in the review:

1) Adetunmbi, O. A., O. O. Obe, and J. N. Iyanda. “Development of Standard Yorùbá Speech-to-Text System Using HTK.” International Journal of Speech Technology 19, no. 4 (December 1, 2016): 929–44. https://doi.org/10.1007/s10772-016-9380-2.

2) Aguilar-Chacon, J.E., and D.A. Segura-Torres. “Evaluation Methodology for Speech To Text Services Similarity and Speed Characteristics Focused on Small Size Computers.” IOP Conference Series: Materials Science and Engineering 844 (June 30, 2020): 012039. https://doi.org/10.1088/1757-899X/844/1/012039.

3) “An Efficient Speech Recognition System for Arm-disabled Students Based on Isolated Words - Darabkh - 2018 - Computer Applications in Engineering Education - Wiley Online Library.” Accessed June 27, 2021. https://onlinelibrary-wileycom.libproxy.uhcl.edu/doi/full/10.1002/cae.21884.

4) “Application of Speech Recognition Technology in Power Grid Dispatching Automation - IOPscience.” Accessed July 8, 2021. https://iopscience-ioporg.libproxy.uhcl.edu/article/10.1088/1757-899X/394/4/042111.

5) Biswas, Astik, Emre Yılmaz, Ewald van der Westhuizen, Febe de Wet, and Thomas Niesler. “Code-Switched Automatic Speech Recognition in Five South African Languages.” Computer Speech & Language, July 1, 2021, 101262. https://doi.org/10.1016/j.csl.2021.101262.

6) Cai, Zhenguang G., Rebecca A. Gilbert, Matthew H. Davis, M. Gareth Gaskell, Lauren Farrar, Sarah Adler, and Jennifer M. Rodd. “Accent Modulates Access to Word Meaning: Evidence for a Speaker-Model Account of Spoken Word

Recognition." Cognitive Psychology 98 (November 1, 2017): 73–101. https://doi.org/10.1016/j.cogpsych.2017.08.003.

7) Contreras, Ruben, Angel Ayala, and Francisco Cruz. "Unmanned Aerial Vehicle Control through Domain-Based Automatic Speech Recognition." Computers 9, no. 3 (September 2020): 75. https://doi.org/10.3390/computers9030075.

8) Dimauro, Giovanni, Vincenzo Di Nicola, Vitoantonio Bevilacqua, Danilo Caivano, and Francesco Girardi. "Assessment of Speech Intelligibility in Parkinson's Disease Using a Speech-To-Text System." IEEE Access 5 (2017): 22199–208. https://doi.org/10.1109/ACCESS.2017.2762475.

9) Gerhana, Y. A., A. R. Atmadja, D. S. Maylawati, A. Rahman, K. Nufus, H. Qodim, Busr, and M. A. Ramdhani. "Computer Speech Recognition to Text for Recite Holy Quran." IOP Conference Series: Materials Science and Engineering 434 (December 2018): 012044. https://doi.org/10.1088/1757-899X/434/1/012044.

10) Gurunath Shivakumar, Prashanth, and Panayiotis Georgiou. "Transfer Learning from Adult to Children for Speech Recognition: Evaluation, Analysis and Recommendations." Computer Speech & Language 63 (September 1, 2020): 101077. https://doi.org/10.1016/j.csl.2020.101077.

11) Huang, Y.-M., C.-J. Liu, R. Shadiev, M.-H. Shen, and W.-Y. Hwang. "Investigating an Application of Speech-to-Text Recognition: A Study on Visual Attention and Learning Behaviour." Journal of Computer Assisted Learning 31, no. 6 (2015): 529–45. https://doi.org/10.1111/jcal.12093.

12) Hussain, Muhammad Azhar, Kamran Ahsan, Sarwat Iqbal, and Adnan Nadeem. "Supporting Deafblind in Congregational Prayer Using Speech Recognition and VibroTactile Stimuli." International Journal of Human-Computer Studies 123 (March 1, 2019): 70–96. https://doi.org/10.1016/j.ijhcs.2018.11.002.

13) Iancu, Bogdan. “Evaluating Google Speech-to-Text API’s Performance for Romanian e-Learning Resources.” Informatica Economica 23, no. 1/2019 (March 30, 2019): 17–25. https://doi.org/10.12948/issn14531305/23.1.2019.02.

14) Junining, Esti, Sony Alif, and Nuria Setiarini. “Automatic Speech Recognition in Computer-Assisted Language Learning for Individual Learning in Speaking.” JEES (Journal of English Educators Society) 5, no. 2 (October 13, 2020): 219–23. https://doi.org/10.21070/jees.v5i2.867.

15) Kumar, Manoj, So Hyun Kim, Catherine Lord, Thomas D. Lyon, and Shrikanth Narayanan. “Leveraging Linguistic Context in Dyadic Interactions to Improve Automatic Speech Recognition for Children.” Computer Speech & Language 63 (September 1, 2020): 101101. https://doi.org/10.1016/j.csl.2020.101101.

16) Kwon, Soonil, Sung-Jae Kim, and Joon Yeon Choeh. “Preprocessing for Elderly Speech Recognition of Smart Devices.” Computer Speech & Language 36 (March 1, 2016): 110–21. https://doi.org/10.1016/j.csl.2015.09.002.

17) Lee, Seongjae, Sunmee Kang, David K Han, and Hanseok Ko. “Dialogue Enabling Speech-to-Text User Assistive Agent System for Hearing-Impaired Person.” Medical & Biological Engineering & Computing 54, no. 6 (June 2016): 915–26. https://doi.org/10.1007/s11517-015-1447-8.

18) Lileikytė, Rasa, Lori Lamel, Jean-Luc Gauvain, and Arseniy Gorin. “Conversational Telephone Speech Recognition for Lithuanian.” Computer Speech & Language 49 (May 1, 2018): 71–82. https://doi.org/10.1016/j.csl.2017.11.005.

19) Lyall, F. C., P. J. Clamp, and D. Hajioff. “Smartphone Speech-to-Text Applications for Communication with Profoundly Deaf Patients.” The Journal of Laryngology and Otology 130, no. 1 (January 2016): 104–6. http://dx.doi.org/10.1017/S0022215115003151.

20) Matthews, Joshua, Junyu Cheng, and John Mitchell O'Toole. "Computer-Mediated Input, Output and Feedback in the Development of L2 Word Recognition from Speech." ReCALL : The Journal of EUROCALL 27, no. 3 (September 2015): 321–39. http://dx.doi.org/10.1017/S0958344014000421.

21) Mirzaei, Maryam Sadat, Kourosh Meshgi, and Tatsuya Kawahara. "Exploiting Automatic Speech Recognition Errors to Enhance Partial and Synchronized Caption for Facilitating Second Language Listening." Computer Speech & Language 49 (May 1, 2018): 17–36. https://doi.org/10.1016/j.csl.2017.11.001.

22) Nenny Anggraini, Angga Kurniawan, Luh Kesuma Wardhani, and Nashrul Hakiem. "Speech Recognition Application for the Speech Impaired Using the Android-Based Google Cloud Speech API." Telkomnika 16, no. 6 (December 2018): 2733–39. https://doi.org/10.12928/TELKOMNIKA.v16i6.9638.

23) Norberg, Ulf, Ursula Stachl-Peier, and Liisa Tiittula. "Speech-to-Text Interpreting in Finland, Sweden and Austria." Translation & Interpreting 7, no. 3 (November 6, 2015): 36–49. https://doi.org/10.12807/t&i.v7i3.418.

24) "Open Source Toolkit for Speech to Text Translation." Accessed June 7, 2021. https://libproxy.uhcl.edu/login?url=https://www.proquest.com/docview/2167894001?Ope nUrlRefId=info:xri/sid:primo&accountid=7108.

25) Remes, Ulpu, Ana Ramírez López, Lauri Juvela, Kalle Palomäki, Guy J. Brown, Paavo Alku, and Mikko Kurimo. "Comparing Human and Automatic Speech Recognition in a Perceptual Restoration Experiment." Computer Speech & Language 35 (January 1, 2016): 14–31. https://doi.org/10.1016/j.csl.2015.06.005.

26) Shadiev, Rustam, Ting-Ting Wu, Ai Sun, and Yueh-Min Huang. "Applications of Speech-to-Text Recognition and Computer-Aided Translation for Facilitating CrossCultural Learning through a Learning Activity: Issues and Their Solutions."

Educational Technology Research & Development 66, no. 1 (February 2018): 191–214. https://doi.org/10.1007/s11423-017-9556-8.

27) Short, Greg, Keikichi Hirose, Mariko Kondo, and Nobuaki Minematsu. “Automatic Recognition of Japanese Vowel Length Accounting for Speaking Rate and Motivated by Perception Analysis.” Speech Communication 73 (October 1, 2015): 47–63. https://doi.org/10.1016/j.specom.2015.07.001.

28) Song, Hye-Jeong, Kim Hong-Ki, Kim Jong-Dae, Park Chan-Young, and Kim YuSeop. “Inter-Sentence Segmentation of YouTube Subtitles Using Long-Short Term Memory (LSTM).” Applied Sciences 9, no. 7 (January 2019). http://dx.doi.org/10.3390/app9071504.

29) “Testing the Use of Voice Input in a Smartphone Web Survey - Melanie Revilla, Mick P. Couper, Oriol J. Bosch, Marc Asensio, 2020.” Accessed June 13, 2021. https://journals-sagepub-com.libproxy.uhcl.edu/doi/full/10.1177/0894439318810715.

30) Vogel, Markus, Wolfgang Kaisers, Ralf Wassmuth, and Ertan Mayatepek. “Analysis of Documentation Speed Using Web-Based Medical Speech Recognition Technology: Randomized Controlled Trial.” Journal of Medical Internet Research 17, no. 11 (November 3, 2015): e5072. https://doi.org/10.2196/jmir.5072.

31) Ziman, Kirsten, Andrew C. Heusser, Paxton C. Fitzpatrick, Campbell E. Field, and Jeremy R. Manning. “Is Automatic Speech-to-Text Transcription Ready for Use in Psychological Experiments?” Behavior Research Methods 50, no. 6 (December 1, 2018): 2597–2605. https://doi.org/10.3758/s13428-018-1037-4.

References B: Removed Articles

The following are the references for the articles included in the review process but removed before the analysis portion:

1) Choi, Wonje, Karthi Duraisamy, Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. “On-Chip Communication Network for Efficient Training of Deep Convolutional Networks on Heterogeneous Manycore Systems.” IEEE Transactions on Computers 67, no. 5 (May 2018): 672–86. https://doi.org/10.1109/TC.2017.2777863.
2) Manor, Ran, and Amir B. Geva. “Convolutional Neural Network for Multi-Category Rapid Serial Visual Presentation BCI.” Frontiers in Computational Neuroscience, December 2, 2015. http://link.gale.com/apps/doc/A465936283/SCIC?sid=bookmarkSCIC&xid=47d55691.
3) Usama, Muhammad, Junaid Qadir, Aunn Raza, Hunain Arif, Kok-lim Alvin Yau, Yehia Elkhatib, Amir Hussain, and Ala Al-Fuqaha. “Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges.” IEEE Access 7 (2019): 65579–615. https://doi.org/10.1109/ACCESS.2019.2916648.
4) Xu, Xingyuan, Mengxi Tan, Bill Corcoran, Jiayang Wu, Andreas Boes, Thach G. Nguyen, Sai T. Chu, et al. “11 TOPS Photonic Convolutional Accelerator for Optical Neural Networks.” Nature 589, no. 7840 (January 7, 2021): 44-51,51A-51E. http://dx.doi.org/10.1038/s41586-020-03063-0.
5) Barfuss, Hendrik, Christian Huemmer, Andreas Schwarz, and Walter Kellermann. “Robust Coherence-Based Spectral Enhancement for Speech Recognition in Adverse Real-World Environments.” Computer Speech & Language 46 (November 1, 2017): 388–400. https://doi.org/10.1016/j.csl.2017.02.005.

6) Barker, Jon, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. “Multi-Microphone Speech Recognition in Everyday Environments.” Computer Speech & Language 46 (November 1, 2017): 386–87. https://doi.org/10.1016/j.csl.2017.02.007.

7) “Building Cognitive Applications with IBM Watson Services: Volume 6 Speech to Text and Text to Speech - Building Cognitive Applications with IBM Watson Services: Volume 6 Speech to Text and Text to Speech [Book].” Accessed June 7, 2021. https://www.oreilly.com/library/view/building-cognitiveapplications/9780738442600/cover.xhtml.

8) Cho, Ji-Won, Jong-Hyeon Park, Joon-Hyuk Chang, and Hyung-Min Park. “Bayesian Feature Enhancement Using Independent Vector Analysis and Reverberation Parameter Re-Estimation for Noisy Reverberant Speech Recognition.” Computer Speech & Language 46 (November 1, 2017): 496–516. https://doi.org/10.1016/j.csl.2017.01.010.

9) Ding, Ing-Jr, and Shih-Kai Lin. “Performance Improvement of Kinect Software Development Kit–Constructed Speech Recognition Using a Client–Server Sensor Fusion Strategy for Smart Human–Computer Interface Control Applications.” IEEE Access 5 (2017): 4154–62. https://doi.org/10.1109/ACCESS.2017.2679116.

10) Diwakar, G., and Veena Karjigi. “Improving Speech to Text Alignment Based on Repetition Detection for Dysarthric Speech.” Circuits, Systems, and Signal Processing 39, no. 11 (November 1, 2020): 5543–67. https://doi.org/10.1007/s00034-020-01419-5.

11) “Enhancing Comprehension of Lecture Content in a Foreign Language as the Medium of Instruction: Comparing Speech-to-Text Recognition With Speech-Enabled Language Translation - Rustam Shadiev, Yu-Cheng Chien, Yueh-Min

Huang, 2020." Accessed June 19, 2021. https://journals.sagepub.com/doi/10.1177/2158244020953177.

12) Fuhl, Wolfgang. "From Perception to Action Using Observed Actions to Learn Gestures." User Modeling and User-Adapted Interaction 31, no. 1 (March 1, 2021): 105–20. https://doi.org/10.1007/s11257-020-09275-3.

13) Ghahabi, Omid, and Javier Hernando. "Restricted Boltzmann Machines for Vector Representation of Speech in Speaker Recognition." Computer Speech & Language 47 (January 1, 2018): 16–29. https://doi.org/10.1016/j.csl.2017.06.007.

14) Kamper, Herman, Aren Jansen, and Sharon Goldwater. "A Segmental Framework for Fully-Unsupervised Large-Vocabulary Speech Recognition." Computer Speech & Language 46 (November 1, 2017): 154–74. https://doi.org/10.1016/j.csl.2017.04.008.

15) Lokesh, S., Priyan Malarvizhi Kumar, M. Ramya Devi, P. Parthasarathy, and C. Gokulnath. "An Automatic Tamil Speech Recognition System by Using Bidirectional Recurrent Neural Network with Self-Organizing Map." Neural Computing & Applications 31, no. 5 (May 2019): 1521–31. https://doi.org/10.1007/s00521-018-3466-5. 16) Maas, Andrew L., Peng Qi, Ziang Xie, Awni Y. Hannun, Christopher T. Lengerich,

16) Daniel Jurafsky, and Andrew Y. Ng. "Building DNN Acoustic Models for Large Vocabulary Speech Recognition." Computer Speech & Language 41 (January 1, 2017): 195–213. https://doi.org/10.1016/j.csl.2016.06.007.

17) Magnuson, James S., Daniel Mirman, Sahil Luthra, Ted Strauss, and Harlan D. Harris. "Interaction in Spoken Word Recognition Models: Feedback Helps." Frontiers in Psychology, April 3, 2018. http://link.gale.com/apps/doc/A533179569/HRCA?sid=bookmarkHRCA&xid=3221bb6b.

18) Media, O'Reilly. "Building Intelligent Apps with Cognitive APIs." Accessed June 13, 2021. http://learning.oreilly.com/library/view/building-intelligentapps/9781492058632/.

19) Moore, A. H., P. Peso Parada, and P. A. Naylor. "Speech Enhancement for Robust Automatic Speech Recognition: Evaluation Using a Baseline System and Instrumental Measures." Computer Speech & Language 46 (November 1, 2017): 574–84. https://doi.org/10.1016/j.csl.2016.11.003.

20) Moritz, Niko, Kamil Adiloğlu, Jörn Anemüller, Stefan Goetze, and Birger Kollmeier. "Multi-Channel Speech Enhancement and Amplitude Modulation Analysis for Noise Robust Automatic Speech Recognition." Computer Speech & Language 46 (November 1, 2017): 558–73. https://doi.org/10.1016/j.csl.2016.11.004.

21) Norris, Dennis, James M. McQueen, and Anne Cutler. "Commentary on 'Interaction in Spoken Word Recognition Models.'" Frontiers in Psychology, August 30, 2018. http://link.gale.com/apps/doc/A552340047/HRCA?sid=bookmarkHRCA&xid=51a0d206.

22) Novoa, José, Josué Fredes, Víctor Poblete, and Néstor Becerra Yoma. "Uncertainty Weighting and Propagation in DNN–HMM-Based Speech Recognition." Computer Speech & Language 47 (January 1, 2018): 30–46. https://doi.org/10.1016/j.csl.2017.06.005.

23) Novotný, Ondřej, Oldřich Plchot, Ondřej Glembek, Jan "Honza" Černocký, and Lukáš Burget. "Analysis of DNN Speech Signal Enhancement for Robust Speaker Recognition." Computer Speech & Language 58 (November 1, 2019): 403–21. https://doi.org/10.1016/j.csl.2019.06.004.

24) Patil, Hemant A., and Maulik C. Madhavi. "Combining Evidences from Magnitude and Phase Information Using VTEO for Person Recognition Using Humming."

Computer Speech & Language 52 (November 1, 2018): 225–56. https://doi.org/10.1016/j.csl.2017.06.009.

25) Rasipuram, Ramya, and Mathew Magimai.-Doss. “Articulatory Feature Based Continuous Speech Recognition Using Probabilistic Lexical Modeling.” Computer Speech & Language 36 (March 1, 2016): 233–59. https://doi.org/10.1016/j.csl.2015.04.003.

26) Rath, Shakti P. “Scalable Algorithms for Unsupervised Clustering of Acoustic Data for Speech Recognition.” Computer Speech & Language 46 (November 1, 2017): 233–48. https://doi.org/10.1016/j.csl.2017.06.001.

27) Sanchez-Cortina, Isaias, Jesús Andrés-Ferrer, Alberto Sanchis, and Alfons Juan. “Speaker-Adapted Confidence Measures for Speech Recognition of Video Lectures.” Computer Speech & Language 37 (May 1, 2016): 11–23. https://doi.org/10.1016/j.csl.2015.10.003.

28) Sivasankaran, Sunit, Emmanuel Vincent, and Irina Illina. “A Combined Evaluation of Established and New Approaches for Speech Recognition in Varied Reverberation Conditions.” Computer Speech & Language 46 (November 1, 2017): 444–60. https://doi.org/10.1016/j.csl.2017.02.003.

29) Smit, Peter, Sami Virpioja, and Mikko Kurimo. “Advances in Subword-Based HMM-DNN Speech Recognition across Languages.” Computer Speech & Language 66 (March 1, 2021): 101158. https://doi.org/10.1016/j.csl.2020.101158.

30) Spille, Constantin, Birger Kollmeier, and Bernd T. Meyer. “Comparing Human and Automatic Speech Recognition in Simple and Complex Acoustic Scenes.” Computer Speech & Language 52 (November 1, 2018): 123–40. https://doi.org/10.1016/j.csl.2018.04.003.

31) Tanaka, Tomohiro, Ryo Masumura, and Takanobu Oba. “Neural Candidate-Aware Language Models for Speech Recognition.” Computer Speech & Language 66 (March 1, 2021): 101157. https://doi.org/10.1016/j.csl.2020.101157.

32) Varjokallio, Matti, Sami Virpioja, and Mikko Kurimo. “Morphologically Motivated Word Classes for Very Large Vocabulary Speech Recognition of Finnish and Estonian.” Computer Speech & Language 66 (March 1, 2021): 101141. https://doi.org/10.1016/j.csl.2020.101141.

33) Anh, Nguyen Tuan, Yongjian Hu, Qianhua He, Tran Thi Ngoc Linh, Hoang Thi Kim Dung, and Chen Guang. “LIS-Net: An End-to-End Light Interior Search Network for Speech Command Recognition.” Computer Speech & Language 65 (January 1, 2021): 101131. https://doi.org/10.1016/j.csl.2020.101131.

34) Barker, Jon, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. “The Third ‘CHiME’ Speech Separation and Recognition Challenge: Analysis and Outcomes.” Computer Speech & Language 46 (November 1, 2017): 605–26. https://doi.org/10.1016/j.csl.2016.10.005.

35) Choi, Yong-Sik, Jin-Gu Kang, Jong Wha J. Joo, and Jin-Woo Jung. “Real-Time Informatized Caption Enhancement Based on Speaker Pronunciation Time Database.” Multimedia Tools and Applications 79, no. 47 (December 1, 2020): 35667–88. https://doi.org/10.1007/s11042-020-09590-2.

36) “Deep Learning with Applications Using Python : Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras [Book].” Accessed June 7, 2021. https://www.oreilly.com/library/view/deep-learning-with/9781484235164/.

37) Diep, N. N., and A. A. Zhdanov. “Neuron-Like Approach to Speech Recognition.” Programming and Computer Software 44, no. 3 (May 1, 2018): 170–80. https://doi.org/10.1134/S0361768818030088.

38) Ebrahim Kafoori, Kian, and Seyed Mohammad Ahadi. “Bounded Cepstral Marginalization of Missing Data for Robust Speech Recognition.” Computer Speech & Language 36 (March 1, 2016): 1–23. https://doi.org/10.1016/j.csl.2015.07.005.

39) Hai, Yanfei, Hoshang Kolivand, Valentina E. Balas, Anand Paul, and Varatharajan Ramachandran. “Computer-Aided Teaching Mode of Oral English Intelligent Learning Based on Speech Recognition and Network Assistance.” Journal of Intelligent & Fuzzy Systems 39, no. 4 (October 2020): 5749–60. https://doi.org/10.3233/JIFS-189052.

40) Hori, Takaaki, Zhuo Chen, Hakan Erdogan, John R. Hershey, Jonathan Le Roux, Vikramjit Mitra, and Shinji Watanabe. “Multi-Microphone Speech Recognition Integrating Beamforming, Robust Feature Extraction, and Advanced DNN/RNN Backend.” Computer Speech & Language 46 (November 1, 2017): 401–18. https://doi.org/10.1016/j.csl.2017.01.013.

41) Pironkov, Gueorgui, Sean UN Wood, and Stéphane Dupont. “Hybrid-Task Learning for Robust Automatic Speech Recognition.” Computer Speech & Language 64 (November 1, 2020): 101103. https://doi.org/10.1016/j.csl.2020.101103.

42) Shahnawazuddin, S., and Rohit Sinha. “Sparse Coding over Redundant Dictionaries for Fast Adaptation of Speech Recognition System.” Computer Speech & Language 43 (May 1, 2017): 1–17. https://doi.org/10.1016/j.csl.2016.10.004.

43) Sinha, Rohit, and S. Shahnawazuddin. “Assessment of Pitch-Adaptive Front-End Signal Processing for Children’s Speech Recognition.” Computer Speech & Language 48 (March 1, 2018): 103–21. https://doi.org/10.1016/j.csl.2017.10.007.

44) Vincent, Emmanuel, Shinji Watanabe, Aditya Arie Nugraha, Jon Barker, and Ricard Marxer. “An Analysis of Environment, Microphone and Data Simulation Mismatches

in Robust Speech Recognition.” Computer Speech & Language 46 (November 1, 2017): 535–57. https://doi.org/10.1016/j.csl.2016.11.005.

45) Yazdani, Reza, Jose-Maria Arnau, and Antonio González. “LAWS: Locality-AWare Scheme for Automatic Speech Recognition.” IEEE Transactions on Computers 69, no. 8 (August 2020): 1197–1208. https://doi.org/10.1109/TC.2020.2991002.

APPENDIX A:

INCLUDED STUDIES

Article ID	Article Name	Author(s)	Year
S1	Accent modulates access to word meaning: Evidence for a speaker-model account of spoken word recognition	Cai, Zhenguang G.; Gilbert, Rebecca A.; Davis, Matthew H.; Gaskell, M. Gareth; Farrar, Lauren; Adler, Sarah; Rodd, Jennifer M.	2017
S2	An efficient speech recognition system for armdisabled students based on isolated words	Sivasankaran, Sunit; Vincent, Emmanuel; Illina, Irina	2017

S3	Analysis of Documentation Speed Using Web-Based Medical Speech Recognition Technology: Randomized Controlled Trial	Vogel, Markus; Kaisers, Wolfgang; Wassmuth, Ralf; Mayatepek, Ertan	2015
S4	Application of Speech Recognition Technology in Power Grid Dispatching Automation	Cho, Ji-Won; Park, Jong-Hyeon; Chang, Joon-Hyuk; Park, Hyung-Min	2017
S5	Applications of speech-to-text recognition and computer-aided translation for facilitating crosscultural learning through a learning activity: issues and their solutions	Shadiev, Rustam; Wu, Ting-Ting; Sun, Ai; Huang, Yueh-Min	2018

S6	Assessment of Speech Intelligibility in Parkinson's Disease Using a Speech-To-Text System	Dimauro, Giovanni; Di Nicola, Vincenzo; Bevilacqua, Vitoantonio; Caivano, Danilo; Girardi, Francesco	2017
S7	Automatic recognition of Japanese vowel length accounting for speaking rate and motivated by perception analysis	Short, Greg; Hirose, Keikichi; Kondo, Mariko; Minematsu, Nobuaki	2015
S8	Automatic speech recognition in computer-assisted language learning for individual learning in speaking	Junining, Esti; Alif, Sony; Setiarini, Nuria	2020
S9	Code-switched automatic speech recognition in five South African languages	Biswas, Astik; Yılmaz, Emre; van der Westhuizen, Ewald; de Wet, Febe; Niesler, Thomas	2021

S10	Comparing human and automatic speech recognition in a perceptual restoration experiment	Remes, Ulpu; Ramírez López, Ana; Juvela, Lauri; Palomäki, Kalle; Brown, Guy J.; Alku, Paavo; Kurimo, Mikko	2016
S11	Computer speech recognition to text for recite Holy Quran	Gerhana, Y. A.; Atmadja, A. R.; Maylawati, D. S.; Rahman, A.; Nufus, K.; Qodim, H.; Busr; Ramdhani, M. A.	2018
S12	Computer-mediated input, output and feedback in the development of L2 word recognition from speech	Matthews, Joshua; Cheng, Junyu; O'Toole, John Mitchell	2015
S13	Conversational telephone speech recognition for Lithuanian	Lileikytė, Rasa; Lamel, Lori; Gauvain, Jean-Luc; Gorin, Arseniy	2018
S14	Development of Standard Yorùbá speech-to-text system using HTK	Adetunmbi, O. A.; Obe, O. O.; Iyanda, J. N.	2016

S15	Dialogue enabling speech-totext user assistive agent system for hearing-impaired person	Lee, Seongjae; Kang, Sunmee; Han, David K; Ko, Hanseok	2016
S16	Evaluating Google Speech-to-Text API's Performance for Romanian e-Learning Resources	Iancu, Bogdan	2019
S17	Evaluation methodology for Speech to Text Services similarity and speed characteristics focused on small size computers	Aguilar-Chacon, J.E.; Segura-Torres, D.A.	2020

S18	Exploiting automatic speech recognition errors to enhance partial and synchronized caption for facilitating second language listening	Mirzaei, Maryam Sadat; Meshgi, Kourosh; Kawahara, Tatsuya	2018
S19	Inter-Sentence Segmentation of YouTube Subtitles Using Long-Short Term Memory (LSTM)	Song, Hye-Jeong; Hong-Ki, Kim; Jong-Dae, Kim; Chan-Young, Park; Yu-Seop, Kim	2019
S20	Investigating an application of speech-to-text recognition: a study on visual attention and learning behaviour	Huang, Y.-M.; Liu, C.J.; Shadiev, R.; Shen, M.-H.; Hwang, W.-Y.	2015

S21	Is automatic speech-to-text transcription ready for use in psychological experiments?	Ziman, Kirsten; Heusser, Andrew C.; Fitzpatrick, Paxton C.; Field, Campbell E.; Manning, Jeremy R.	2018
S22	Leveraging Linguistic Context in Dyadic Interactions to Improve Automatic Speech Recognition for Children	Kumar, Manoj; Kim, So Hyun; Lord, Catherine; Lyon, Thomas D.; Narayanan, Shrikanth	2020
S23	Open Source Toolkit for Speech to Text Translation	Choi, Wonje; Duraisamy, Karthi; Kim, Ryan Gary; Doppa, Janardhan Rao; Pande, Partha Pratim; Marculescu, Diana; Marculescu, Radu	2018
S24	Preprocessing for elderly speech recognition of smart devices	Kwon, Soonil; Kim, Sung-Jae; Choeh, Joon Yeon	2016

S25	Smartphone speech-to-text applications for communication with profoundly deaf patients	Lyall, F. C.; Clamp, P. J.; Hajioff, D.	2016

S26	Speech Recognition Application for the Speech Impaired using the Androidbased Google Cloud Speech API	Nenny Anggraini; Angga Kurniawan; Luh Kesuma Wardhani; Nashrul Hakiem	2018
S27	Speech-to-text interpreting in Finland, Sweden and Austria	Norberg, Ulf; Stachl-Peier, Ursula; Tiittula, Liisa	2015
S28	Supporting deafblind in congregational prayer using speech recognition and vibrotactile stimuli	Hussain, Muhammad Azhar; Ahsan, Kamran; Iqbal, Sarwat; Nadeem, Adnan	2019

S29	Testing the Use of Voice Input in a Smartphone Web Survey - Melanie Revilla, Mick P. Couper, Oriol J. Bosch, Marc Asensio, 2020	Vogel, Markus; Kaisers, Wolfgang; Wassmuth, Ralf; Mayatepek, Ertan	2015
S30	Transfer learning from adult to children for speech recognition: Evaluation, analysis and recommendations	Gurunath Shivakumar, Prashanth; Georgiou, Panayiotis	2020
S31	Unmanned Aerial Vehicle Control through Domain-Based Automatic Speech Recognition	Contreras, Ruben; Ayala, Angel; Cruz, Francisco	2020

APPENDIX B:

REMOVED STUDIES

Article Name	**Author(s)**	**Year**	**Phase Removed**
11 TOPS photonic convolutional accelerator for optical neural networks	Xu, Xingyuan; Tan, Mengxi; Corcoran, Bill; Wu, Jiayang; Boes, Andreas; Nguyen, Thach G.; Chu, Sai T.; Little, Brent E.; Hicks, Damien G.; Morandotti, Roberto; Mitchell, Arnan; Moss, David J.	2021	Title Review

On-Chip Communication Network for Efficient Training of Deep Convolutional Networks on Heterogeneous Manycore Systems	Choi, Wonje; Duraisamy, Karthi; Kim, Ryan Gary; Doppa, Janardhan Rao; Pande, Partha Pratim; Marculescu, Diana; Marculescu, Radu	2018	Title Review
Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges	Usama, Muhammad; Qadir, Junaid; Raza, Aunn; Arif, Hunain; Yau, Kok-lim Alvin; Elkhatib, Yehia; Hussain, Amir; Al-Fuqaha, Ala	2019	Title Review

Convolutional Neural Network for Multi-Category Rapid Serial Visual Presentation BCI	Manor, Ran; Geva, Amir B.	2015	Title Review
A combined evaluation of established and new approaches for speech recognition in varied reverberation conditions	Sivasankaran, Sunit; Vincent, Emmanuel; Illina, Irina	2017	Abstract Review
A segmental framework for fullyunsupervised largevocabulary speech recognition	Kamper, Herman; Jansen, Aren; Goldwater, Sharon	2017	Abstract Review

Advances in subword-based HMMDNN speech recognition across languages	Smit, Peter; Virpioja, Sami; Kurimo, Mikko	2021	Abstract Review
An Automatic Tamil Speech Recognition system by using Bidirectional Recurrent Neural Network with Self-Organizing Map	Lokesh, S.; Malarvizhi Kumar, Priyan; Ramya Devi, M.; Parthasarathy, P.; Gokulnath, C.	2019	Abstract Review
Analysis of DNN Speech Signal Enhancement for Robust Speaker Recognition	Novotný, Ondřej; Plchot, Oldřich; Glembek, Ondřej; Černocký, Jan “Honza”; Burget, Lukáš	2019	Abstract Review

Articulatory feature based continuous speech recognition using probabilistic lexical modeling	Rasipuram, Ramya; Magimai.-Doss, Mathew	2016	Abstract Review

Bayesian feature enhancement using independent vector analysis and reverberation parameter re-estimation for noisy reverberant speech recognition	Cho, Ji-Won; Park, Jong-Hyeon; Chang, Joon-Hyuk; Park, Hyung-Min	2017	Abstract Review

Building Cognitive Applications with IBM Watson Services: Volume 6 Speech to Text and Text to Speech - Building Cognitive Applications with IBM Watson Services: Volume 6 Speech to Text and Text to Speech [Book]			Abstract Review
Building DNN acoustic models for large vocabulary speech recognition	Maas, Andrew L.; Qi, Peng; Xie, Ziang; Hannun, Awni Y.; Lengerich, Christopher T.; Jurafsky, Daniel; Ng, Andrew Y.	2017	Abstract Review

Building Intelligent Apps with Cognitive APIs	Media, O'Reilly		Abstract Review
Combining evidences from magnitude and phase information using VTEO for person recognition using humming	Patil, Hemant A.; Madhavi, Maulik C.	2018	Abstract Review
Commentary on "Interaction in Spoken Word Recognition Models"	Norris, Dennis; McQueen, James M.; Cutler, Anne	2018	Abstract Review
Comparing human and automatic speech recognition in simple and complex acoustic scenes	Spille, Constantin; Kollmeier, Birger; Meyer, Bernd T.	2018	Abstract Review

Enhancing Comprehension of Lecture Content in a Foreign Language as the Medium of Instruction: Comparing Speech-to-Text Recognition With Speech-Enabled Language Translation - Rustam Shadiev, Yu-Cheng Chien, Yueh-Min Huang, 2020			Abstract Review
From perception to action using observed actions to learn gestures	Fuhl, Wolfgang	2021	Abstract Review

Improving Speech to Text Alignment Based on Repetition Detection for Dysarthric Speech	Diwakar, G.; Karjigi, Veena	2020	Abstract Review
Interaction in Spoken Word Recognition Models: Feedback Helps	Magnuson, James S.; Mirman, Daniel; Luthra, Sahil; Strauss, Ted; Harris, Harlan D.	2018	Abstract Review
Morphologically motivated word classes for very large vocabulary speech recognition of Finnish and Estonian	Varjokallio, Matti; Virpioja, Sami; Kurimo, Mikko	2021	Abstract Review

Multi-Channel Speech Enhancement and Amplitude Modulation Analysis for Noise Robust Automatic Speech Recognition	Moritz, Niko; Adiloğlu, Kamil; Anemüller, Jörn; Goetze, Stefan; Kollmeier, Birger	2017	Abstract Review
Multi-microphone speech recognition in everyday environments	Barker, Jon; Marxer, Ricard; Vincent, Emmanuel; Watanabe, Shinji	2017	Abstract Review
Neural candidate-aware language models for speech recognition	Tanaka, Tomohiro; Masumura, Ryo; Oba, Takanobu	2021	Abstract Review
Performance Improvement of Kinect Software Development Kit–Constructed Speech Recognition Using a Client–Server	Ding, Ing-Jr; Lin, Shih-Kai	2017	Abstract Review

Sensor Fusion Strategy for Smart Human–Computer Interface Control Applications			
Restricted Boltzmann machines for vector representation of speech in speaker recognition	Ghahabi, Omid; Hernando, Javier	2018	Abstract Review
Robust coherence-based spectral enhancement for speech recognition in adverse real-world environments	Barfuss, Hendrik; Huemmer, Christian; Schwarz, Andreas; Kellermann, Walter	2017	Abstract Review
Scalable algorithms for unsupervised clustering of acoustic data for speech recognition	Rath, Shakti P.	2017	Abstract Review

Speaker-adapted confidence measures for speech recognition of video lectures	Sanchez-Cortina, Isaias; Andrés-Ferrer, Jesús; Sanchis, Alberto; Juan, Alfons	2016	Abstract Review
Speech enhancement for robust automatic speech recognition: Evaluation using a baseline system and instrumental measures	Moore, A. H.; Peso Parada, P.; Naylor, P. A.	2017	Abstract Review
Uncertainty weighting and propagation in DNN–HMM-based speech recognition	Novoa, José; Fredes, Josué; Poblete, Víctor; Yoma, Néstor Becerra	2018	Abstract Review

APPENDIX C:

RESEARCH QUESTIONS MAPPING

Selected ID	RQ ID	RQ	Answer
S 1	Q1	“The study assesses whether listeners make use of speaker accent in accessing word meanings, and if so, what the mechanism is that supports such accent-based meaning inference.”	“In conclusion, the current study showed that listeners use accent information to establish the dialectic background of the speaker and use their knowledge of that linguistic variety to guide access to the meanings of the words uttered by the speaker.”
S 2	Q2	“In this paper, we propose an efficient system for arm disabled students that allows them to interact with their computers by just giving vocal commands.”	“Utilizing the best values for all parameters in just mentioned techniques, our proposed system achieved a recognition rate of 98.7% using the first approach, and 98.86% using the second approach of which is better in ratio than the first one but slower in processing which is a critical point for a real time system.”

Selected ID	RQ ID	RQ	Answer
S3	Q3	“We hypothesize that the addition of a Web-based, front-end ASR system to the clinical documentation process leads to an increase in documentation speed and documentation amount, and thereby increased physician satisfaction.”	“Average documentation speed without ASR was 173 (SD 101) characters per minute, while it was 217 (SD 120) characters per minute using ASR.” “We conclude that medical documentation with the assistance of Web-based speech recognition leads to an increase in documentation speed, document length, and participant mood when compared to self-typing.”
S4	Q4	“This article mainly expounds the development status of intelligent dispatching system and the application of human-computer interaction technology in intelligent dispatching system.”	“Then use a good voice recognition interface to provide a very effective help for the dispatcher to deal with the problems in the grid work process.”

Selected ID	RQ ID	RQ	Answer
S5	Q5	“In this study, we aimed to: (a) measure the accuracy rate of current STR and CAT technologies...”	“According to our results, the lowest STR accuracy rate was for English (the average was 93.94%), and the highest STR accuracy rate was for French and Hindi (the average was 98.51%).”
S5	Q6	"...(b) explore issues associated with STR and CAT processes and how they can be solved…"	“However, when considering communication on complex and advanced topics, STR and CAT produced more accurate content only for widely used languages that are similar to English (e.g., Russian, French, and Spanish).”

S5	Q7	"...(c) investigate whether the use of STR and CAT applications are a feasible way by which to facilitate cross-cultural learning."	“Finally, our results demonstrated that cross cultural learning took place; the participants understood and could explain foreign traditions to others and could also compare foreign traditions with their own.”
S6	Q8	“In this paper, we present a study about a new way to measure how much speech is intelligible in Parkinson's disease, basing on a public STT system as Google Speech to Text.”	“The results discussed in the paper state that the specific defined protocol and realized software system are useful to show the efficacy of evaluating the intelligibility of speech in Parkinson's Disease by analyzing the variations of the misrecognition of words through a well-known STT public system, such as Google.”

Selected ID	**RQ ID**	**RQ**	**Answer**

S 7	Q9	“To incorporate temporal information into automatic vowel length recognition, a natural approach is to do post-processing on the phoneme alignments obtained by a speech recognizer as in Fig. 1.”	“We obtained results that showed that the duration of the vowel two vowels prior, one vowel prior and one vowel following had an effect on the perception of vowel length.”
S 8	Q10	“In this study, the writer aimed to compile an application using ASR technology which allows students to practice speaking individually and to find the benefits and limitations of ASR for individual learning in speaking.”	“From the result of this study, it can be concluded that the application of ASR is comfortable and can be used for individual learning.”
S 9	Q11	“This paper reports on various strategies that we have evaluated in developing codeswitching ASR systems for five South African languages.”	“Despite the improvements we have achieved, error rates remain high.”
S 10	Q12	“We evaluate our missing data ASR system on a perceptual restoration task, in which the goal is to recognise speech utterances in	“Previous studies indicate that listener performance in perceptual restoration tasks improves as additive-noise level

Selected ID	RQ ID	RQ	Answer
		which acoustic content has either been removed, or substituted with additive noise.”	increases, and reaches an optimal level when the noise is not too intense. In the current work, the same trends were observed in the performance of a missing data ASR system.”
S 11	Q13	“In this research, we build Android mobile app that helps memorize Al-Qur'an on Juz 30 with concept of connection of verse which utilizes speech recognition method.”	“The speech recognition process is done using Google Speech API quite well and can be compared with the original Quran text, with 91% of accuracy. But not yet able to distinguish in detail the Arabic letters in verses of Al- Quran that have similarities in pronunciation.”

Selected ID	**RQ ID**	**RQ**	**Answer**
S12	Q14	“The research presented here seeks to tackle the research problem of how to build increased capacity of learners to recognise L2 words from speech in a real language learning context. In an effort to achieve this objective, a computer application specifically developed to improve L2 WRS was designed and evaluated using the tripartite framework of input, output and feedback.”	“The finding that those participants who used the application experienced significantly greater improvements in L2 WRS than those in the control group provides empirical support for the value of CALL in the development of L2 WRS.”
S13	Q15	“This paper reports on research work aimed at developing conversational telephone speech (CTS) recognition and keyword spotting (KWS) systems for the Lithuanian language.”	“As has been reported using grapheme-based acoustic units for other languages, the phoneme models gave only a slight improvement for the two training conditions (3 or 40h of transcribed audio data).”

S 14	Q16	“This paper focuses on automatically identifying the Standard Yoru`ba´ (a unified language among the native speakers) isolated words spoken by the individual speaker using a syllablebased approach.”	“The result obtained for both bi and tri-syllabic words revealed that this system was a promising approach that could be adopted for Standard Yoru`ba´ continuous speech recognition system.”
S 15	Q17	“In order to provide such service to a hearing-impaired, this paper considers three significant user	“The proposed system was proven to show robust performance against non-stationary noise

Selected ID	**RQ ID**	**RQ**	**Answer**

		requirements. The first is that the device has to be small and portable for usage in daily activities. At minimal, however, a touch screen, microphones, and a speaker have to be integrated into the device. Second requirement is that it needs to be equipped with a robust speech recognition module for providing fast and reliable automated speechto-text (STT) interface to hearingimpaired in a variety of noisy environments. The main reason for speech recognition failure is due mostly to the interfering speech from unintended people nearby and loud vehicle traffic noise on streets. The third crucial requirement is a userfriendly interface for rendering natural and intuitive communication between hearing-impaired and intended speaker of normal hearing.”	compared with existing similar STT systems as it recorded higher speech recognition success rate. Also, the user-friendly interface was shown to achieve highly satisfying user experience.”

Selected ID	RQ ID	RQ	Answer
S16	Q18	“The main objective of this research paper was to obtain an accurate enough ASR model that can process multimedia e-learning resources which contain Romanian speech.”	“The Google Cloud Speechto-Text API obtained a WER of 30.96% for the used dataset, which is a better result that the one obtained by similar works which are using Google Speech Recognition API, but worse than other solutions performed for Romanian in controlled environments with homogenous datasets. Even so, some videos obtained promising results, having an WER of just 9.93%, which gives us hope that by tuning the system properly and by using more qualitative audio recordings, the current model has the potential of obtaining better results.”
S17	Q19	“This work presents some selection criteria and a methodology to evaluate the Speech to Text services, using similarity and speed as metrics and the results obtained	“The faster system was Google’s, and IBM’s was the slowest. The difference between Microsoft’s and Google’s takes around 33.7% less time than

Selected ID	**RQ ID**	**RQ**	**Answer**
		from an experiment deployed on a Raspberry Pi 3 using the proposed methodology with Spanish speakers."	Microsoft's STT system, and around 48% less time than IBM's, its means significative differences between IBM's system and Google's."
S 18	Q20	"In order to overcome the short comings of conventional captioning, we have proposed a novel captioning system called PSC (Mirzaeietal.,2014;MirzaeiandKawahara,2015), which automatically detects difficult words and presents them on the screen to scaffold the L2 listeners, while hiding easy words to encourage more listening than reading."	"The results of the latter experiment revealed that L2 listeners noticeably preferred the enhanced PSC to the baseline and gained better recognition and paraphrasing scores with the enhanced PSC."

S19	Q21	“In this paper, we propose an automatic sentence segmentation method that automatically generates a period mark using deep neural networks to improve the accuracy of automatic translation of YouTube subtitles.”	“In the experiment, the accuracy of the approach was measured to be 70.84%.”

Selected ID	**RQ ID**	**RQ**	**Answer**
S20	Q22	Do students pay visual attention to STR-texts during learning and how much effort do they put into?	“The results showed that the fixation count and percentage of fixation were significantly higher for STR texts than for video of the instructor and slides during both lectures.”
S20	Q23	How differently effective STR-texts can be to influence learning achievement of students with different EFL abilities?	“The results of statistical analyses suggest that STR texts were important media for participants, particularly for students with low EFL ability, to facilitate their comprehension of learning content during the lectures.”

Selected ID	RQ ID	RQ	Answer
S20	Q24	How different are visual attention and learning behaviour of students with different EFL abilities, learning style preferences and gender to use STR-texts?	“According to the results of the questionnaire survey, most participants expressed high perceptions towards usefulness of STR-texts during the first (M = 3.57, sd = 1.08) and second (M = 3.25, sd = 0.97) lectures for learning.”
S20	Q25	What are the perceptions of students towards STR-texts?”	“Based upon these findings, this study suggests employing STR technology to support learning of non native speakers during lectures in English. However, for some participants, it would be more useful to enable ‘pause’ option that allows them to pause the lecture and to take time reading STR-texts thoroughly.”

S21	Q26	“We sought to explore the feasibility of embedding a modern speech-to-text translation engine into a psychological experiment that relies on verbal responses as its primary data source.”	“Overall, we found that the human-generated and computergenerated transcripts matched to a high degree.” “Our results suggest that automated speech-to-text transcription tools are mature enough to provide (within limits) a viable alternative to human annotation.”
S22	Q27	“In this work, we study robust child ASR in a specific spoken interaction setting, namely semi-structured, goal oriented interactions between a child and an adult.”	“In this work, we show that the adult interlocutor’s spoken language is useful in improving child speech recognition accuracy in a child-adult dyadic interaction setting.”

Selected ID	**RQ ID**	**RQ**	**Answer**

S 23	Q28	“In this paper we present an open-source toolkit2 that combines the following components:1. • A CTC and an attention based ASR system2. • A system to generate the punctuation and the casing of the ASR output3. • A neural MT system”	“The attention based system tries to produce output for these segments, which leads to a high number of insertion errors. On the other hand, the CTC model handles this situation well and outputs an empty transcript. The CTC 300 model has a higher error rate, which is mostly due to misspelling of words. This can be fixed by training an additional language model as in Zenkel et al. (2018). Combining all three systems improves the results and yields balanced insertion and deletion errors.”
S 24	Q29	"This study aims to investigate the factors that impair speaking ability among the elderly by comparing speech patterns between the elderly and young adults while also identifying which areas of speech that can be normalized or	“As a result, we concluded that the functional decline of elderly adults’ vocal organs causes the elderly to have a slow average speech rate compared to young adults, while elderly females had a notable longer inter-syllabic silence

Selected ID	RQ ID	RQ	Answer
		adapted for improving speech recognition.”	length and elderly males had no difference in inter-syllabic silence length from that of a young adult’s.”
S 25	Q30	“This study aimed to establish the feasibility of communicating through smartphone speech recognition software compared with writing or typing.”	“Smartphone dictation was less accurate than the other methods (dictation - 92.5 per cent words correct, 95 per cent CI = 89.8–95.1; writing - 99.5 per cent words correct, 95 per cent CI = 99.1–99.9; and typing - 99.8 per cent words correct, 95 per cent CI = 99.5–100.1) (p < 0.001).”

Selected ID	RQ ID	RQ	Answer
S26	Q31	“One effort that can be done is to develop tools or applications that can help speech impairment with normal people to communication.”	“In this paper based on the results of the research as observed by the authors, it can be concluded that the Speech Recognition Application using Google Cloud Speech can recognise and translate the speech of the speech impaired in terms of the digits one to ten.” "A recognition rate of 80% was obtained for the speech impaired and 100% for normal voice speech recognition when speaking the numbers 1 to 10.”

Selected ID	RQ ID	RQ	Answer
S27	Q32	“This paper examines the situation in Finland and Sweden, where STT interpreting training programmes have been available since the 1980s, and Austria, where the first training programme started in 2010, and investigates the norms, values and expectations that guide STT interpreters’ practice in the three countries.”	“In Sweden and Finland, more and more sign and spoken language interpreters are planning to add STT interpreting to their qualifications in order to cope with the increasing number of settings in which STT interpreting is used and the growing demand for STT interpreting in educational contexts where hearing impaired and deaf students want STT interpreting especially in foreign language classes (interviews on 17 March 2014 and 26 May 2014).”
S28	Q33	“This research aimed at exploiting speech recognition for recognition of events of Namaz and vibration feature for conveying event occurrence information to deafblind in congregation Namaz as an assistive tool for deafblind person.”	“It is concluded that if speech recognition technology improves then the SmartPrayerAid can act with greater accuracy.”

S 29	Q34	“Our goal is to test the feasibility of using built-in features of smartphones, rather than an app or third-party software, to record VI. We aim to address four research questions: (1) How well does it work to ask respondents to use VI options to answer open narrative questions? What effect does the use of VI options have on (2) respondents’ behavior, (3) data quality, and (4) survey evaluation?”	“We found that the use of VI in practice still faces a number of challenges. This is especially true of the voice recording option on Android smartphones, where two thirds (63.3%) of respondents failed to answer any of the six open questions, and 26.3% reported some problem completing the survey, compared to 1.5% who failed to answer the six questions and 5.4% who reported problems in the Android text entry control group.”
S 30	Q35	“In this work, we conduct Evaluations on large vocabulary continuous speech recognition (LVCSR) for children, to: 1. Compare older GMM-HMM models and newer DNN models. 2. Investigate different transfer learning	“Our work validated the benefits of age dependent transfer learning and examined the portability and extensibility of models over the different age groups.”
Selected ID	**RQ ID**	**RQ**	**Answer**

		adaptation techniques. Particularly we look at two factors degrading children ASR: acoustic variability and pronunciation variability in a DNN setup. 3. Assess effectiveness of different speaker normalization and adaptation techniques like VTLN, fMLLR, i-vector based adaptation versus the employed transfer learning technique.”	
S 31	Q36	“In this paper, we present an experimental approach for drone control through a cloud-based speech recognition system, improved by a domain-based language.”	“In conclusion, the algorithm obtained high accuracy when interpreting instructions given by an end-user through speech.”

APPENDIX D:

PARTICIPANT MANUAL AUDIO TRANSCRIPTIONS

Participant ID	**Manual Transcription**
1	For zero to one open flower bracket print open parenthesis double colon hello space world double quotes close parenthesis for x zero to one open flower bracket var space test is equal to ten close flower bracket close flower bracket switch open parenthesis food close open parenthesis open flower bracket case taco colon good is equal to true case rice colon good is equal to true case ketchup good is equal to true default colon break close flower bracket while participating greater than skipping open flower bracket if job is equal to is equal to good open flower bracket variable is equal to thank you very much close flower bracket compensation plus is equal to ten close flower bracket
2	For zero to one print hello world for x to one or to zero uh variable test equals to ten if that works uh switch food case taco good equals to true if it is true case rice ah good equals to true case ketchup good equals to false default break and uh while participating skipping if job equals to good var variable response equals to thank you very much ah compensation equals to ten

3	For zero to one ah second line will be print hello world third line will be for x zero to one fourth line will be var test equals to ten and then bracket close well both bracket close then second method switch food bracket is start case one tacos good equals to true case two rice good equals to true case three will be ketchup good equals to false default break bracket while then while method while participant is great participating is

Participant ID	Manual Transcription
	greater than speaking skipping if go ah if job is equals to equals to good var response thank you very much bracket compensation plus equals to ten bracket close
5	For underscore zero ellipses one open bracket print parenthesis open parenthesis quotations hello world close quotations close parenthesis for x zero ellipses one open bracket variable test equals ten close bracket close bracket switch open parentheses food close parenthesis open bracket case open quotation taco close quotations colon good equals true case open quotations rice close quotation colon good equals true case open quotation ketchup close quotation good equals false default quotes default colon break close brackets while participating greater than is greater than skipping open bracket if job is good open bracket variable response equals quote open quotation thank you very much close quotation close bracket compensation plus or equal to ten close bracket

6	For underscore O to I flower brackets open print f double quotes hello world double quotes close end of flower brace for x O to one flower brace var test equals to ten flower brackets flower brackets switch food flower brackets case double quotes taco double quotes semi colon good equals to rice case uh colon rice colon close end of colon good equals to true case uh double quotes ketchup double quotes good equals to false default colon break uh close flower braces while participating greater than skipping flower brackets open if God job equals to equals to good flower bracket ah var response equals to double quotes thank you very much

Participant ID	**Manual Transcription**
	double quotes close end of flower brace compensation plus equals to ten and close flower brace

7	For underscore zero dot dot dot one flower bracket open print bracket open hello world bracket close for x zero dot dot dot one flower bracket open V A R test equal to ten flower bracket close flower bracket close switch bracket open food bracket close flower bracket open case taco semi colon good equal to true case rice semi colon good equal to true case ketchup good equal to false default semi colon break flower bracket close while participating greater than skipping flower bracket open if job equal to equal to good flower bracket open V A R response equal to thank you very much flower bracket close compensation plus equal to ten flower bracket close
8	For underscore zero to one uh flower braces open move to the next line print braces open quotes open hello space world quotes close uh braces close move to the next line for uh x zero to one uh braces open var test equal to ten uh flower braces close and flower braces close switch braces open food braces close flower uh flower brackets open case quotes open taco quotes close uh colon move to the next line uh good equal to true move to the next line case quotes open uh rice quotes close colon move to the next line good equal to true move to the next line uh case uh quotes ketchup quotes close move to the next line good equal to false uh move to the next line default colon uh move to the next line break and uh move to the next line flower bracket uh flower brackets close uh move to

Participant	**Manual Transcription**

ID	
	the next line while participating greater than skipping flower brackets open move to the next line if job equals to equals to good flower brackets open move to the next line var uh response equal to quotes open thank you very much quotes close move to the next line flower brackets close compensation plus equal to ten uh move to the next line uh flower brackets close
9	quotation hello space world double quotation close parenthesis next line for space x space O three periods one flower brackets next line var var space test is equals to ten next line close flower brackets next line close flower brackets uh next line switch open parenthesis food close parenthesis flower bracket next line case double quotations taco double quotation colon next line good is equals to true next line case double quotation rice double quotation colon next line good is equals to true next line case double quotation ketchup double quotation next line good is equals to false next line default colon next line break next line flower bracket close next line while space participating is greater than skipping open flower brackets next line if space job is equals to is equals to good flower brackets open next line var response is equals to double quotation thank you very much double quotation next line close flower brackets next line compensation plus is equals to ten next line close flower braces

10	For I zero to one flower brackets open print hello world uh for x zero to one flower brackets open var test is equal to ten flower brackets close flower brackets close uh switch uh food flower brack flower

Participant ID	Manual Transcription
	brackets open case uh taco colon good is equal is is true case rice good is true case ketchup good is false default break flower brackets close uh while participating greater skipping flower brackets open if job is equal to is equal to good flower brackets open var response is thank you very much flower brackets close compensation plus is equal to ten flower brackets close

11	For underscore zero to one open flower braces print open parenthesis end of end of quotations hello world close the parenthesis for x zero to one open flower brackets var space test equal to ten close flower braces and again close flower braces the next line is switch of food switch open parenthesis food close parenthesis open flower brackets case uh open quotations taco semi colon good equal to true case rice end of parenthesis rice uh colon good equal to true case ketchup end of parenthesis ketchup good equal to false default colon break close flower brackets while participating is greater than skipping open flower brackets if job double equal to good open flower brackets var response is assigned to end of quotations thank you very much close flower brackets compensation plus equal to 10 close the flower brackets
12	For underscore zero until one open curly brace print open parenthesis hello world end inverted quotes close parenthesis for x in zero until one uh open curly brace var test equals ten close curly brace close curly brace switch open parenthesis food close parenthesis open curly brace case inverted colon taco inverted colon and colon good equals true

Participant ID	**Manual Transcription**

	case inverted colon rice inverted colon colon good equals true case uh inverted colon ketchup inverted colon good equals false default colon break close curly brace while participating greater than skipping open curly brace if job equals equals good open curly brace var response equals inverted quotes thank you very much inverted quotes close curly brace compensation plus equals ten close curly brace
13	For underscore zero ellipsis one open curly brace print open parenthesis double quotes hello world close double quotes close parenthesis for x zero ellipsis one open curly brace var test equals ten close curly brace close curly brace switch open parenthesis food close parenthesis open curly brace case double quotes taco colon good equals true case double quotes rice colon good equals true case double quotes ketchup good equals false default colon break close curly brace while participating greater than skipping open curly brace if job double equals good open curly brace var response equals double quotes thank you very much close curly brace compensation plus equals ten close curly brace
14	For wait zero to one open parenthesis print ah semi colon hello world close semi colon for x zero to one open parenthesis variable test equal to ten close parenthesis close parenthesis switch parenthesis food close parenthesis semi um what do you call this what ever you think um open parenthesis case taco ah semi ah two dots good equals to true next case uh rice good equals to true next case ketchup good equals to false else default break close parenthesis while participant greater than skipping

Participant ID	Manual Transcription
	em enter parenthesis if job equals equals good open parenthesis variable response equals thank you very much close parenthesis compensation plus equals ten close parenthesis
15	For zero to one flower bracket open print hello world for x zero to one flower bracket open var test equals to ten flower bracket closed flower bracket closed switch food flower bracket open case taco good is equals to true and case rice good is equals to true case ketchup good is equals to false default break flower bracket closed while participating greater than skipping flower bracket open if job equals to equals to good flower bracket open var response is equals to thank you so much thank you very much flower bracket closed compensation uh plus equals plus equals to ten flower bracket closed
16	For underscore zero to one flower brackets print hello world for x zero to one flower brackets var test is equal to ten switch food case taco good is equal to true case rice good is equal to true case ketchup good is equal to false default break flower brackets while participating greater than skipping flower brackets if job is equal to is equal to good flower bracket var response is equal to thank you very much flower bracket compensation plus is equal to ten flower bracket

Participant ID	Manual Transcription
18	For underscore zero dot dot dot one open parenthesis print um hello world for loop x O dot dot dot one open parenthesis var test equals to ten close parenthesis close parenthesis switch of food open parenthesis case taco good equals to true case rice good equals to true case ketchup good equals to false default break close parenthesis while participate participating greater than skipping open parenthesis if job equals to good open parenthesis var response equals to thank you very much close parenthesis compensation plus equals to ten close parenthesis
19	For underscore zero to one print hello world for x zero to one var test equals ten uh switch food case taco good equals to true case rice good equals true case ketchup good equals false default break while participating greater than skipping if job equals good var response equals thank you very much compensation plus equals ten
20	For zero to one print hello world for x zero to one variable test is equal to zero switch food case taco good is equal to true case rice good is equal to true case ketchup good is equal to false default break while participating is greater than skipping if job is equal to good variable response is equal to thank you very much compensation plus ten

Participant ID	Manual Transcription
21	For uh zero to one print hello world for x uh zero to one variable test is equal to zero switch food case taco good equals to true case rice good eqauls to true case ketchup good equals to false default break while participating is greater than skipping if job is equals to good oh variable response equals to thank you very much and compensation will increment to ten
22	For uh I for uh I equal to zero to one parenthesis open print parenthesis open double quotes hello world end double quotes parenthesis for x goes from zero to one parenthesis open var test equal to ten close parenthesis close parenthesis switch parenthesis open food close parenthesis open parenthesis case double quotes taco double quotes close colon uh good equal to true case double quotes rice end double quotes colon good equal to true case double quotes open ketchup double quotes close good equal to false default colon break parenthesis close while participating is greater than skipping while put while participating is greater than skipping open parenthesis if job equals equals good open parenthesis var response equal to open open double quotes thank you very much close double quotes close parenthesis compensation plus equal to ten close parenthesis

23	For uh zero to one uh open the loop uh print hello world for zero to one uh variable test is equal to ten close uh close the two for loops switch case uh of food case one taco uh if it is true then good is true case two is rice good true case ketchup good is false and default and break close the loop while participating is greater than skipping if job is equal to good uh response is thank you very much else compensation is compensation plus ten
24	For I to one print hello world for I to one variable var variable test equal to ten and two brackets close and after switch case we use food as a keyword in there and case taco uh if its taco good equal to true the next case rice good equal to true and case number three ketchup good equal to false and default break condition when participating greater than skipping in that if job equal to equal to good then response I mean variable

Participant ID	**Manual Transcription**
	response equal to thank you very much and then increase compensation ten for ten

25	Uh for underscore zero to one uh open curly braces print open parenthesis double quotes hello world close the uh parenthesis and next for x zero to one open curly braces var test equals to ten close curly braces and again close the curly braces um in switch condition open parenthesis food um open curly braces case in quotes taco and colon in the next line good equals to true and another case open open quotations rice colon and the next line good equals to true and next case open quotations ketchup and next line good equals to false next default statement colon break close curly braces and next while participating greater greater than skipping open curly braces if job equals to to good open curly braces var response equals to in open quotation quotations thank you very much and close the quotations close the curly brace and next compensation plus equals to ten close the curly braces
26	Yeah for uh for loop from zero to one print hello world inside that for loop for x equal zero to one inside that uh var test equal to ten uh uh take a switch case uh take an input food as an variable and um in the case one it should be taco and good equal to true if the case equal to rice good equal to true case equal to ketchup good equal to false and by default break that's it and while participating is greater than skipping inside that if job equal to good var variable response equal to thank you so very much and compensation equal to compensation plus ten

Participant	**Manual Transcription**

ID	
27	For underscore zero one open braces print hello world end double quotes for x zero one open the braces var test is equal to ten close the braces again end the loop switch food o eh open curly braces case taco good is equals to true case in double quotes rice enter the loop good is equals to true case ketchup in double quotes good is equals to false default break the statement and exit the loop while participating is greater than skipping is greater than skipping enter the loop if job is equals to is equals to good open the loop var response is equals to thank you very much in double quotes end the loop compensation is plus is equals to ten end the loop

28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces print uh brackets uh a pause double quotes hello world double quotes bracket next line for x zero dot dot dot one curly braces var space test is equal to one zero curly braces curly braces space next line switch bracket food bracket space curly braces next line case uh double quotes taco double quotes colon new line good is equal to true new line case double quotes rice double quotes colon new line good is equal to true new line case double quotes ketchup double quotes new line good is equal to false new line default colon new line break curly braces new line while space participating space skipping curly braces new line if space job double equal to good curly braces new line var space response is equal to double quotes thank you very much double quotes new line curly braces new line compensation space plus is equal to ten curly braces

Participant ID	**Manual Transcription**
29	For underscore zero to one print hello world for zero to one variable test equals ten switch food case one taco good equals true case two rice good equals true case three ketchup good equals false default break while participating greater than skipping if job equals good variable response equals thank you very much compensation plus equals ten

30	For underscore zero to one open braces print o open brackets double quotes hello world enter for x x zero to one open brace enter var var code var test is equal to ten close the brace close the brace switch open brace food open brace case taco double quotes taco uh semi colon uh good equals to true case double quotes rice semi colon good equals to true case double quotes ketchup good equals to false default semi colon break close brace while paraphrasing uh greater uh greater than skipping open brace if job equals to equals to good open brace var response equals to double quotes thank you very much o o close the brace competition plus equals to ten close the brace
31	For some integer between zero and one print hello world for some integer x between zero and one variable test equal zero switch food case taco good equals true case rice good equals true case ketchup good equals true default break while participating Is greater than skipping if job equals equals good variable response equals thank you very much compensation plus equals ten
32	For zero to one print hello world for x zero to one var test equals ten switch food case taco good equals true case rice good equals true case

Participant ID	**Manual Transcription**

	ketchup good equals false default break while participating greater than skipping if jobs e uh is equal to good var response equals thank you very much compensation rare is equal to ten
33	or underscore from zero to one um print hello world for x zero to one var test is equal to ten switch food case taco good equal true case rice good equal true case ketchup good equal false default break while participating greater than skipping if job equal good var response equal thank you so much compensation plus equal ten
34	For zero to one open parenthesis print hello world for x equal to zero one open parenthesis var test equal to ten close parenthesis close parenthesis switch open brackets food close brackets open parenthesis case taco good equal to true case rice uh uh colon good equal to true case ketchup good equal to false default uh colon break close parenthesis while participating great greater than skipping uh close parent open parenthesis if job equal to equal to good open parenthesis var response equal to thank you very much close parenthesis compensation plus uh plus or equal to ten close parenthesis
35	For zero to one print hello world and for x for zero to one assign ten to test variable exit the loop add a switch statement and if the case is taco then good true is assigned to good and if case is rice then true is assigned to good and is case k uh case is ketchup then false is assigned to good add a default and a break skip add a while loop which runs as long as the participating is greater than skipping and inside the while if job is

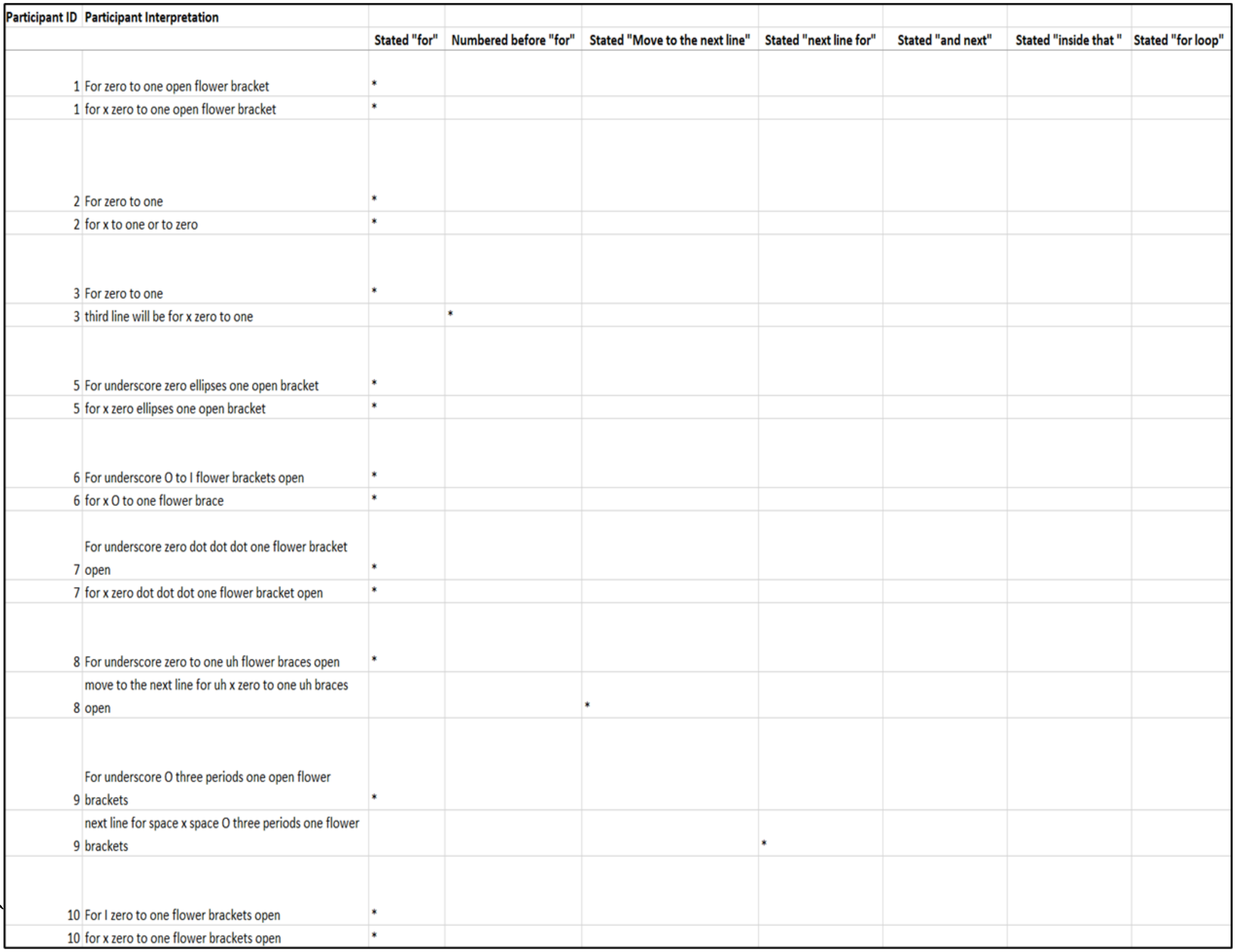
Participant ID	Manual Transcription
	good then add a response thank you very much add oh compensation variable to ten
36	Right for zero to one print hello world for x equals zero to one variable test equals ten switch of food case one taco good equals to true case two rice good equals true case three ketchup good equals false default break while participating is greater than skipping if job equals good variable response equals in hyphens thank you very much and compensation equals compensation plus ten
37	For loop uh zero to one print hello world then for loop zero x zero to one variable test equal to ten close close the loop then close the up upper loop switch uh with the case of food and then case one taco and case taco if good equal to true case rice good equal to true case ketchup good equal to false default break close the switch loop while participating greater than skipping loop if condition job doubles good to two good variable response equal to thank you very much close the if loop compensation equal to increment of compensation by ten close the while loop

Participant ID	Manual Transcription
38	For underscore zero to one print hello world for x zero to one var test equal to ten close for loop close outer for loop switch food case taco good equal to true case rice good equal to true case ketchup good equal to false default break close switch loop while participating greater than skipping if job equal to equal to good initialize response equal to thank you very much close if loop compensation equal to compensation plus ten close while loop

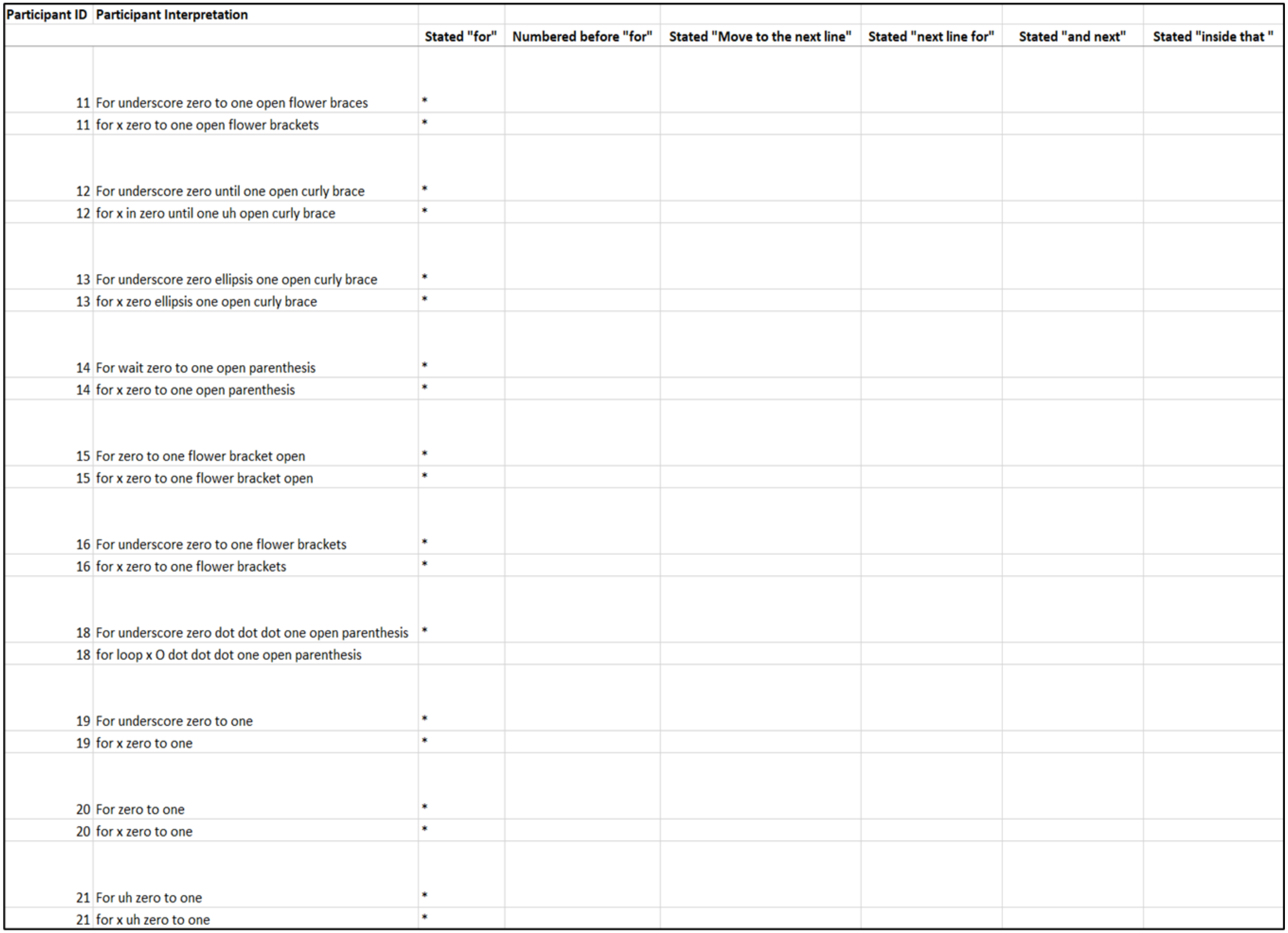
APPENDIX E:

WORD AND CHARACTER PARTICIPANT INTERPRETATIONS

E.1) For

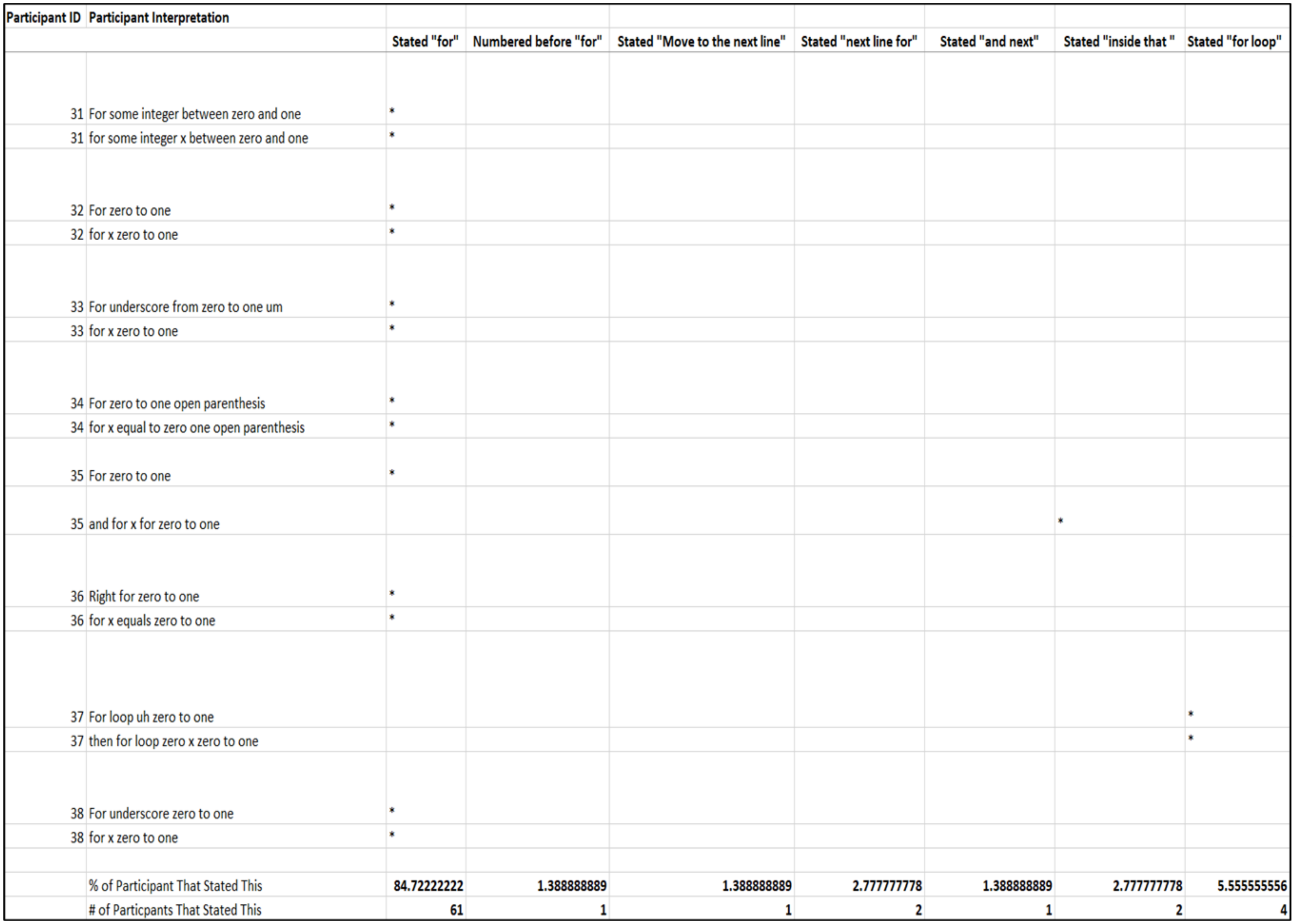


Participant ID	Participant Interpretation							
		Stated "for"	Numbered before "for"	Stated "Move to the next line"	Stated "next line for"	Stated "and next"	Stated "inside that "	Stated "for loop"
1	For zero to one open flower bracket	*						
1	for x zero to one open flower bracket	*						
2	For zero to one	*						
2	for x to one or to zero	*						
3	For zero to one	*						
3	third line will be for x zero to one		*					
5	For underscore zero ellipses one open bracket	*						
5	for x zero ellipses one open bracket	*						
6	For underscore O to I flower brackets open	*						
6	for x O to one flower brace	*						
7	For underscore zero dot dot dot one flower bracket open	*						
7	for x zero dot dot dot one flower bracket open	*						
8	For underscore zero to one uh flower braces open	*						
8	move to the next line for uh x zero to one uh braces open			*				
9	For underscore O three periods one open flower brackets	*						
9	next line for space x space O three periods one flower brackets				*			
10	For I zero to one flower brackets open	*						
10	for x zero to one flower brackets open	*						



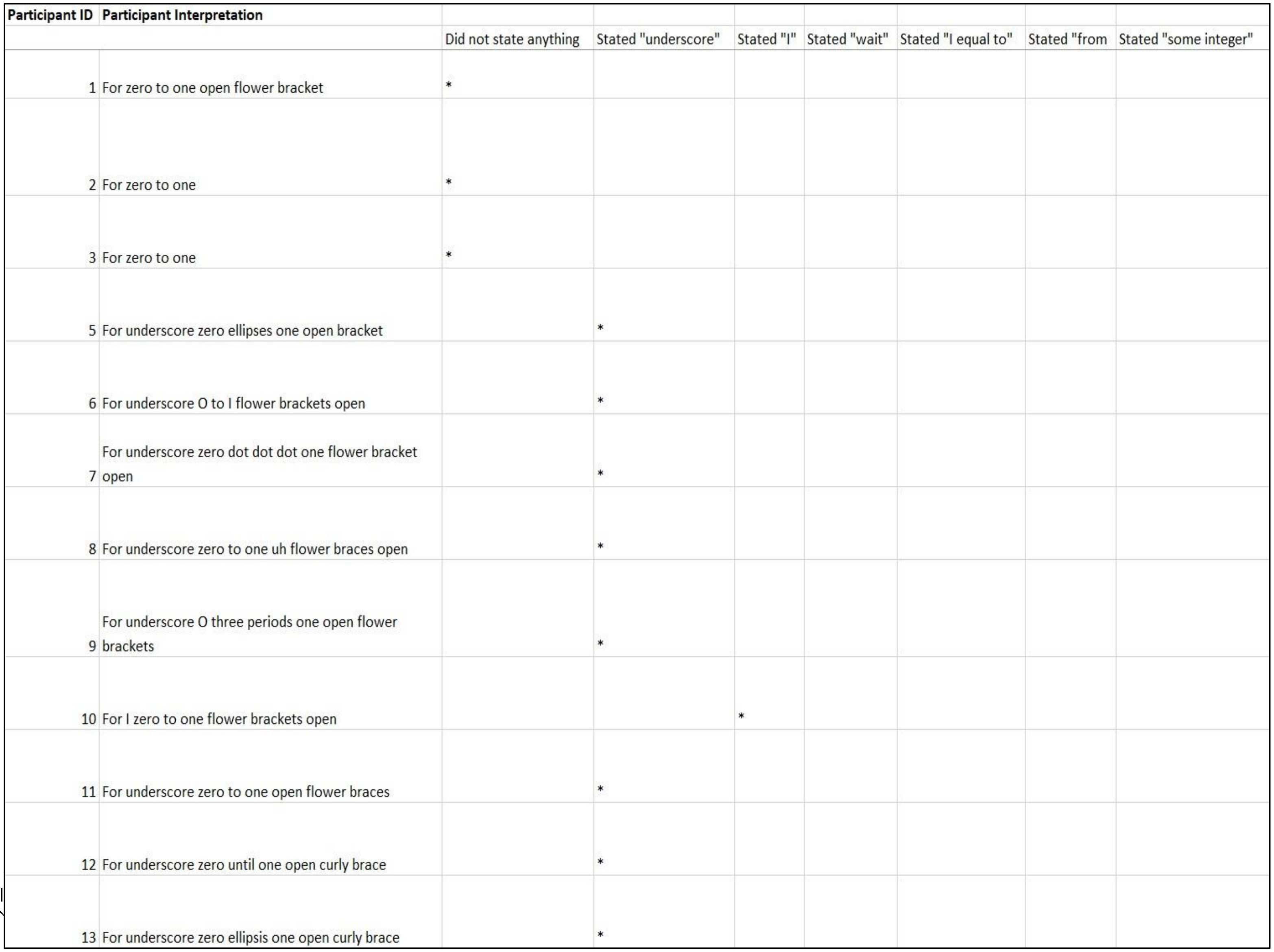
Participant ID	Participant Interpretation						
		Stated "for"	Numbered before "for"	Stated "Move to the next line"	Stated "next line for"	Stated "and next"	Stated "inside that "
11	For underscore zero to one open flower braces	*					
11	for x zero to one open flower brackets	*					
12	For underscore zero until one open curly brace	*					
12	for x in zero until one uh open curly brace	*					
13	For underscore zero ellipsis one open curly brace	*					
13	for x zero ellipsis one open curly brace	*					
14	For wait zero to one open parenthesis	*					
14	for x zero to one open parenthesis	*					
15	For zero to one flower bracket open	*					
15	for x zero to one flower bracket open	*					
16	For underscore zero to one flower brackets	*					
16	for x zero to one flower brackets	*					
18	For underscore zero dot dot dot one open parenthesis	*					
18	for loop x O dot dot dot one open parenthesis						
19	For underscore zero to one	*					
19	for x zero to one	*					
20	For zero to one	*					
20	for x zero to one	*					
21	For uh zero to one	*					
21	for x uh zero to one	*					

Participant ID	Participant Interpretation							
		Stated "for"	Numbered before "for"	Stated "Move to the next line"	Stated "next line for"	Stated "and next"	Stated "inside that "	Stated "for loop"
22	For uh I for uh I equal to zero to one parenthesis open	*						
22	for x goes from zero to one parenthesis open	*						
23	For uh zero to one uh open the loop uh	*						
23	for zero to one uh	*						
24	For I to one	*						
24	for I to one	*						
25	Uh for underscore zero to one uh open curly braces	*						
25	and next for x zero to one open curly braces					*		
26	Yeah for uh for loop from zero to one							*
26	inside that for loop for x equal zero to one						*	
27	For underscore zero one open braces	*						
27	for x zero one open the braces	*						
28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces	*						
28	next line for x zero dot dot dot one curly braces				*			
29	For underscore zero to one	*						
29	for zero to one	*						
30	For underscore zero to one open braces	*						
30	enter for x x zero to one open brace	*						

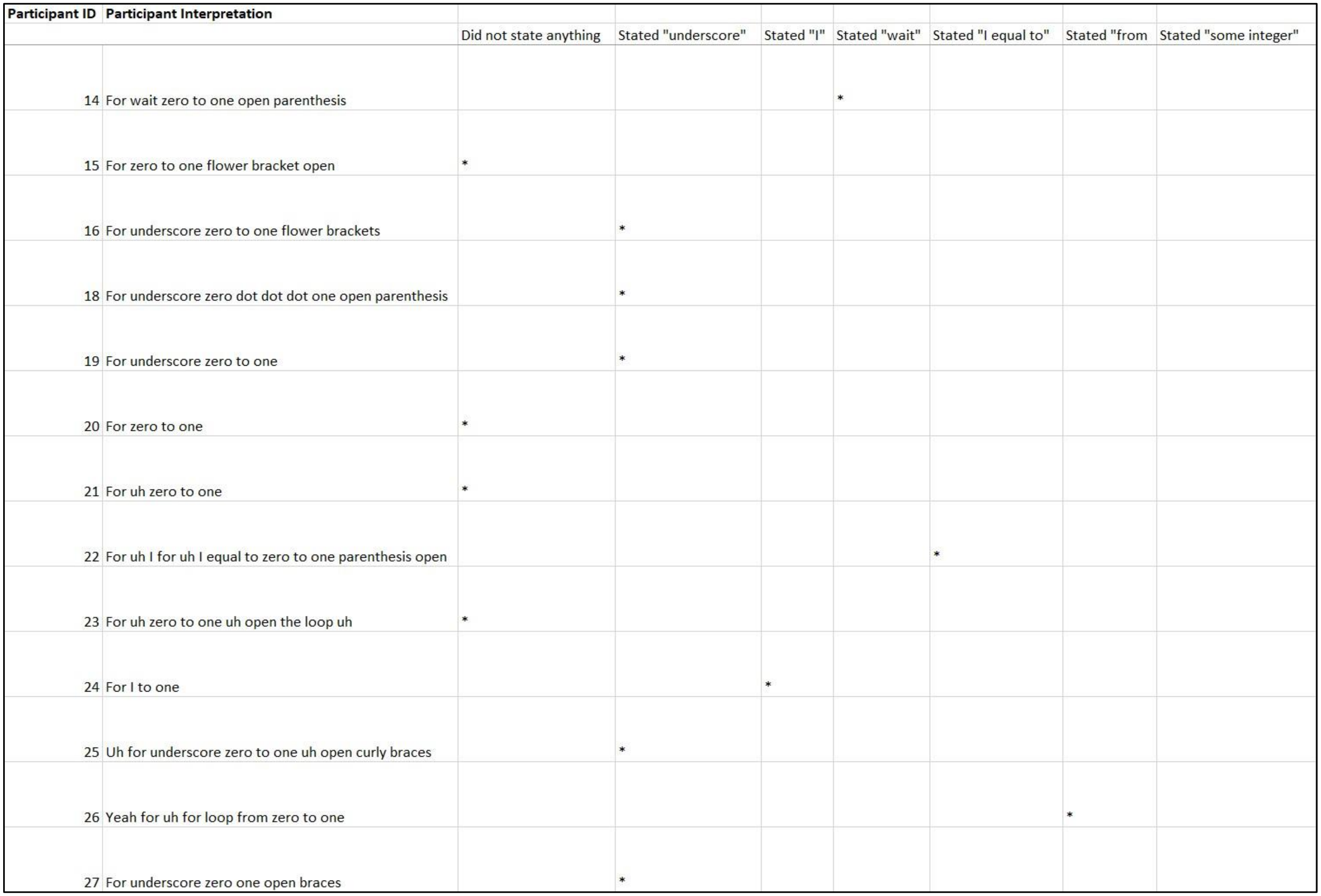


Participant ID	Participant Interpretation	Stated "for"	Numbered before "for"	Stated "Move to the next line"	Stated "next line for"	Stated "and next"	Stated "inside that "	Stated "for loop"
31	For some integer between zero and one	*						
31	for some integer x between zero and one	*						
32	For zero to one	*						
32	for x zero to one	*						
33	For underscore from zero to one um	*						
33	for x zero to one	*						
34	For zero to one open parenthesis	*						
34	for x equal to zero one open parenthesis	*						
35	For zero to one	*						
35	and for x for zero to one						*	
36	Right for zero to one	*						
36	for x equals zero to one	*						
37	For loop uh zero to one							*
37	then for loop zero x zero to one							*
38	For underscore zero to one	*						
38	for x zero to one	*						
	% of Participant That Stated This	**84.72222222**	**1.388888889**	**1.388888889**	**2.777777778**	**1.388888889**	**2.777777778**	**5.555555556**
	# of Particpants That Stated This	**61**	**1**	**1**	**2**	**1**	**2**	**4**

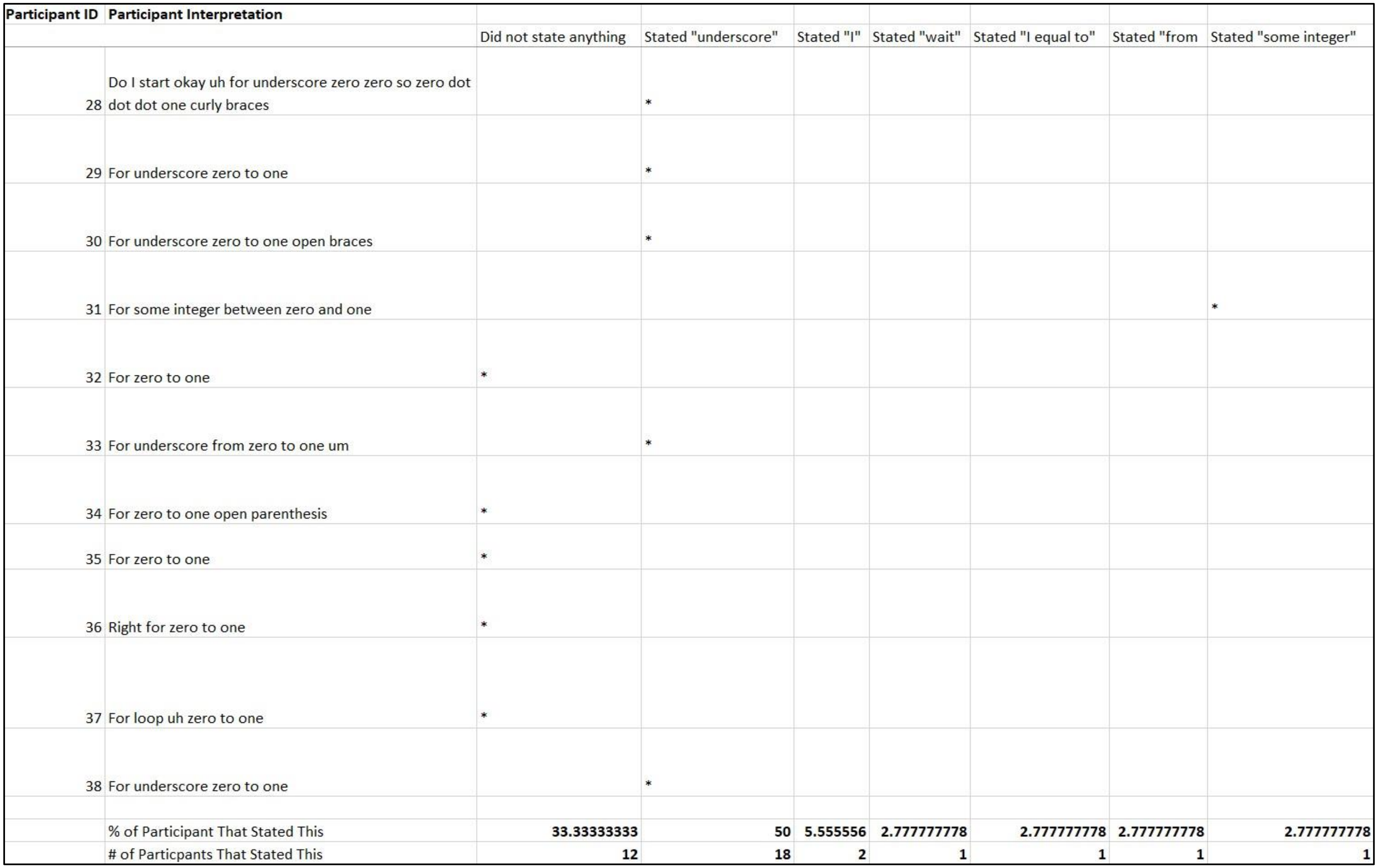
E.2) _



Participant ID	Participant Interpretation							
		Did not state anything	Stated "underscore"	Stated "I"	Stated "wait"	Stated "I equal to"	Stated "from	Stated "some integer"
1	For zero to one open flower bracket	*						
2	For zero to one	*						
3	For zero to one	*						
5	For underscore zero ellipses one open bracket		*					
6	For underscore O to I flower brackets open		*					
7	For underscore zero dot dot dot one flower bracket open		*					
8	For underscore zero to one uh flower braces open		*					
9	For underscore O three periods one open flower brackets		*					
10	For I zero to one flower brackets open			*				
11	For underscore zero to one open flower braces		*					
12	For underscore zero until one open curly brace		*					
13	For underscore zero ellipsis one open curly brace		*					

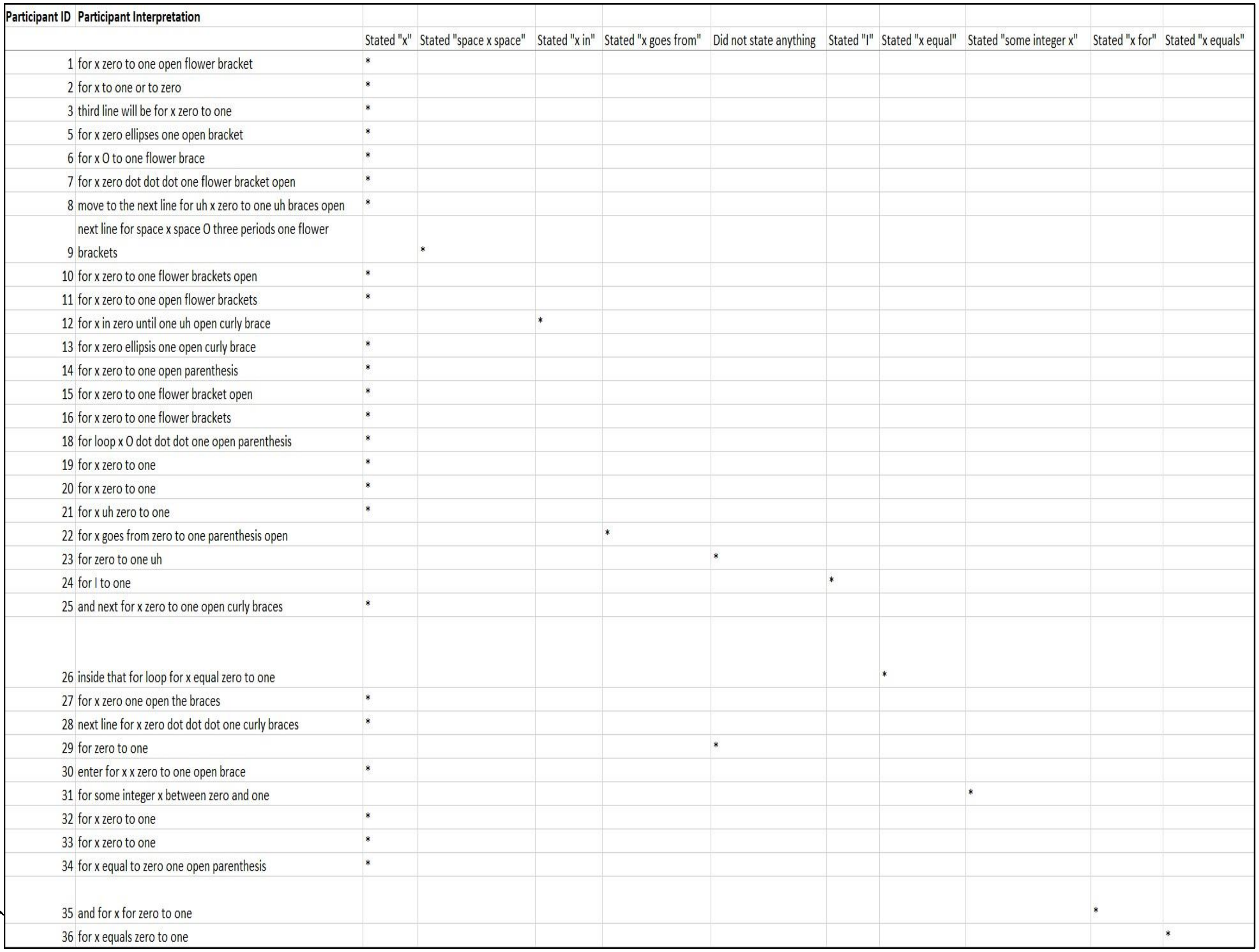


Participant ID	Participant Interpretation							
		Did not state anything	Stated "underscore"	Stated "I"	Stated "wait"	Stated "I equal to"	Stated "from	Stated "some integer"
14	For wait zero to one open parenthesis				*			
15	For zero to one flower bracket open	*						
16	For underscore zero to one flower brackets		*					
18	For underscore zero dot dot dot one open parenthesis		*					
19	For underscore zero to one		*					
20	For zero to one	*						
21	For uh zero to one	*						
22	For uh I for uh I equal to zero to one parenthesis open					*		
23	For uh zero to one uh open the loop uh	*						
24	For I to one			*				
25	Uh for underscore zero to one uh open curly braces		*					
26	Yeah for uh for loop from zero to one						*	
27	For underscore zero one open braces		*					

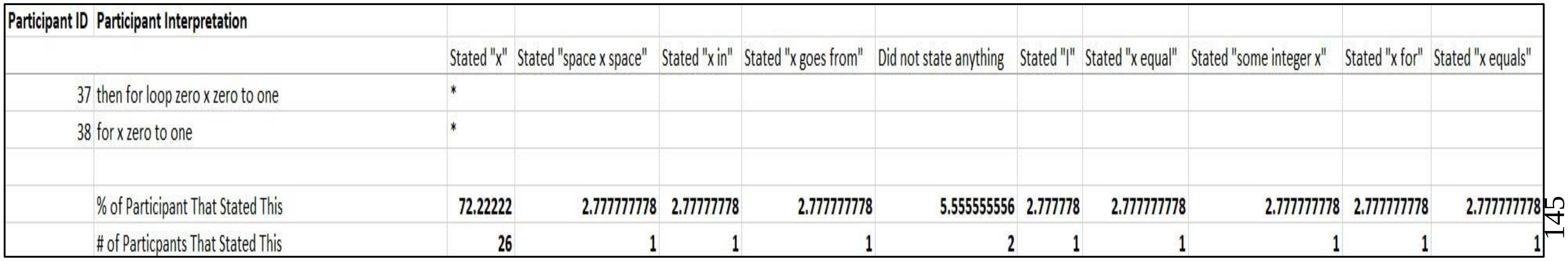


Participant ID	Participant Interpretation							
		Did not state anything	Stated "underscore"	Stated "I"	Stated "wait"	Stated "I equal to"	Stated "from	Stated "some integer"
28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces		*					
29	For underscore zero to one		*					
30	For underscore zero to one open braces		*					
31	For some integer between zero and one							*
32	For zero to one	*						
33	For underscore from zero to one um		*					
34	For zero to one open parenthesis	*						
35	For zero to one	*						
36	Right for zero to one	*						
37	For loop uh zero to one	*						
38	For underscore zero to one		*					
	% of Participant That Stated This	**33.33333333**	**50**	**5.555556**	**2.777777778**	**2.777777778**	**2.777777778**	**2.777777778**
	# of Particpants That Stated This	**12**	**18**	**2**	**1**	**1**	**1**	**1**

E.3) x

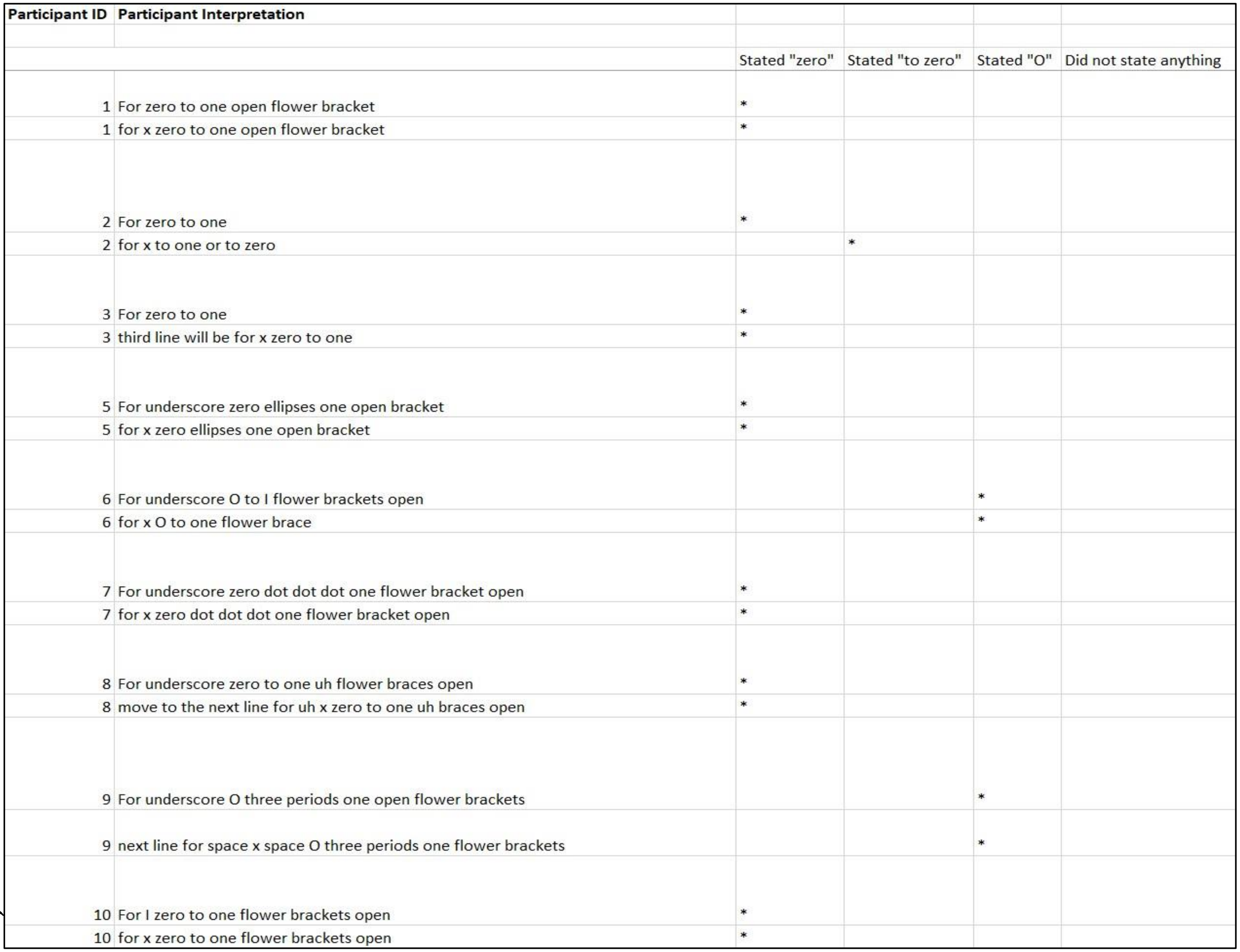


Participant ID	Participant Interpretation										
		Stated "x"	Stated "space x space"	Stated "x in"	Stated "x goes from"	Did not state anything	Stated "I"	Stated "x equal"	Stated "some integer x"	Stated "x for"	Stated "x equals"
1	for x zero to one open flower bracket	*									
2	for x to one or to zero	*									
3	third line will be for x zero to one	*									
5	for x zero ellipses one open bracket	*									
6	for x O to one flower brace	*									
7	for x zero dot dot dot one flower bracket open	*									
8	move to the next line for uh x zero to one uh braces open	*									
9	next line for space x space O three periods one flower brackets		*								
10	for x zero to one flower brackets open	*									
11	for x zero to one open flower brackets	*									
12	for x in zero until one uh open curly brace			*							
13	for x zero ellipsis one open curly brace	*									
14	for x zero to one open parenthesis	*									
15	for x zero to one flower bracket open	*									
16	for x zero to one flower brackets	*									
18	for loop x O dot dot dot one open parenthesis	*									
19	for x zero to one	*									
20	for x zero to one	*									
21	for x uh zero to one	*									
22	for x goes from zero to one parenthesis open				*						
23	for zero to one uh					*					
24	for I to one						*				
25	and next for x zero to one open curly braces	*									
26	inside that for loop for x equal zero to one							*			
27	for x zero one open the braces	*									
28	next line for x zero dot dot dot one curly braces	*									
29	for zero to one					*					
30	enter for x x zero to one open brace	*									
31	for some integer x between zero and one								*		
32	for x zero to one	*									
33	for x zero to one	*									
34	for x equal to zero one open parenthesis	*									
35	and for x for zero to one									*	
36	for x equals zero to one										*

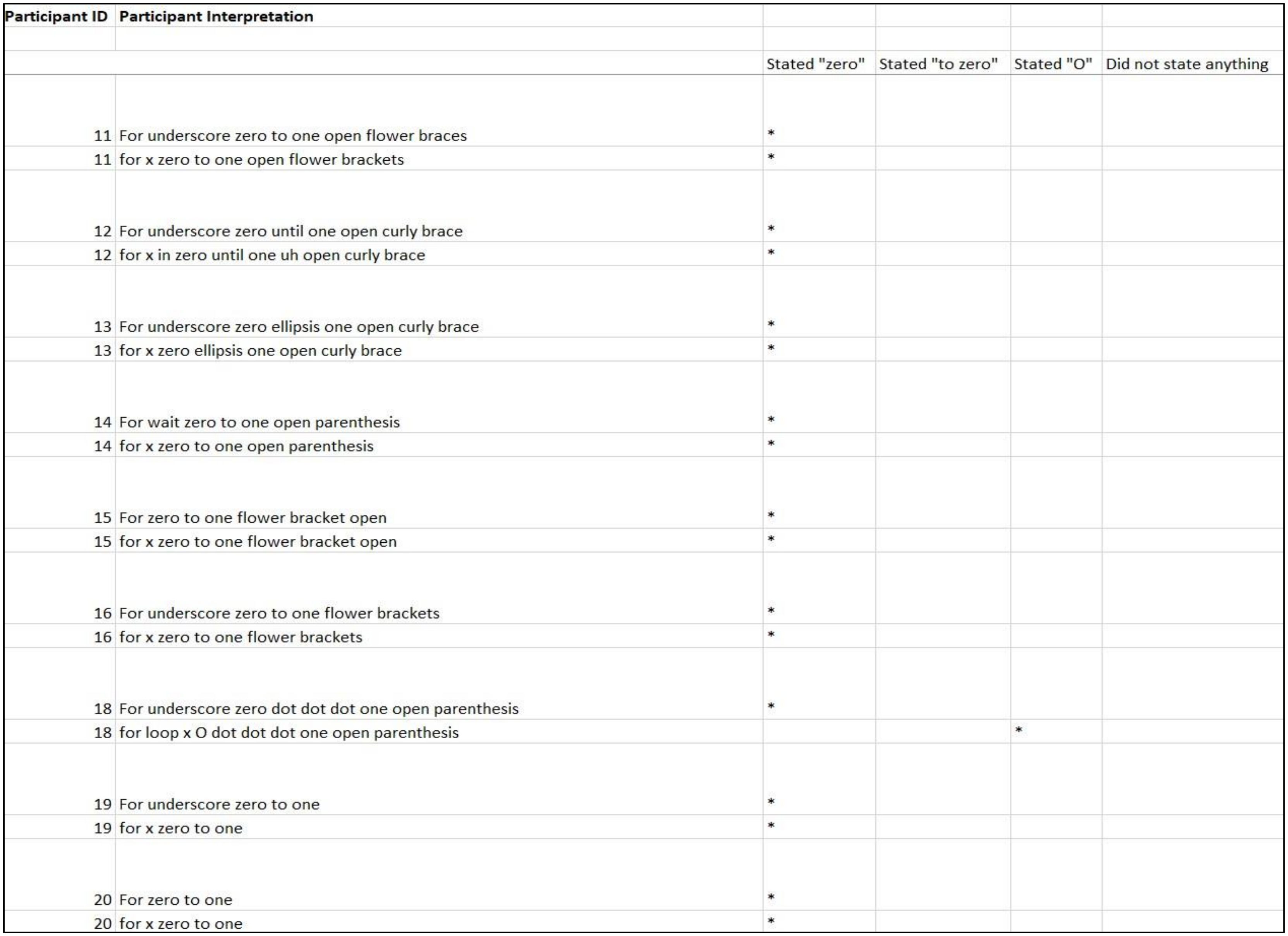


Participant ID	Participant Interpretation										
		Stated "x"	Stated "space x space"	Stated "x in"	Stated "x goes from"	Did not state anything	Stated "I"	Stated "x equal"	Stated "some integer x"	Stated "x for"	Stated "x equals"
37	then for loop zero x zero to one	*									
38	for x zero to one	*									
	% of Participant That Stated This	**72.22222**	**2.777777778**	**2.77777778**	**2.777777778**	**5.555555556**	**2.777778**	**2.777777778**	**2.777777778**	**2.777777778**	**2.777777778**
	# of Particpants That Stated This	**26**	**1**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**

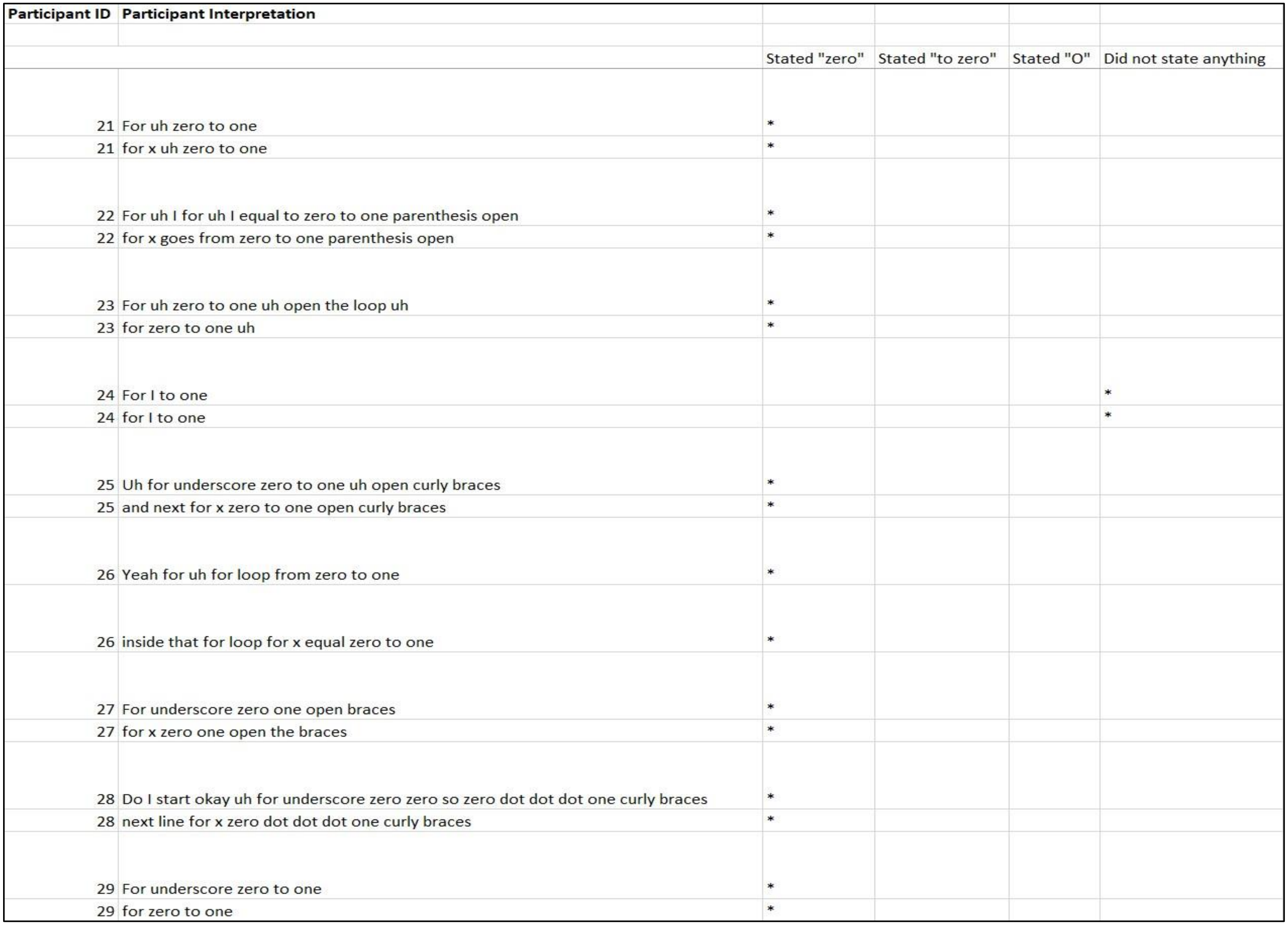
E.4) 0



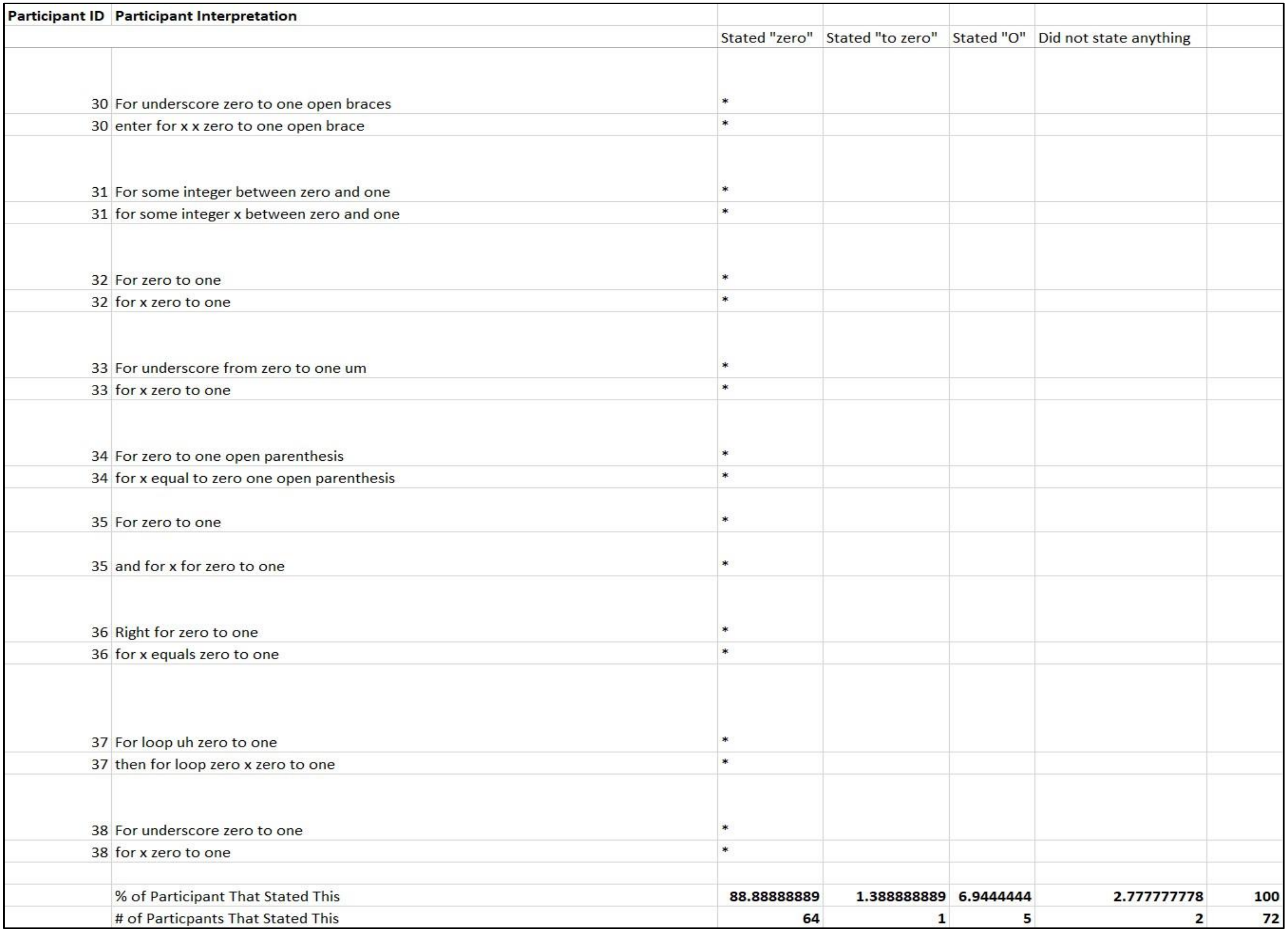
Participant ID	Participant Interpretation				
		Stated "zero"	Stated "to zero"	Stated "O"	Did not state anything
1	For zero to one open flower bracket	*			
1	for x zero to one open flower bracket	*			
2	For zero to one	*			
2	for x to one or to zero		*		
3	For zero to one	*			
3	third line will be for x zero to one	*			
5	For underscore zero ellipses one open bracket	*			
5	for x zero ellipses one open bracket	*			
6	For underscore O to I flower brackets open			*	
6	for x O to one flower brace			*	
7	For underscore zero dot dot dot one flower bracket open	*			
7	for x zero dot dot dot one flower bracket open	*			
8	For underscore zero to one uh flower braces open	*			
8	move to the next line for uh x zero to one uh braces open	*			
9	For underscore O three periods one open flower brackets			*	
9	next line for space x space O three periods one flower brackets			*	
10	For I zero to one flower brackets open	*			
10	for x zero to one flower brackets open	*			



Participant ID	Participant Interpretation				
		Stated "zero"	Stated "to zero"	Stated "O"	Did not state anything
11	For underscore zero to one open flower braces	*			
11	for x zero to one open flower brackets	*			
12	For underscore zero until one open curly brace	*			
12	for x in zero until one uh open curly brace	*			
13	For underscore zero ellipsis one open curly brace	*			
13	for x zero ellipsis one open curly brace	*			
14	For wait zero to one open parenthesis	*			
14	for x zero to one open parenthesis	*			
15	For zero to one flower bracket open	*			
15	for x zero to one flower bracket open	*			
16	For underscore zero to one flower brackets	*			
16	for x zero to one flower brackets	*			
18	For underscore zero dot dot dot one open parenthesis	*			
18	for loop x O dot dot dot one open parenthesis			*	
19	For underscore zero to one	*			
19	for x zero to one	*			
20	For zero to one	*			
20	for x zero to one	*			

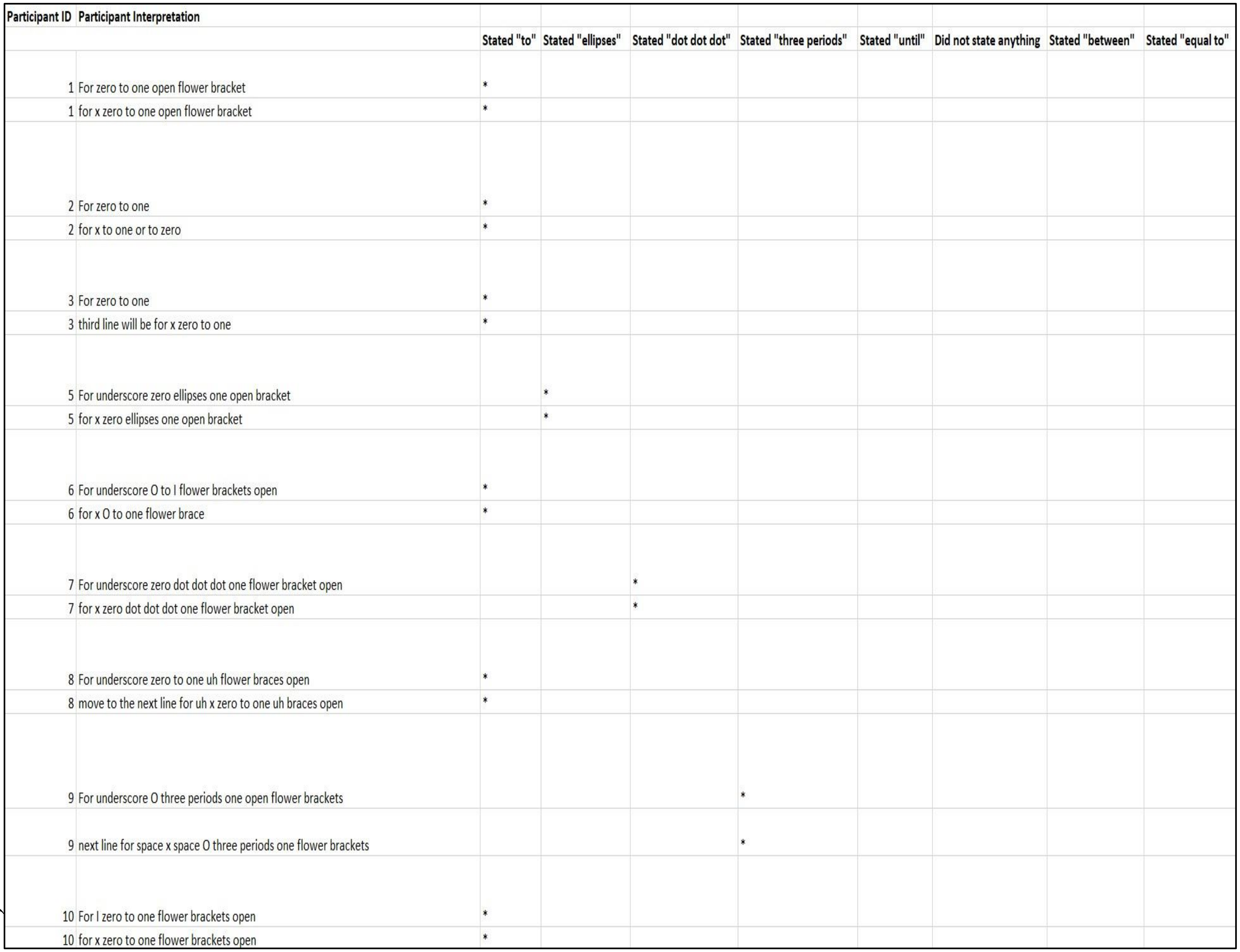


Participant ID	Participant Interpretation				
		Stated "zero"	Stated "to zero"	Stated "O"	Did not state anything
21	For uh zero to one	*			
21	for x uh zero to one	*			
22	For uh I for uh I equal to zero to one parenthesis open	*			
22	for x goes from zero to one parenthesis open	*			
23	For uh zero to one uh open the loop uh	*			
23	for zero to one uh	*			
24	For I to one				*
24	for I to one				*
25	Uh for underscore zero to one uh open curly braces	*			
25	and next for x zero to one open curly braces	*			
26	Yeah for uh for loop from zero to one	*			
26	inside that for loop for x equal zero to one	*			
27	For underscore zero one open braces	*			
27	for x zero one open the braces	*			
28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces	*			
28	next line for x zero dot dot dot one curly braces	*			
29	For underscore zero to one	*			
29	for zero to one	*			

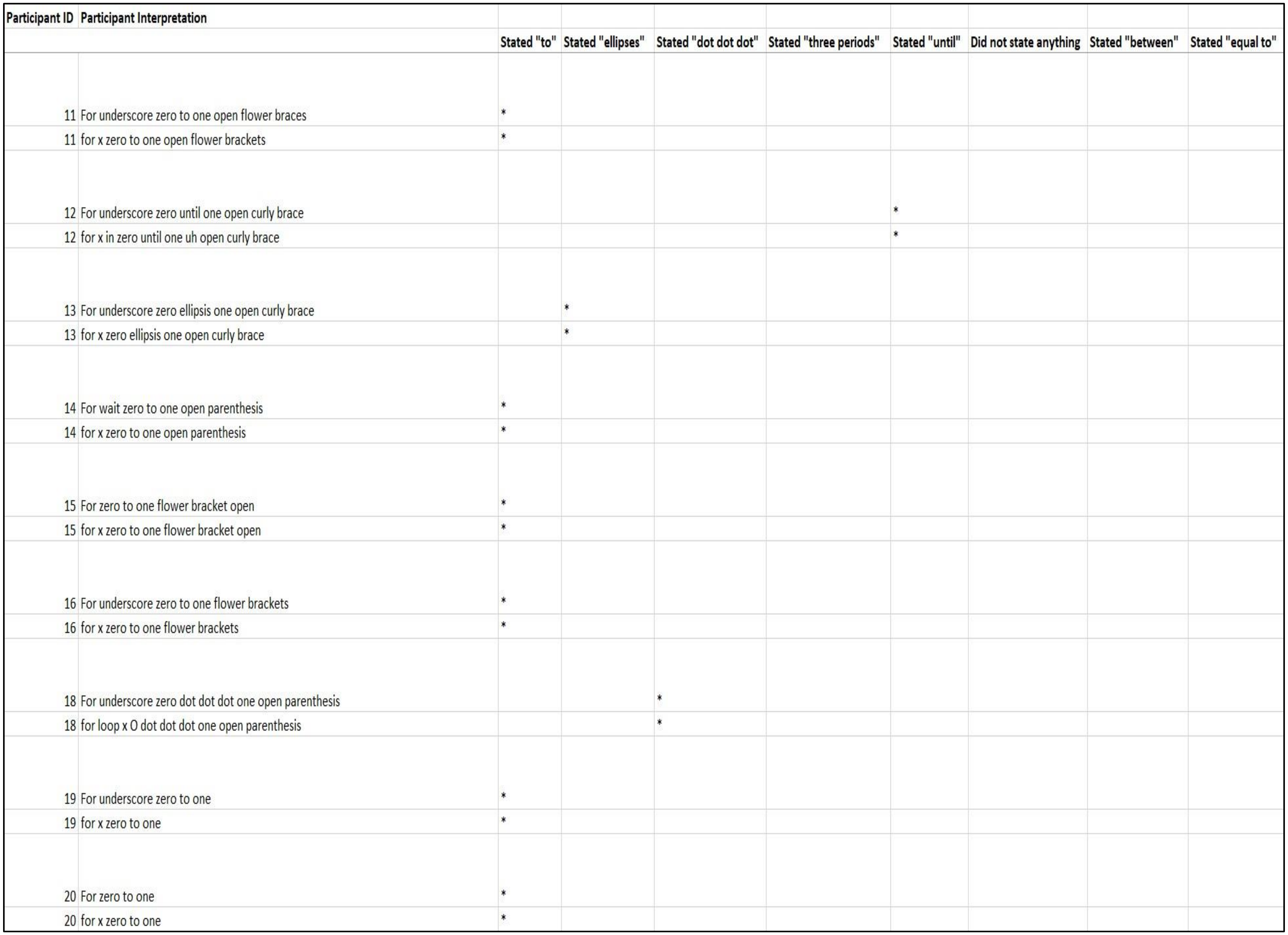


Participant ID	Participant Interpretation					
		Stated "zero"	Stated "to zero"	Stated "O"	Did not state anything	
30	For underscore zero to one open braces	*				
30	enter for x x zero to one open brace	*				
31	For some integer between zero and one	*				
31	for some integer x between zero and one	*				
32	For zero to one	*				
32	for x zero to one	*				
33	For underscore from zero to one um	*				
33	for x zero to one	*				
34	For zero to one open parenthesis	*				
34	for x equal to zero one open parenthesis	*				
35	For zero to one	*				
35	and for x for zero to one	*				
36	Right for zero to one	*				
36	for x equals zero to one	*				
37	For loop uh zero to one	*				
37	then for loop zero x zero to one	*				
38	For underscore zero to one	*				
38	for x zero to one	*				
	% of Participant That Stated This	**88.88888889**	**1.388888889**	**6.9444444**	**2.777777778**	**100**
	# of Particpants That Stated This	**64**	**1**	**5**	**2**	**72**

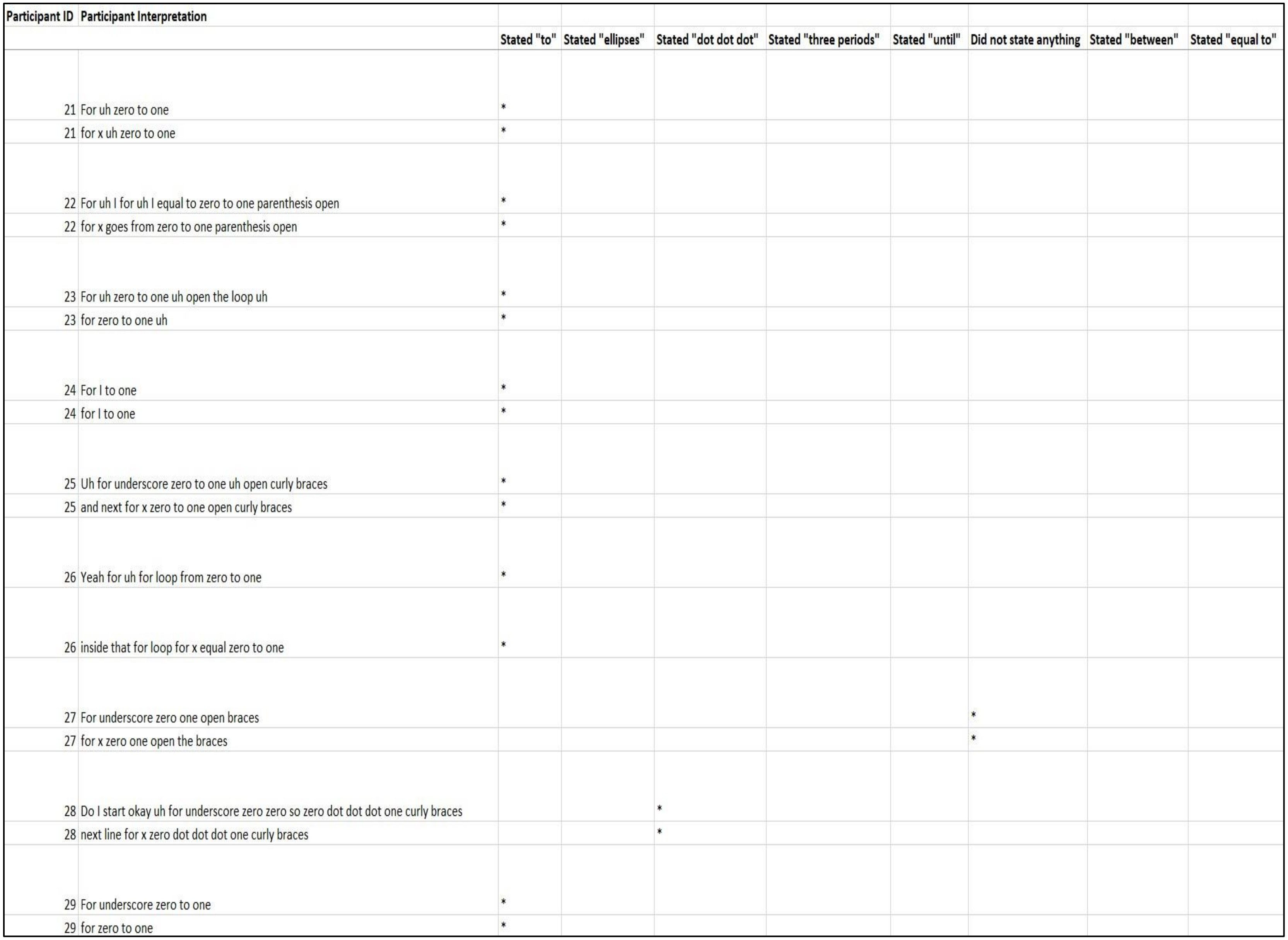
E.5) …



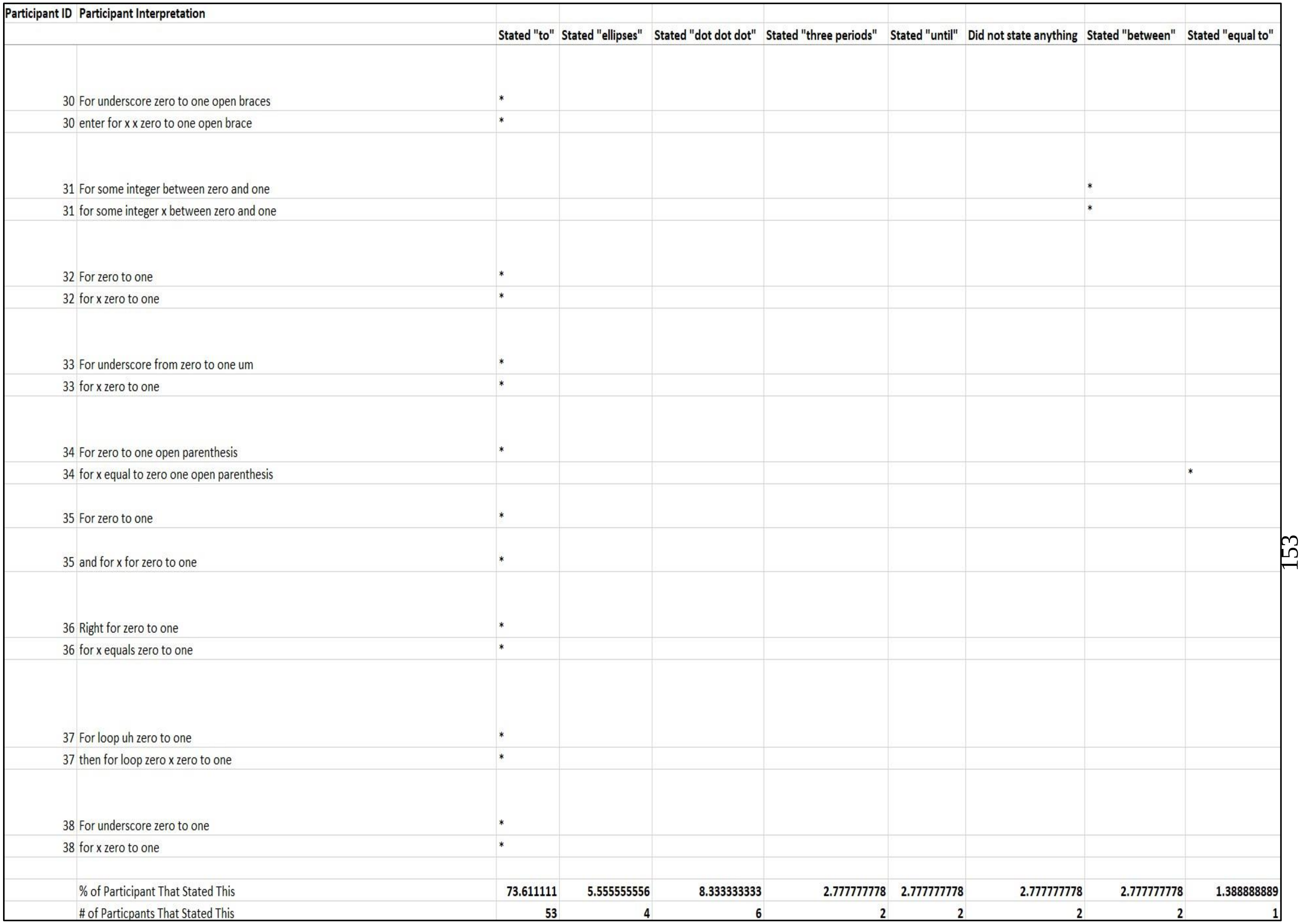
Participant ID	Participant Interpretation								
		Stated "to"	Stated "ellipses"	Stated "dot dot dot"	Stated "three periods"	Stated "until"	Did not state anything	Stated "between"	Stated "equal to"
1	For zero to one open flower bracket	*							
1	for x zero to one open flower bracket	*							
2	For zero to one	*							
2	for x to one or to zero	*							
3	For zero to one	*							
3	third line will be for x zero to one	*							
5	For underscore zero ellipses one open bracket		*						
5	for x zero ellipses one open bracket		*						
6	For underscore O to I flower brackets open	*							
6	for x O to one flower brace	*							
7	For underscore zero dot dot dot one flower bracket open			*					
7	for x zero dot dot dot one flower bracket open			*					
8	For underscore zero to one uh flower braces open	*							
8	move to the next line for uh x zero to one uh braces open	*							
9	For underscore O three periods one open flower brackets				*				
9	next line for space x space O three periods one flower brackets				*				
10	For I zero to one flower brackets open	*							
10	for x zero to one flower brackets open	*							



Participant ID	Participant Interpretation								
		Stated "to"	Stated "ellipses"	Stated "dot dot dot"	Stated "three periods"	Stated "until"	Did not state anything	Stated "between"	Stated "equal to"
11	For underscore zero to one open flower braces	*							
11	for x zero to one open flower brackets	*							
12	For underscore zero until one open curly brace					*			
12	for x in zero until one uh open curly brace					*			
13	For underscore zero ellipsis one open curly brace		*						
13	for x zero ellipsis one open curly brace		*						
14	For wait zero to one open parenthesis	*							
14	for x zero to one open parenthesis	*							
15	For zero to one flower bracket open	*							
15	for x zero to one flower bracket open	*							
16	For underscore zero to one flower brackets	*							
16	for x zero to one flower brackets	*							
18	For underscore zero dot dot dot one open parenthesis			*					
18	for loop x O dot dot dot one open parenthesis			*					
19	For underscore zero to one	*							
19	for x zero to one	*							
20	For zero to one	*							
20	for x zero to one	*							

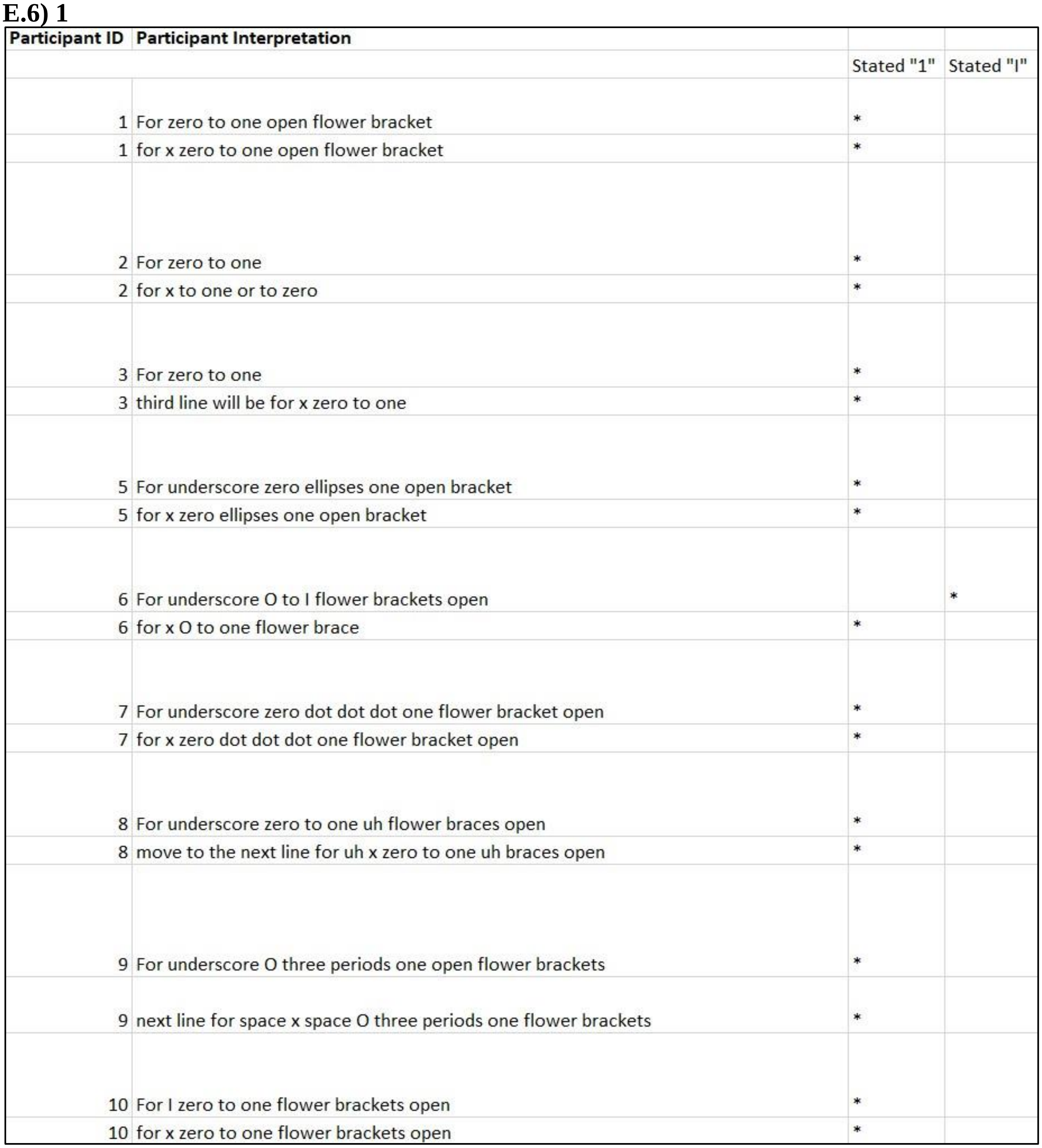


Participant ID	Participant Interpretation								
		Stated "to"	Stated "ellipses"	Stated "dot dot dot"	Stated "three periods"	Stated "until"	Did not state anything	Stated "between"	Stated "equal to"
21	For uh zero to one	*							
21	for x uh zero to one	*							
22	For uh I for uh I equal to zero to one parenthesis open	*							
22	for x goes from zero to one parenthesis open	*							
23	For uh zero to one uh open the loop uh	*							
23	for zero to one uh	*							
24	For I to one	*							
24	for I to one	*							
25	Uh for underscore zero to one uh open curly braces	*							
25	and next for x zero to one open curly braces	*							
26	Yeah for uh for loop from zero to one	*							
26	inside that for loop for x equal zero to one	*							
27	For underscore zero one open braces						*		
27	for x zero one open the braces						*		
28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces			*					
28	next line for x zero dot dot dot one curly braces			*					
29	For underscore zero to one	*							
29	for zero to one	*							

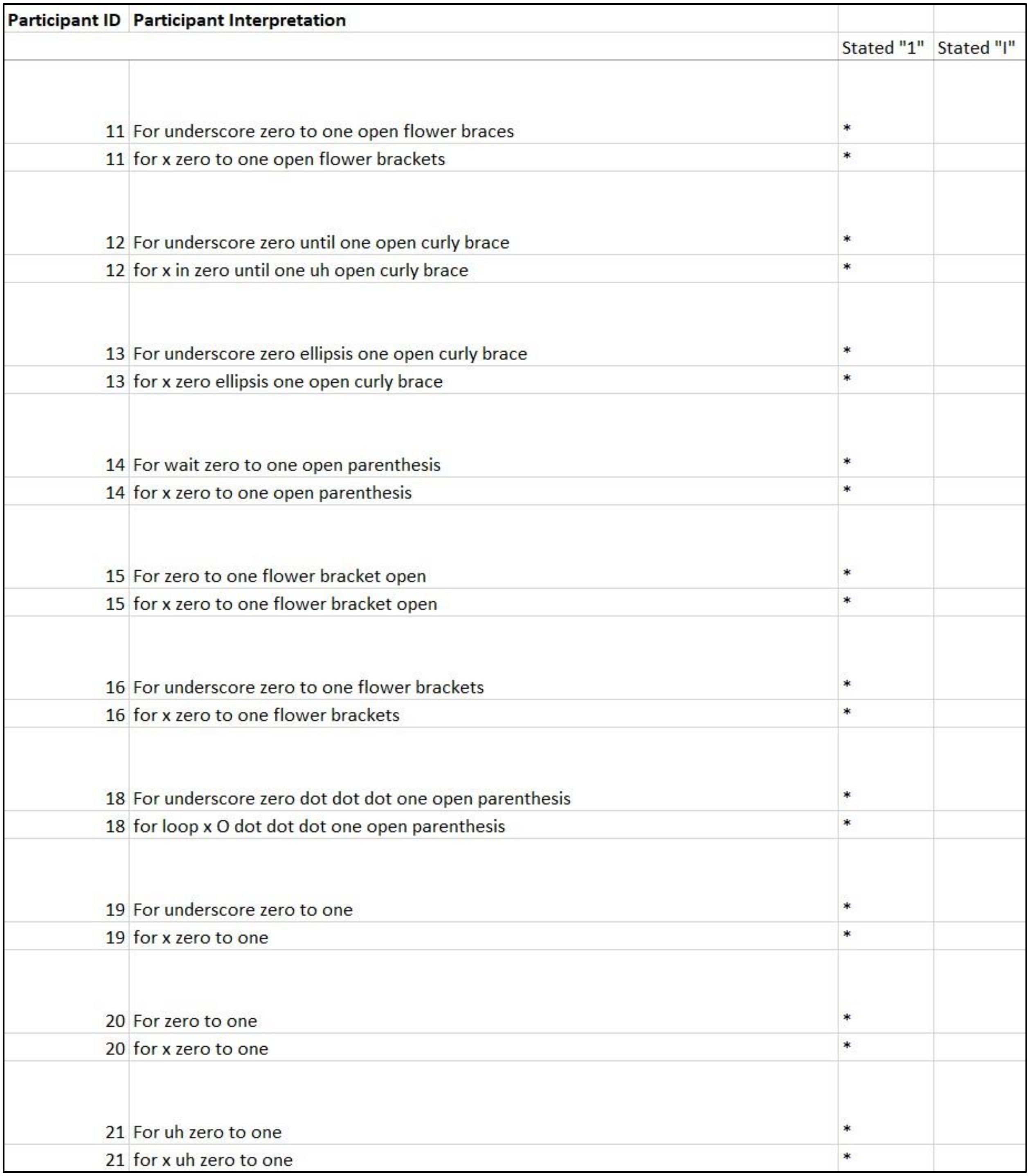


Participant ID	Participant Interpretation								
		Stated "to"	Stated "ellipses"	Stated "dot dot dot"	Stated "three periods"	Stated "until"	Did not state anything	Stated "between"	Stated "equal to"
30	For underscore zero to one open braces	*							
30	enter for x x zero to one open brace	*							
31	For some integer between zero and one							*	
31	for some integer x between zero and one							*	
32	For zero to one	*							
32	for x zero to one	*							
33	For underscore from zero to one um	*							
33	for x zero to one	*							
34	For zero to one open parenthesis	*							
34	for x equal to zero one open parenthesis								*
35	For zero to one	*							
35	and for x for zero to one	*							
36	Right for zero to one	*							
36	for x equals zero to one	*							
37	For loop uh zero to one	*							
37	then for loop zero x zero to one	*							
38	For underscore zero to one	*							
38	for x zero to one	*							
	% of Participant That Stated This	**73.611111**	**5.555555556**	**8.333333333**	**2.777777778**	**2.777777778**	**2.777777778**	**2.777777778**	**1.388888889**
	# of Particpants That Stated This	**53**	**4**	**6**	**2**	**2**	**2**	**2**	**1**

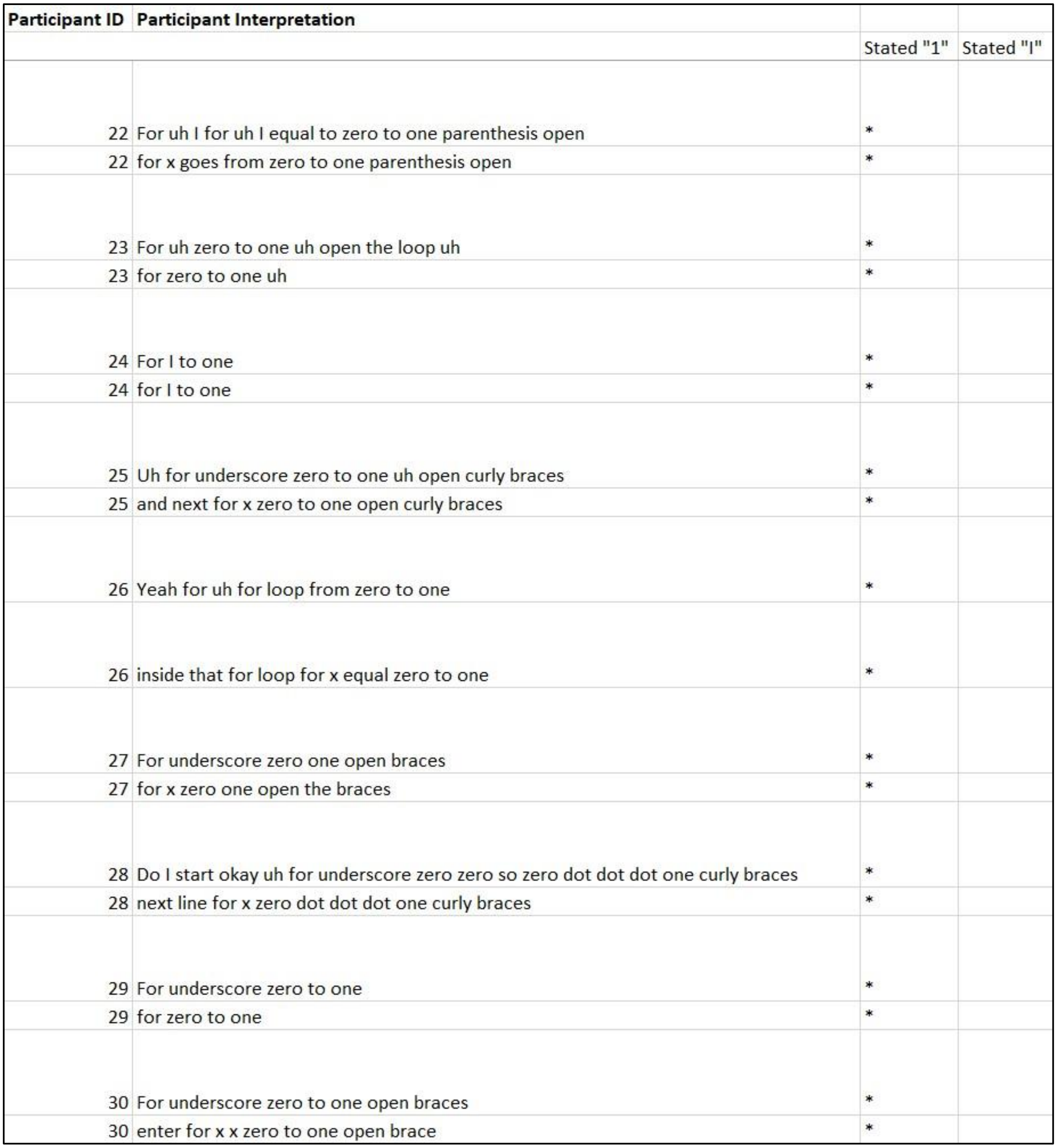
E.6) 1



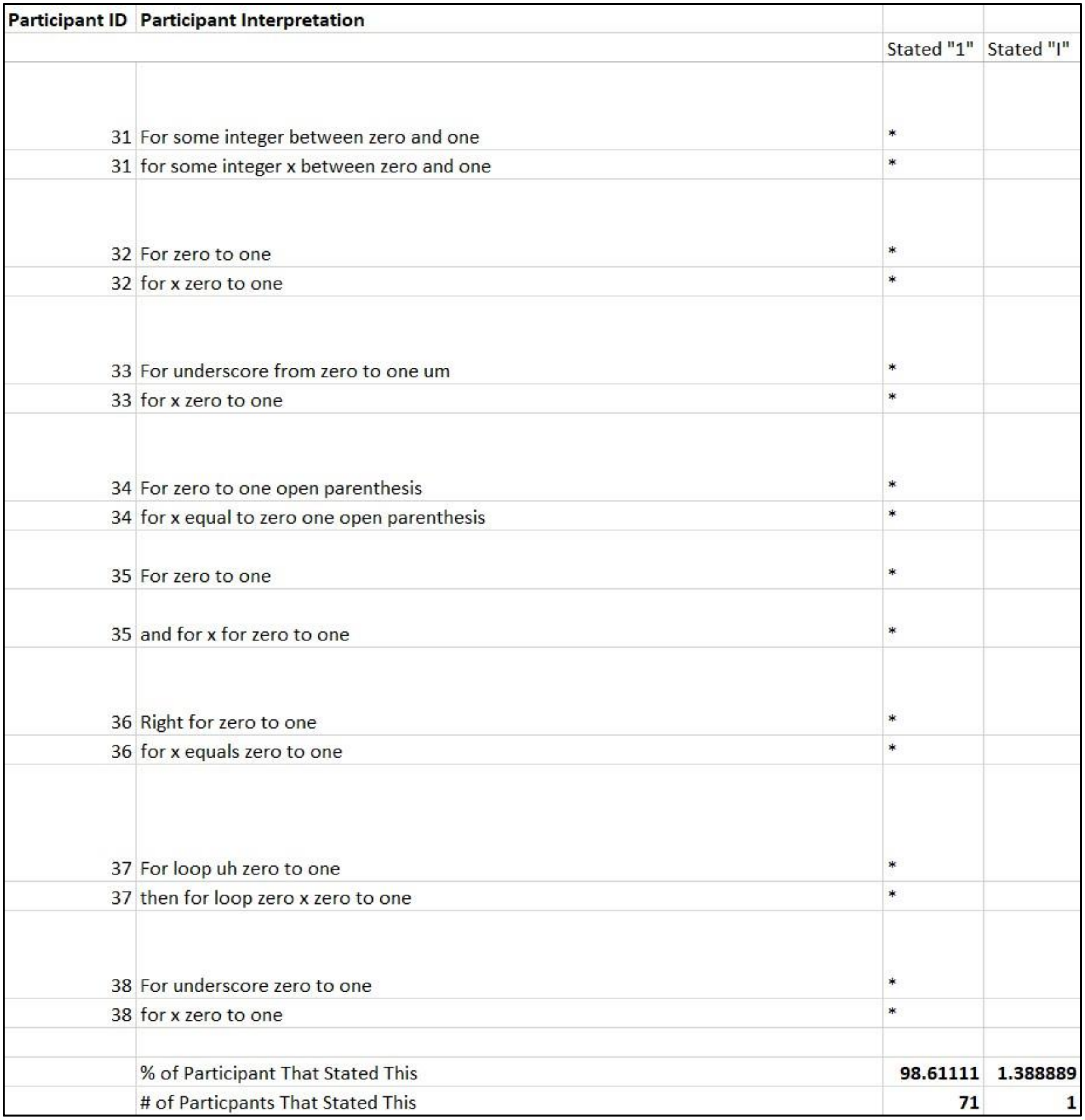
Participant ID	Participant Interpretation		
		Stated "1"	Stated "I"
1	For zero to one open flower bracket	*	
1	for x zero to one open flower bracket	*	
2	For zero to one	*	
2	for x to one or to zero	*	
3	For zero to one	*	
3	third line will be for x zero to one	*	
5	For underscore zero ellipses one open bracket	*	
5	for x zero ellipses one open bracket	*	
6	For underscore O to I flower brackets open		*
6	for x O to one flower brace	*	
7	For underscore zero dot dot dot one flower bracket open	*	
7	for x zero dot dot dot one flower bracket open	*	
8	For underscore zero to one uh flower braces open	*	
8	move to the next line for uh x zero to one uh braces open	*	
9	For underscore O three periods one open flower brackets	*	
9	next line for space x space O three periods one flower brackets	*	
10	For I zero to one flower brackets open	*	
10	for x zero to one flower brackets open	*	



Participant ID	Participant Interpretation		
		Stated "1"	Stated "I"
11	For underscore zero to one open flower braces	*	
11	for x zero to one open flower brackets	*	
12	For underscore zero until one open curly brace	*	
12	for x in zero until one uh open curly brace	*	
13	For underscore zero ellipsis one open curly brace	*	
13	for x zero ellipsis one open curly brace	*	
14	For wait zero to one open parenthesis	*	
14	for x zero to one open parenthesis	*	
15	For zero to one flower bracket open	*	
15	for x zero to one flower bracket open	*	
16	For underscore zero to one flower brackets	*	
16	for x zero to one flower brackets	*	
18	For underscore zero dot dot dot one open parenthesis	*	
18	for loop x O dot dot dot one open parenthesis	*	
19	For underscore zero to one	*	
19	for x zero to one	*	
20	For zero to one	*	
20	for x zero to one	*	
21	For uh zero to one	*	
21	for x uh zero to one	*	

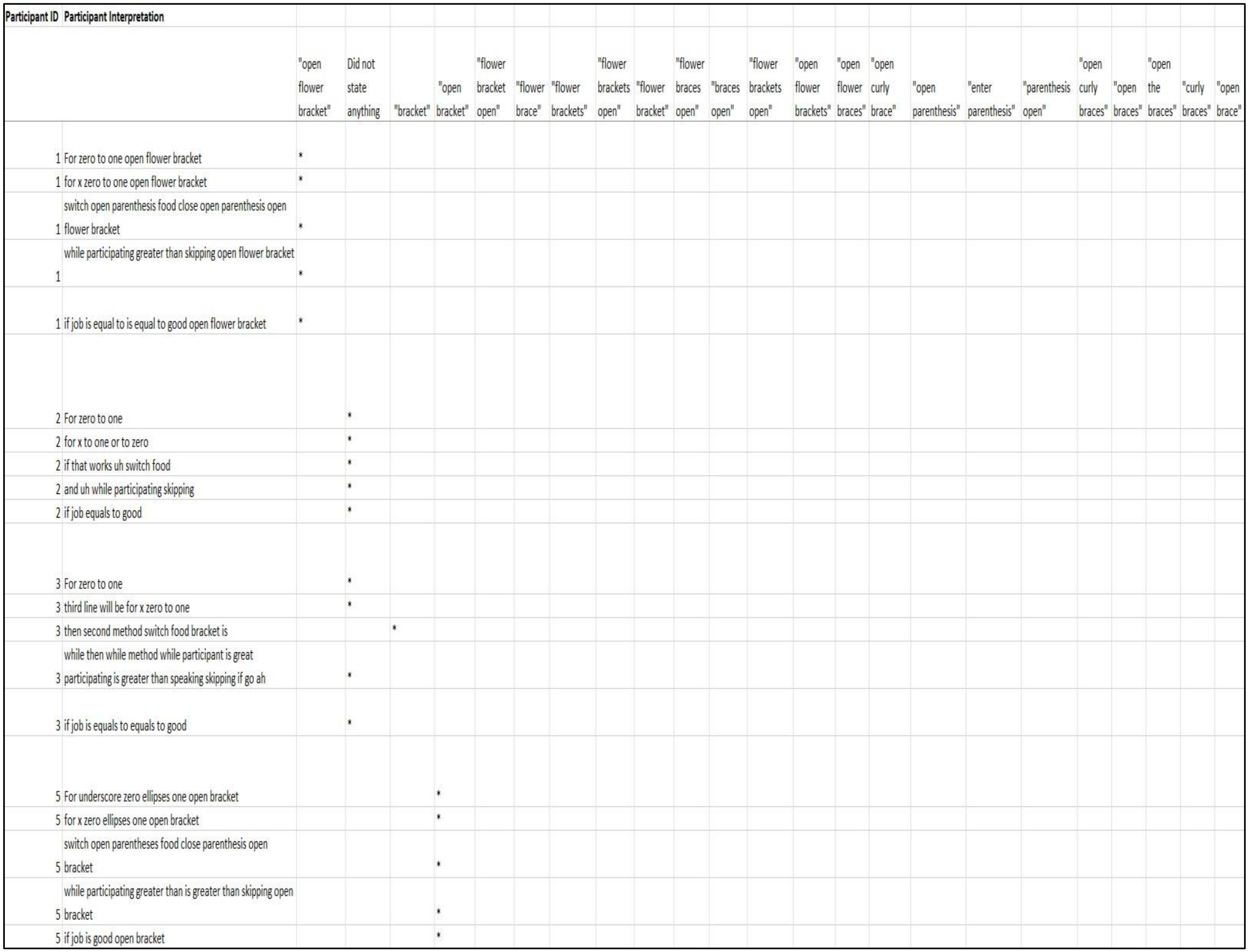


Participant ID	Participant Interpretation		
		Stated "1"	Stated "I"
22	For uh I for uh I equal to zero to one parenthesis open	*	
22	for x goes from zero to one parenthesis open	*	
23	For uh zero to one uh open the loop uh	*	
23	for zero to one uh	*	
24	For I to one	*	
24	for I to one	*	
25	Uh for underscore zero to one uh open curly braces	*	
25	and next for x zero to one open curly braces	*	
26	Yeah for uh for loop from zero to one	*	
26	inside that for loop for x equal zero to one	*	
27	For underscore zero one open braces	*	
27	for x zero one open the braces	*	
28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces	*	
28	next line for x zero dot dot dot one curly braces	*	
29	For underscore zero to one	*	
29	for zero to one	*	
30	For underscore zero to one open braces	*	
30	enter for x x zero to one open brace	*	

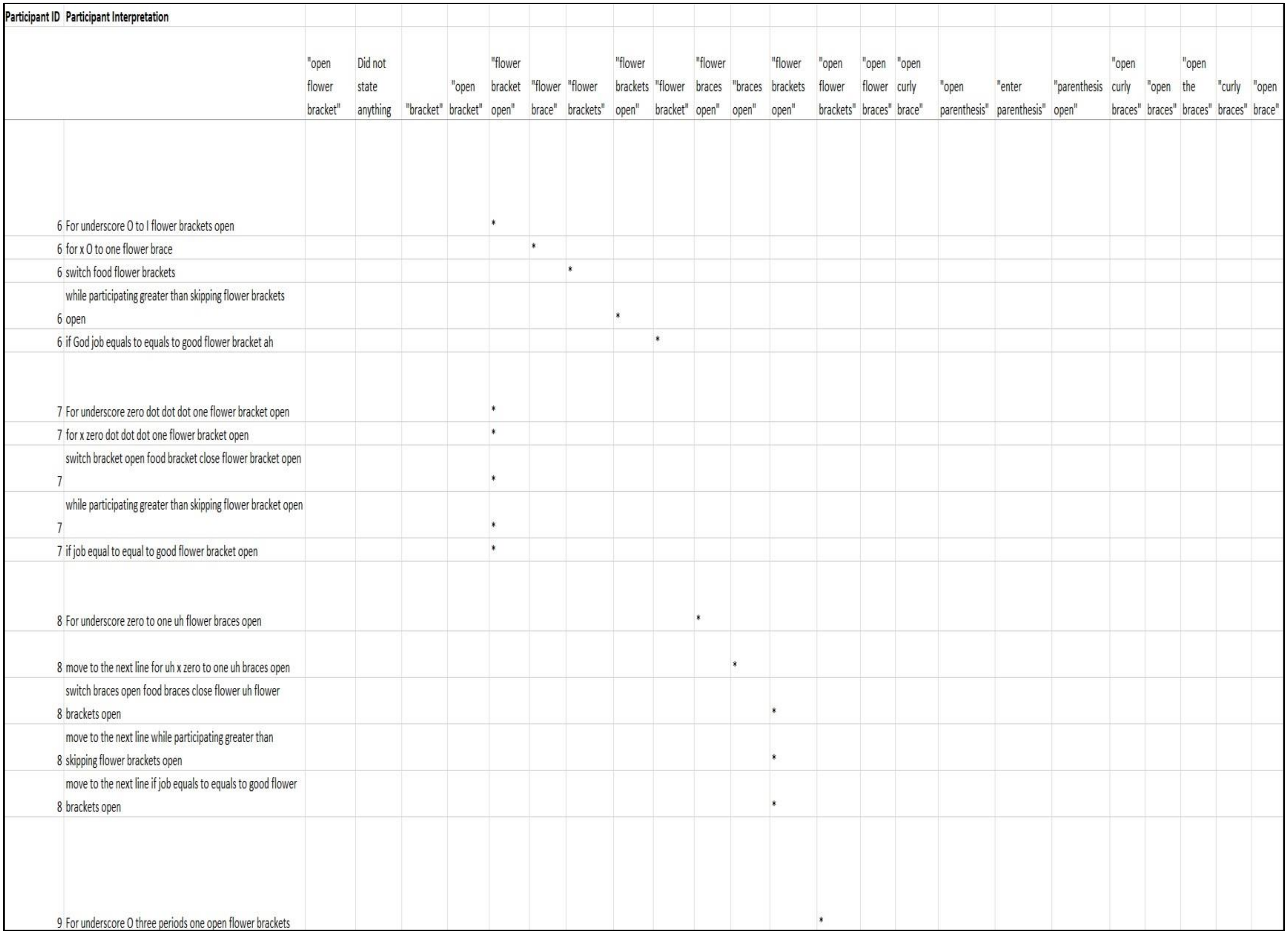


Participant ID	Participant Interpretation		
		Stated "1"	Stated "I"
31	For some integer between zero and one	*	
31	for some integer x between zero and one	*	
32	For zero to one	*	
32	for x zero to one	*	
33	For underscore from zero to one um	*	
33	for x zero to one	*	
34	For zero to one open parenthesis	*	
34	for x equal to zero one open parenthesis	*	
35	For zero to one	*	
35	and for x for zero to one	*	
36	Right for zero to one	*	
36	for x equals zero to one	*	
37	For loop uh zero to one	*	
37	then for loop zero x zero to one	*	
38	For underscore zero to one	*	
38	for x zero to one	*	
	% of Participant That Stated This	**98.61111**	**1.388889**
	# of Particpants That Stated This	**71**	**1**

E.7) {

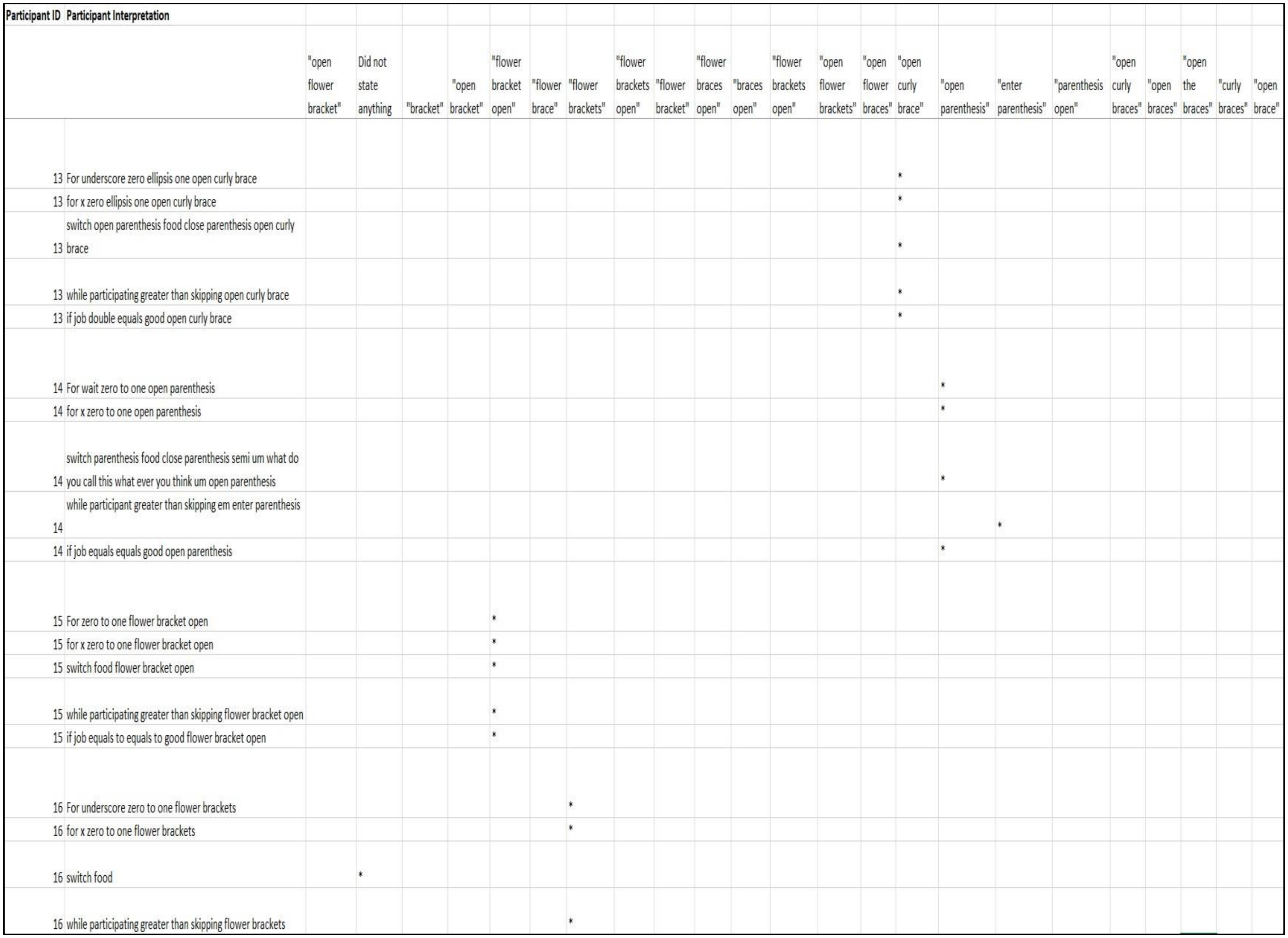


Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
1	For zero to one open flower bracket	*																						
1	for x zero to one open flower bracket	*																						
1	switch open parenthesis food close open parenthesis open flower bracket	*																						
1	while participating greater than skipping open flower bracket	*																						
1	if job is equal to is equal to good open flower bracket	*																						
2	For zero to one		*																					
2	for x to one or to zero		*																					
2	if that works uh switch food		*																					
2	and uh while participating skipping		*																					
2	if job equals to good		*																					
3	For zero to one		*																					
3	third line will be for x zero to one		*																					
3	then second method switch food bracket is			*																				
3	while then while method while participant is great participating is greater than speaking skipping if go ah		*																					
3	if job is equals to equals to good		*																					
5	For underscore zero ellipses one open bracket				*																			
5	for x zero ellipses one open bracket				*																			
5	switch open parentheses food close parenthesis open bracket				*																			
5	while participating greater than is greater than skipping open bracket				*																			
5	if job is good open bracket				*																			

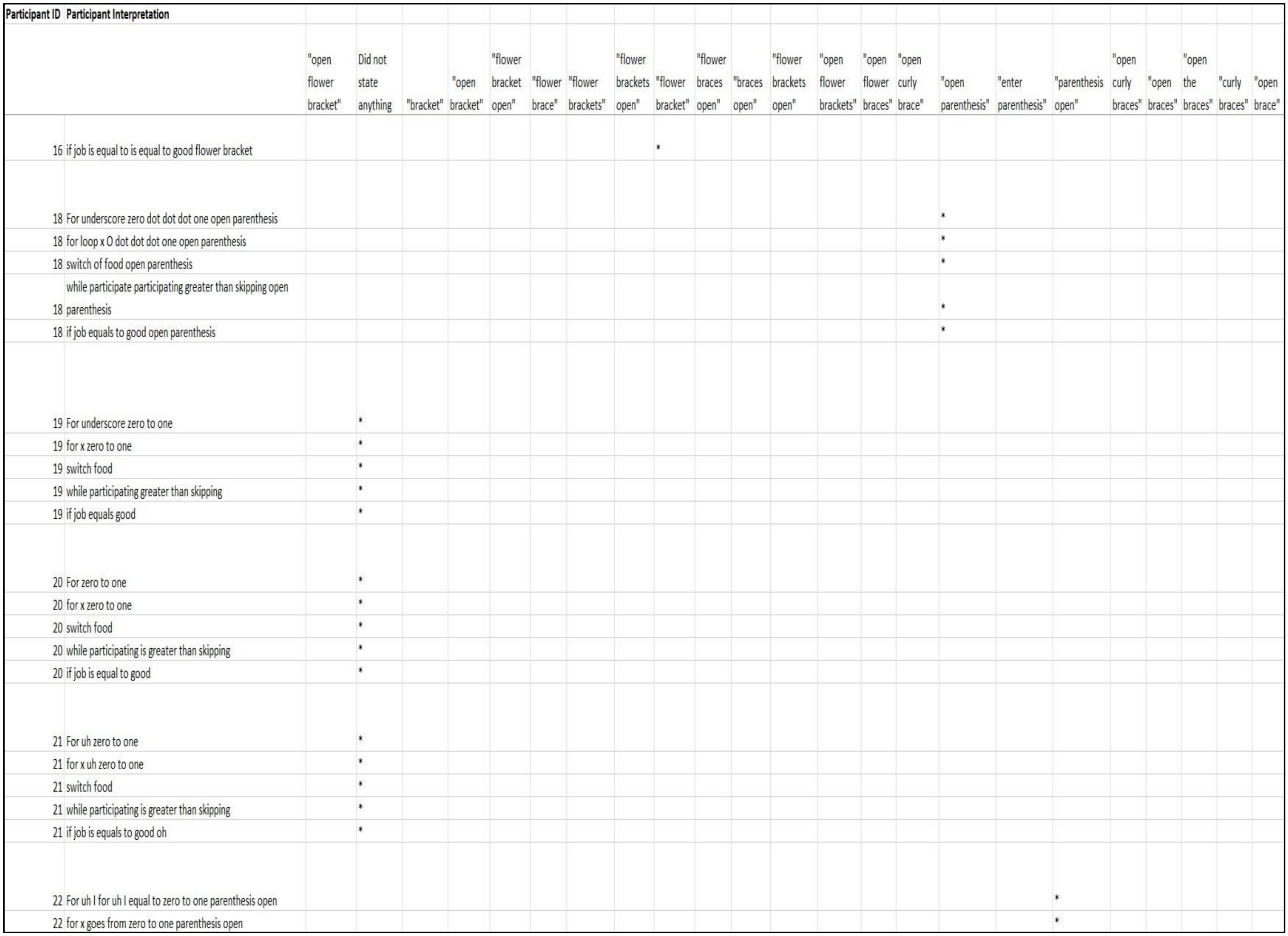


Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
6	For underscore O to I flower brackets open					*																		
6	for x O to one flower brace						*																	
6	switch food flower brackets							*																
6	while participating greater than skipping flower brackets open								*															
6	if God job equals to equals to good flower bracket ah									*														
7	For underscore zero dot dot dot one flower bracket open					*																		
7	for x zero dot dot dot one flower bracket open					*																		
7	switch bracket open food bracket close flower bracket open					*																		
7	while participating greater than skipping flower bracket open					*																		
7	if job equal to equal to good flower bracket open					*																		
8	For underscore zero to one uh flower braces open										*													
8	move to the next line for uh x zero to one uh braces open											*												
8	switch braces open food braces close flower uh flower brackets open												*											
8	move to the next line while participating greater than skipping flower brackets open												*											
8	move to the next line if job equals to equals to good flower brackets open												*											
9	For underscore O three periods one open flower brackets													*										

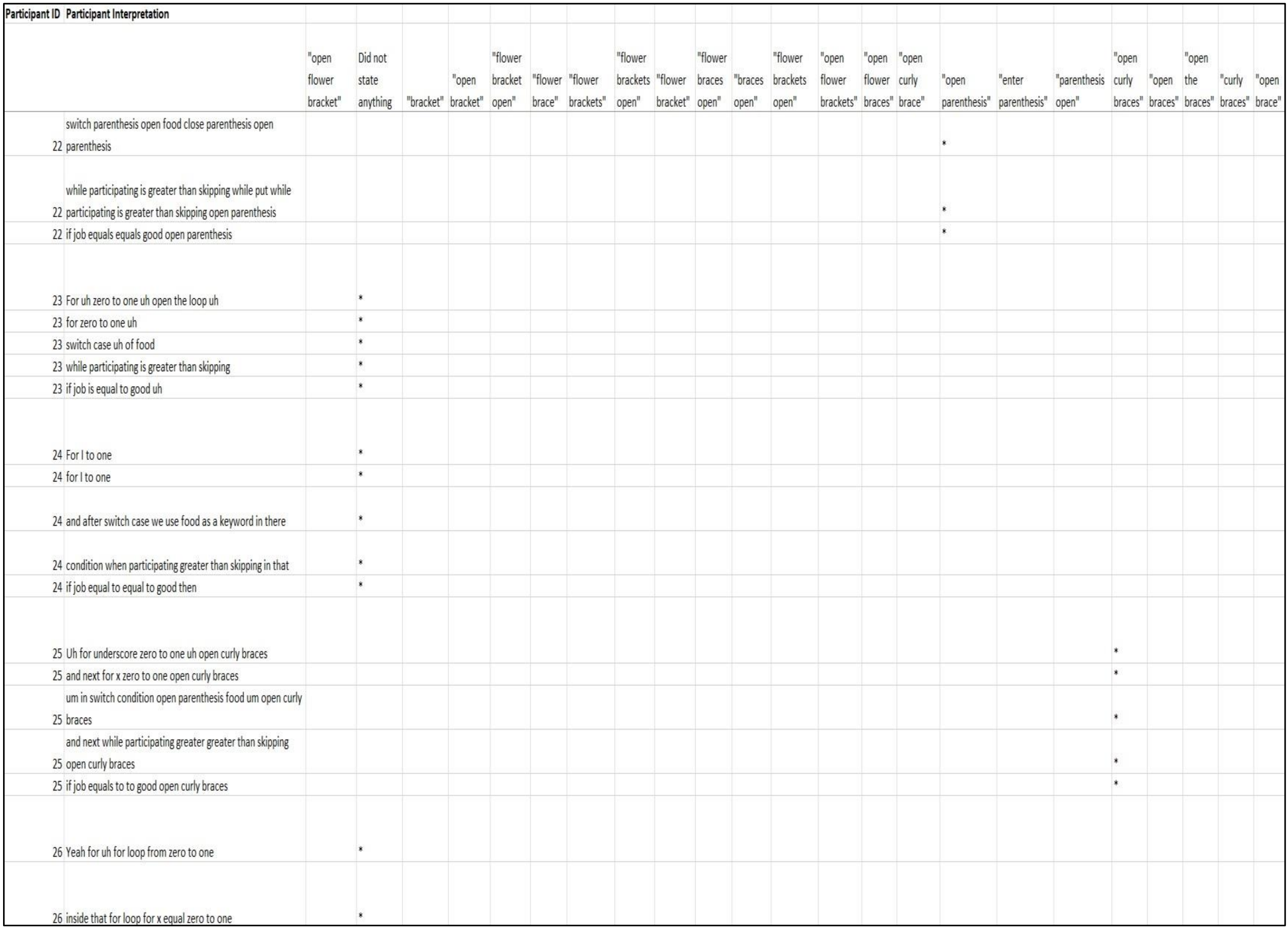
Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
9	next line for space x space O three periods one flower brackets							*																
9	next line switch open parenthesis food close parenthesis flower bracket									*														
9	next line while space participating is greater than skipping open flower brackets													*										
9	next line if space job is equals to is equals to good flower brackets open								*															
10	For I zero to one flower brackets open								*															
10	for x zero to one flower brackets open								*															
10	switch uh food flower brack flower brackets open								*															
10	while participating greater skipping flower brackets open								*															
10	if job is equal to is equal to good flower brackets open								*															
11	For underscore zero to one open flower braces														*									
11	for x zero to one open flower brackets													*										
11	the next line is switch of food switch open parenthesis food close parenthesis open flower brackets													*										
11	while participating is greater than skipping open flower brackets													*										
11	if job double equal to good open flower brackets													*										
12	For underscore zero until one open curly brace															*								
12	for x in zero until one uh open curly brace															*								
12	switch open parenthesis food close parenthesis open curly brace															*								
12	while participating greater than skipping open curly brace															*								
12	if job equals equals good open curly brace															*								



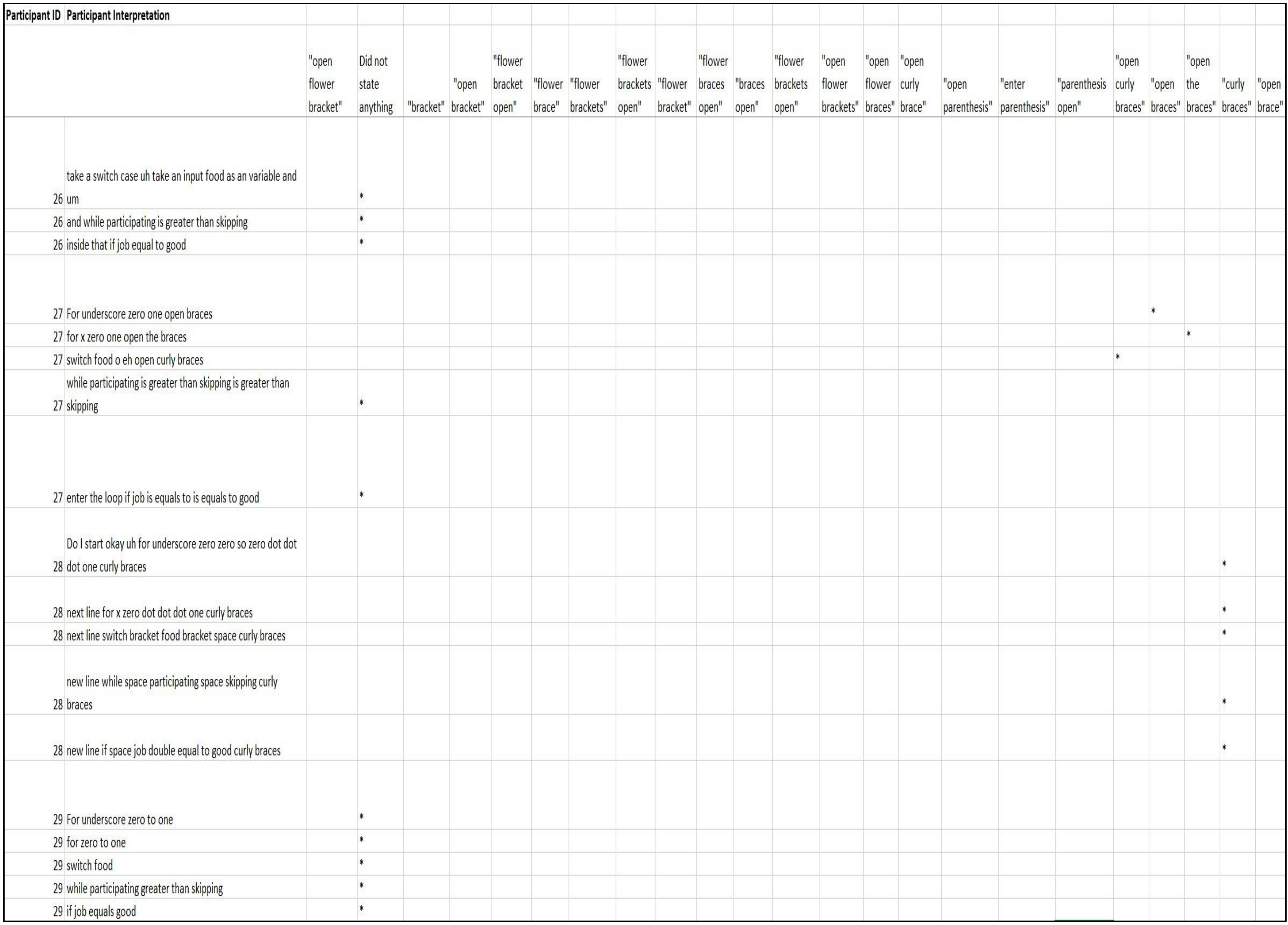
Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
13	For underscore zero ellipsis one open curly brace															*								
13	for x zero ellipsis one open curly brace															*								
13	switch open parenthesis food close parenthesis open curly brace															*								
13	while participating greater than skipping open curly brace															*								
13	if job double equals good open curly brace															*								
14	For wait zero to one open parenthesis																*							
14	for x zero to one open parenthesis																*							
14	switch parenthesis food close parenthesis semi um what do you call this what ever you think um open parenthesis																*							
14	while participant greater than skipping em enter parenthesis																	*						
14	if job equals equals good open parenthesis																*							
15	For zero to one flower bracket open					*																		
15	for x zero to one flower bracket open					*																		
15	switch food flower bracket open					*																		
15	while participating greater than skipping flower bracket open					*																		
15	if job equals to equals to good flower bracket open					*																		
16	For underscore zero to one flower brackets							*																
16	for x zero to one flower brackets							*																
16	switch food		*																					
16	while participating greater than skipping flower brackets							*																



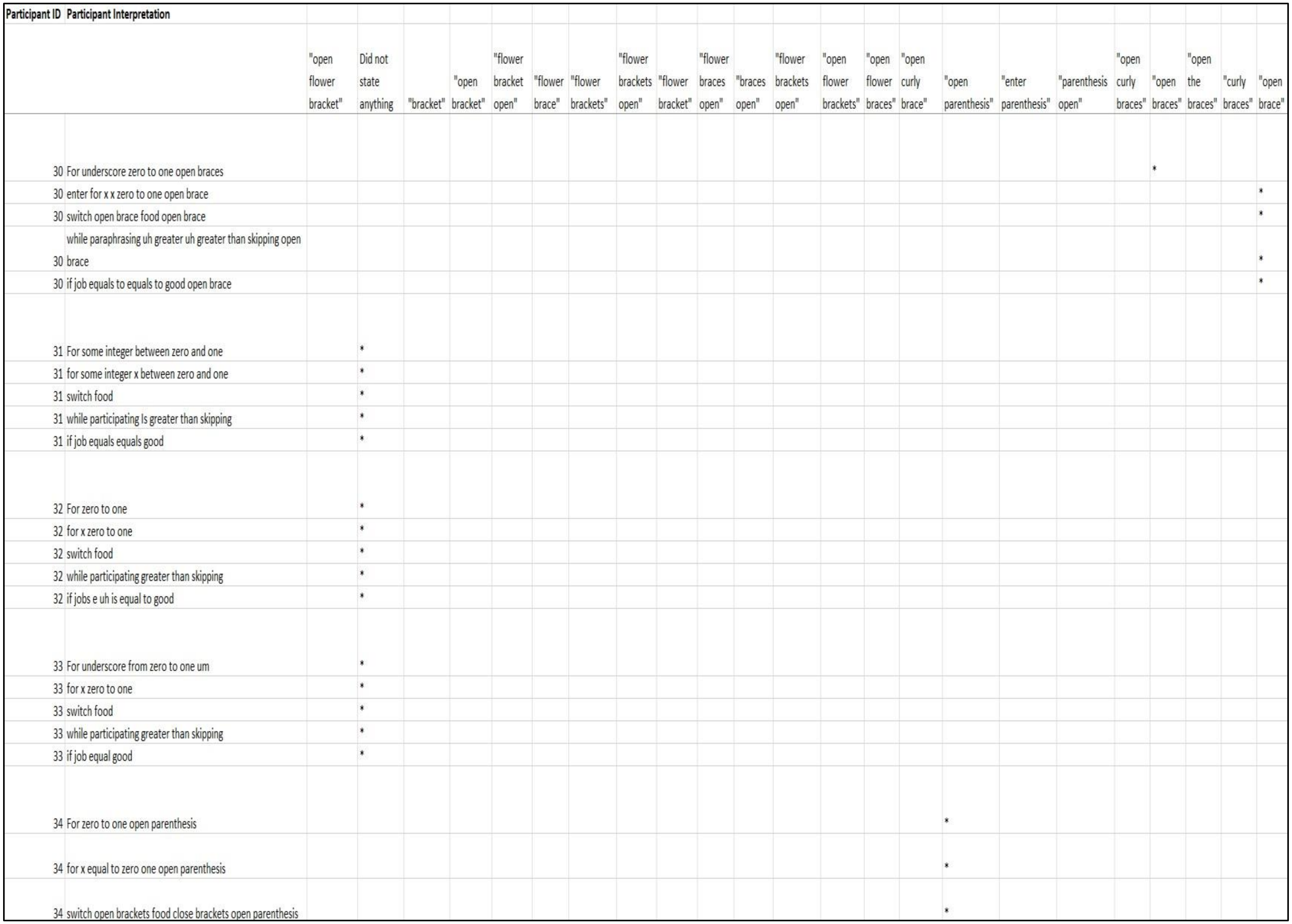
Participant ID	Participant Interpretation																							
		"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
16	if job is equal to is equal to good flower bracket									*														
18	For underscore zero dot dot one open parenthesis																*							
18	for loop x O dot dot one open parenthesis																*							
18	switch of food open parenthesis																*							
18	while participate participating greater than skipping open parenthesis																*							
18	if job equals to good open parenthesis																*							
19	For underscore zero to one		*																					
19	for x zero to one		*																					
19	switch food		*																					
19	while participating greater than skipping		*																					
19	if job equals good		*																					
20	For zero to one		*																					
20	for x zero to one		*																					
20	switch food		*																					
20	while participating is greater than skipping		*																					
20	if job is equal to good		*																					
21	For uh zero to one		*																					
21	for x uh zero to one		*																					
21	switch food		*																					
21	while participating is greater than skipping		*																					
21	if job is equals to good oh		*																					
22	For uh I for uh I equal to zero to one parenthesis open																		*					
22	for x goes from zero to one parenthesis open																		*					



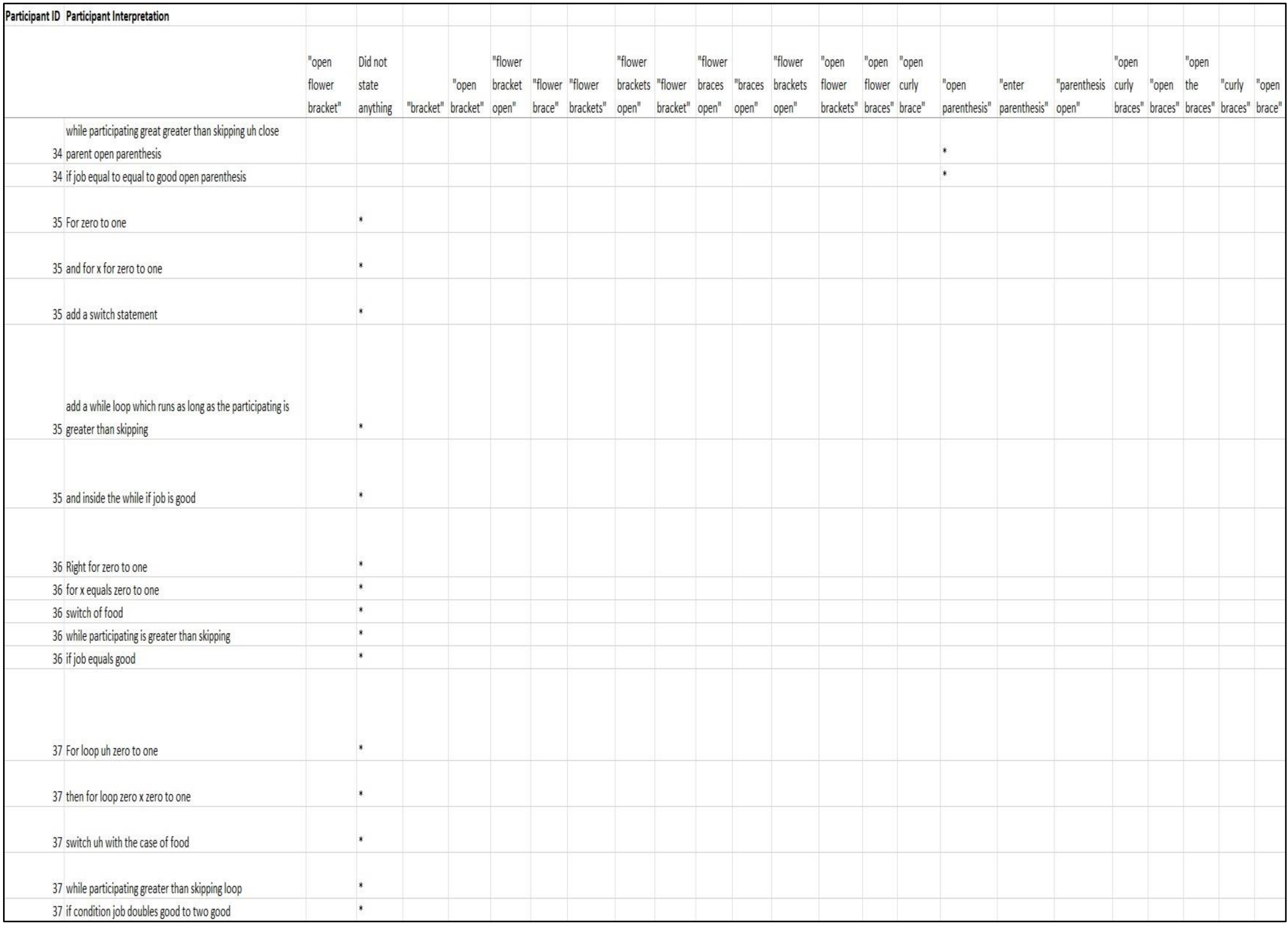
Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
22	switch parenthesis open food close parenthesis open parenthesis																*							
22	while participating is greater than skipping while put while participating is greater than skipping open parenthesis																*							
22	if job equals equals good open parenthesis																*							
23	For uh zero to one uh open the loop uh		*																					
23	for zero to one uh		*																					
23	switch case uh of food		*																					
23	while participating is greater than skipping		*																					
23	if job is equal to good uh		*																					
24	For I to one		*																					
24	for I to one		*																					
24	and after switch case we use food as a keyword in there		*																					
24	condition when participating greater than skipping in that		*																					
24	if job equal to equal to good then		*																					
25	Uh for underscore zero to one uh open curly braces																			*				
25	and next for x zero to one open curly braces																			*				
25	um in switch condition open parenthesis food um open curly braces																			*				
25	and next while participating greater greater than skipping open curly braces																			*				
25	if job equals to to good open curly braces																			*				
26	Yeah for uh for loop from zero to one		*																					
26	inside that for loop for x equal zero to one		*																					



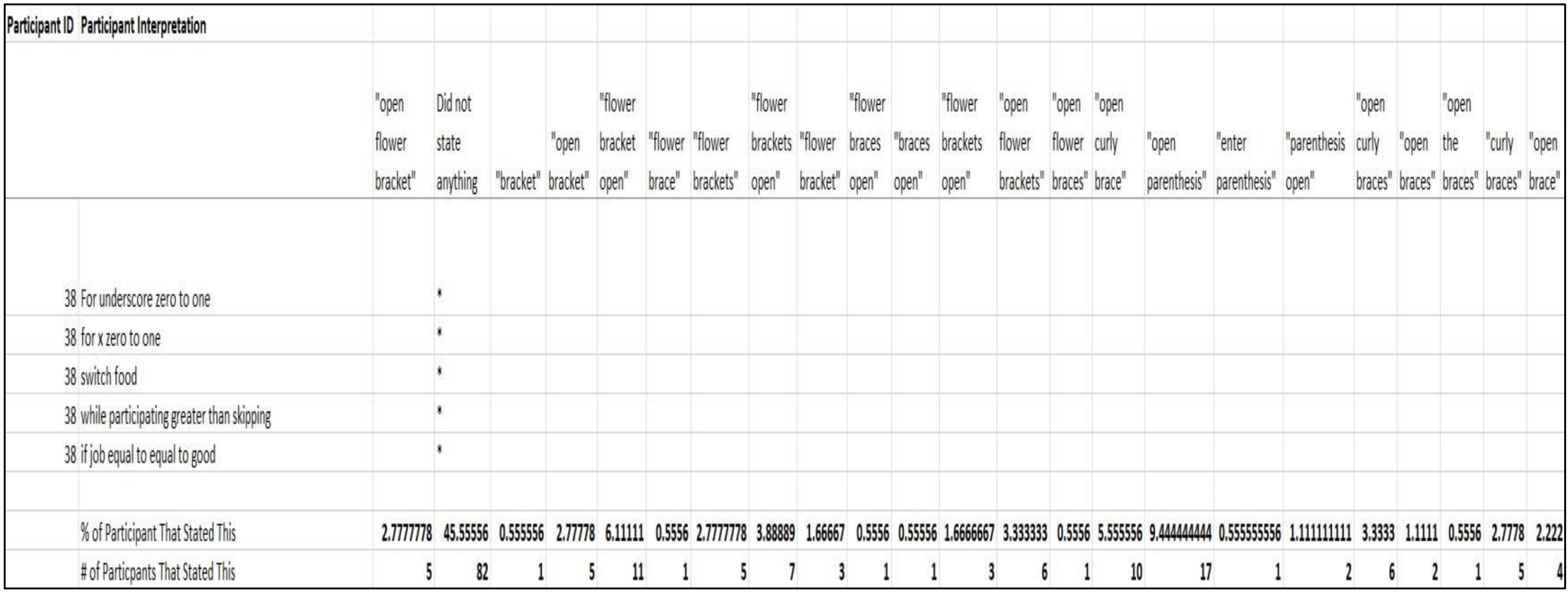
Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
26	take a switch case uh take an input food as an variable and um		*																					
26	and while participating is greater than skipping		*																					
26	inside that if job equal to good		*																					
27	For underscore zero one open braces																				*			
27	for x zero one open the braces																					*		
27	switch food o eh open curly braces																			*				
27	while participating is greater than skipping is greater than skipping		*																					
27	enter the loop if job is equals to is equals to good		*																					
28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces																						*	
28	next line for x zero dot dot dot one curly braces																						*	
28	next line switch bracket food bracket space curly braces																						*	
28	new line while space participating space skipping curly braces																						*	
28	new line if space job double equal to good curly braces																						*	
29	For underscore zero to one		*																					
29	for zero to one		*																					
29	switch food		*																					
29	while participating greater than skipping		*																					
29	if job equals good		*																					



Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
30	For underscore zero to one open braces																				*			
30	enter for x x zero to one open brace																							*
30	switch open brace food open brace																							*
30	while paraphrasing uh greater uh greater than skipping open brace																							*
30	if job equals to equals to good open brace																							*
31	For some integer between zero and one		*																					
31	for some integer x between zero and one		*																					
31	switch food		*																					
31	while participating Is greater than skipping		*																					
31	if job equals equals good		*																					
32	For zero to one		*																					
32	for x zero to one		*																					
32	switch food		*																					
32	while participating greater than skipping		*																					
32	if jobs e uh is equal to good		*																					
33	For underscore from zero to one um		*																					
33	for x zero to one		*																					
33	switch food		*																					
33	while participating greater than skipping		*																					
33	if job equal good		*																					
34	For zero to one open parenthesis																*							
34	for x equal to zero one open parenthesis																*							
34	switch open brackets food close brackets open parenthesis																*							

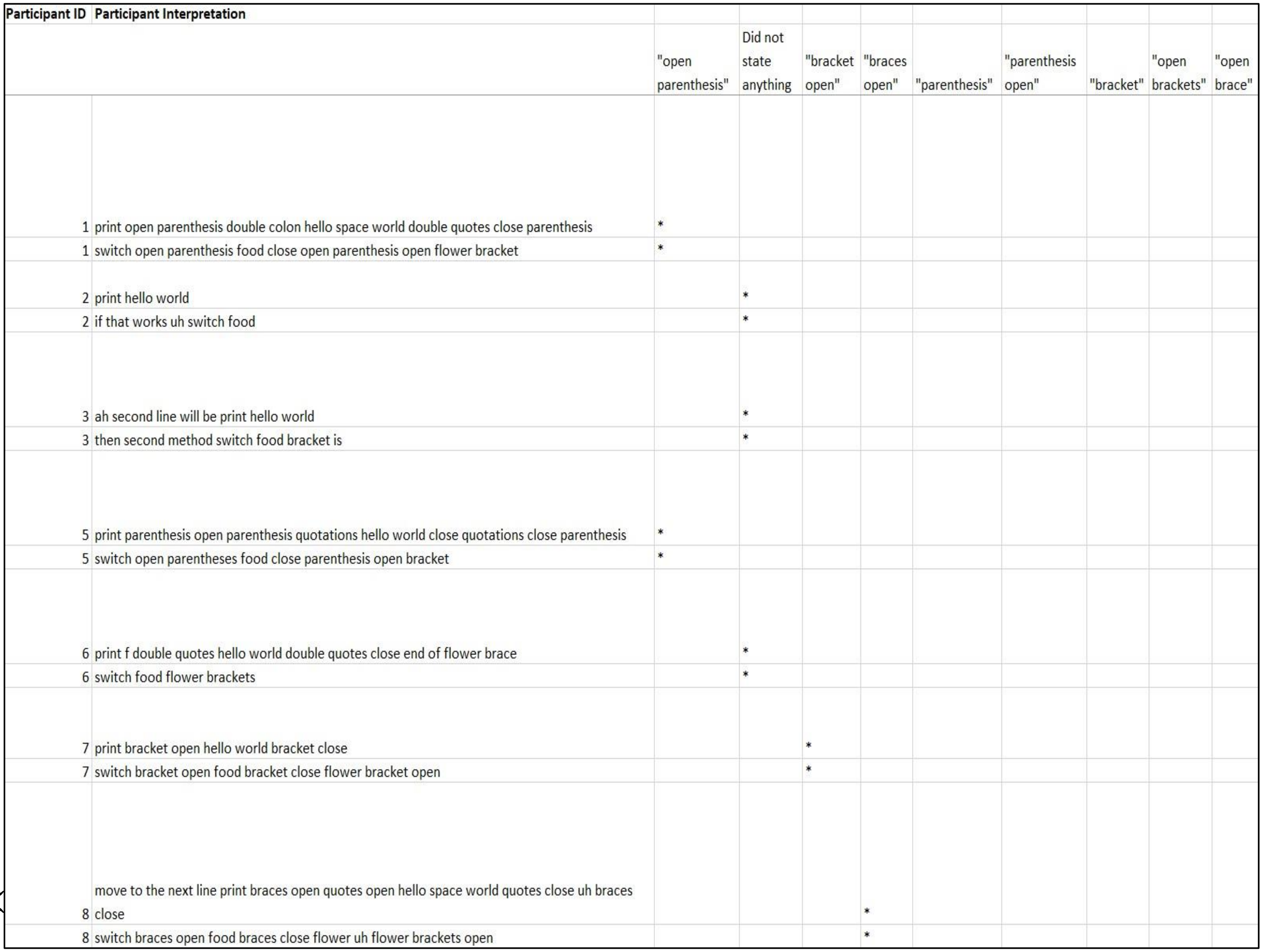


Participant ID	Participant Interpretation	"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
34	while participating great greater than skipping uh close parent open parenthesis																*							
34	if job equal to equal to good open parenthesis																*							
35	For zero to one		*																					
35	and for x for zero to one		*																					
35	add a switch statement		*																					
35	add a while loop which runs as long as the participating is greater than skipping		*																					
35	and inside the while if job is good		*																					
36	Right for zero to one		*																					
36	for x equals zero to one		*																					
36	switch of food		*																					
36	while participating is greater than skipping		*																					
36	if job equals good		*																					
37	For loop uh zero to one		*																					
37	then for loop zero x zero to one		*																					
37	switch uh with the case of food		*																					
37	while participating greater than skipping loop		*																					
37	if condition job doubles good to two good		*																					

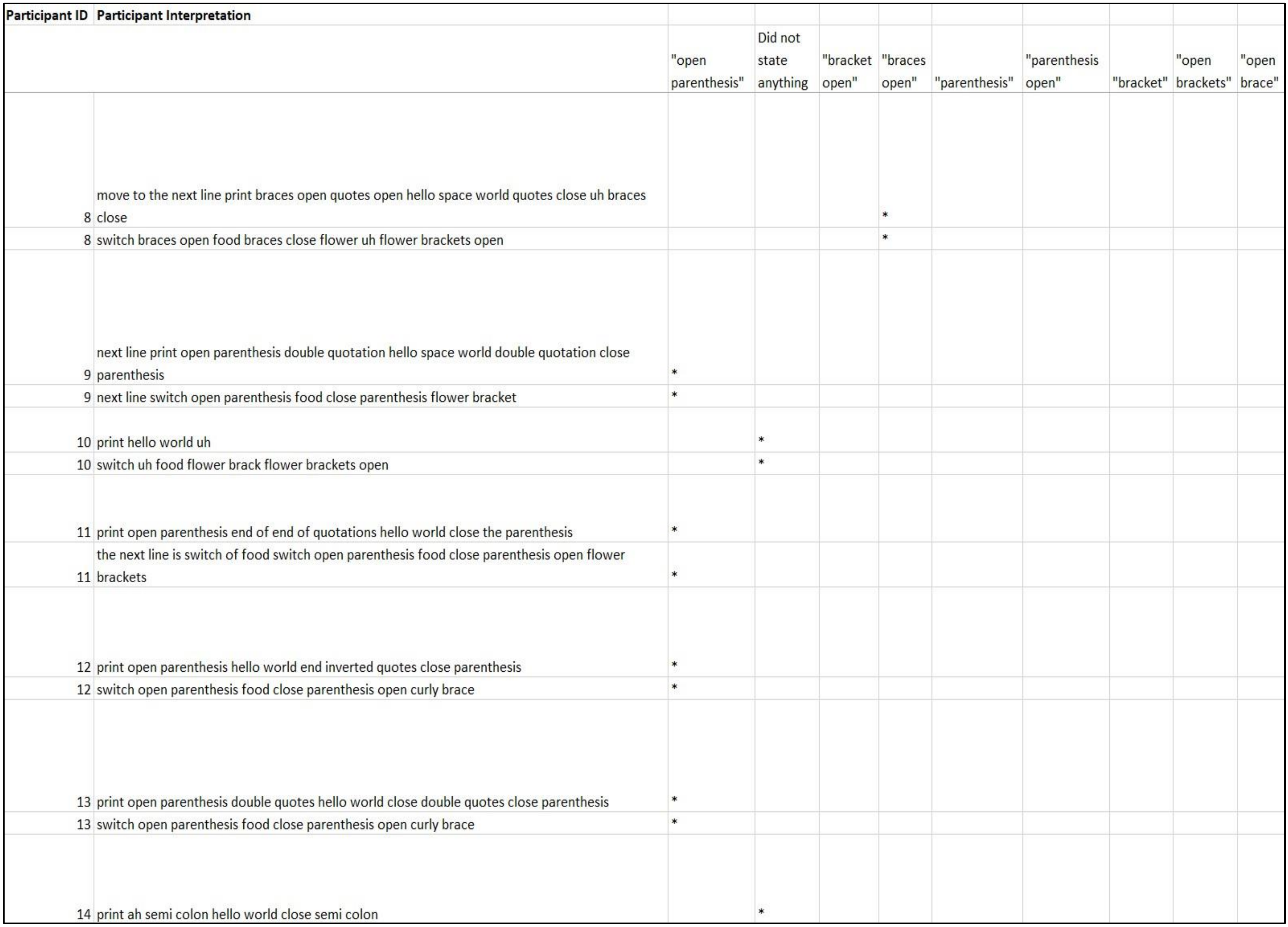


Participant ID	Participant Interpretation																							
		"open flower bracket"	Did not state anything	"bracket"	"open bracket"	"flower bracket open"	"flower brace"	"flower brackets"	"flower brackets open"	"flower bracket"	"flower braces open"	"braces open"	"flower brackets open"	"open flower brackets"	"open flower braces"	"open curly brace"	"open parenthesis"	"enter parenthesis"	"parenthesis open"	"open curly braces"	"open braces"	"open the braces"	"curly braces"	"open brace"
38	For underscore zero to one		*																					
38	for x zero to one		*																					
38	switch food		*																					
38	while participating greater than skipping		*																					
38	if job equal to equal to good		*																					
	% of Participant That Stated This	**2.7777778**	**45.55556**	**0.555556**	**2.77778**	**6.11111**	**0.5556**	**2.7777778**	**3.88889**	**1.66667**	**0.5556**	**0.55556**	**1.6666667**	**3.333333**	**0.5556**	**5.555556**	**9.444444444**	**0.555555556**	**1.111111111**	**3.3333**	**1.1111**	**0.5556**	**2.7778**	**2.222**
	# of Particpants That Stated This	**5**	**82**	**1**	**5**	**11**	**1**	**5**	**7**	**3**	**1**	**1**	**3**	**6**	**1**	**10**	**17**	**1**	**2**	**6**	**2**	**1**	**5**	**4**

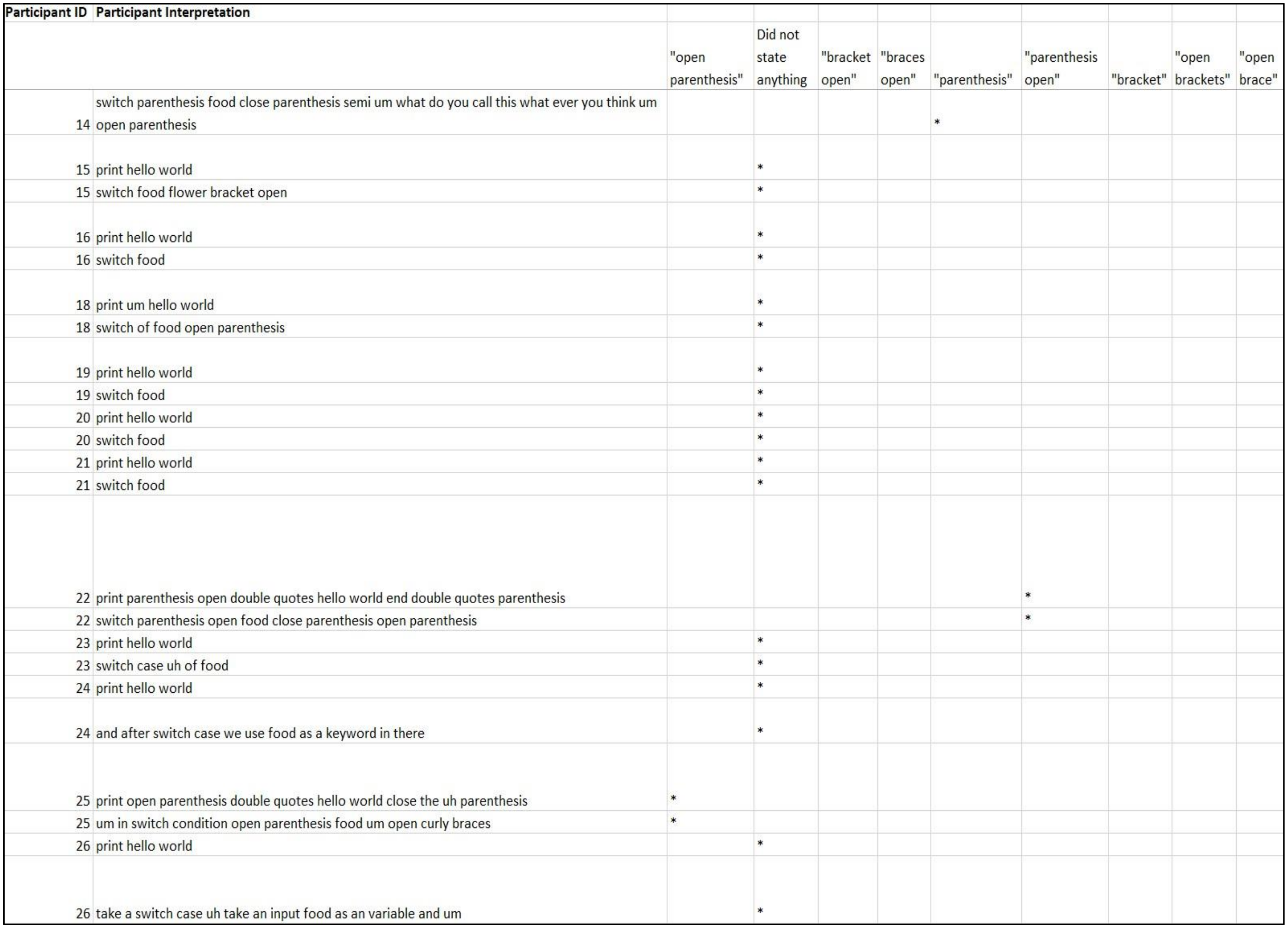
E.8) (



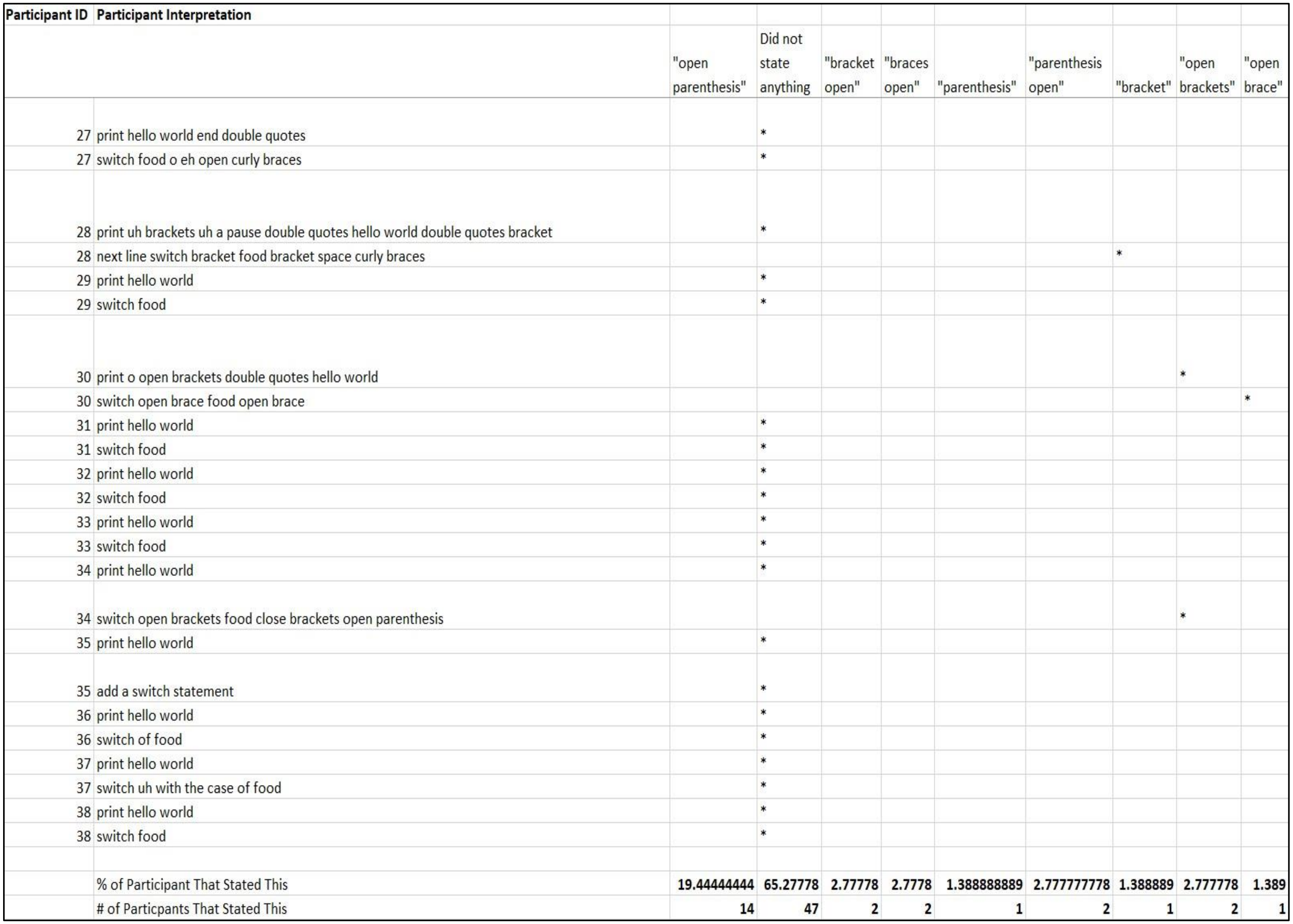
Participant ID	Participant Interpretation									
		"open parenthesis"	Did not state anything	"bracket open"	"braces open"	"parenthesis"	"parenthesis open"	"bracket"	"open brackets"	"open brace"
1	print open parenthesis double colon hello space world double quotes close parenthesis	*								
1	switch open parenthesis food close open parenthesis open flower bracket	*								
2	print hello world		*							
2	if that works uh switch food		*							
3	ah second line will be print hello world		*							
3	then second method switch food bracket is		*							
5	print parenthesis open parenthesis quotations hello world close quotations close parenthesis	*								
5	switch open parentheses food close parenthesis open bracket	*								
6	print f double quotes hello world double quotes close end of flower brace		*							
6	switch food flower brackets		*							
7	print bracket open hello world bracket close			*						
7	switch bracket open food bracket close flower bracket open			*						
8	move to the next line print braces open quotes open hello space world quotes close uh braces close				*					
8	switch braces open food braces close flower uh flower brackets open				*					



Participant ID	Participant Interpretation									
		"open parenthesis"	Did not state anything	"bracket open"	"braces open"	"parenthesis"	"parenthesis open"	"bracket"	"open brackets"	"open brace"
8	move to the next line print braces open quotes open hello space world quotes close uh braces close				*					
8	switch braces open food braces close flower uh flower brackets open				*					
9	next line print open parenthesis double quotation hello space world double quotation close parenthesis	*								
9	next line switch open parenthesis food close parenthesis flower bracket	*								
10	print hello world uh		*							
10	switch uh food flower brack flower brackets open		*							
11	print open parenthesis end of end of quotations hello world close the parenthesis	*								
11	the next line is switch of food switch open parenthesis food close parenthesis open flower brackets	*								
12	print open parenthesis hello world end inverted quotes close parenthesis	*								
12	switch open parenthesis food close parenthesis open curly brace	*								
13	print open parenthesis double quotes hello world close double quotes close parenthesis	*								
13	switch open parenthesis food close parenthesis open curly brace	*								
14	print ah semi colon hello world close semi colon		*							

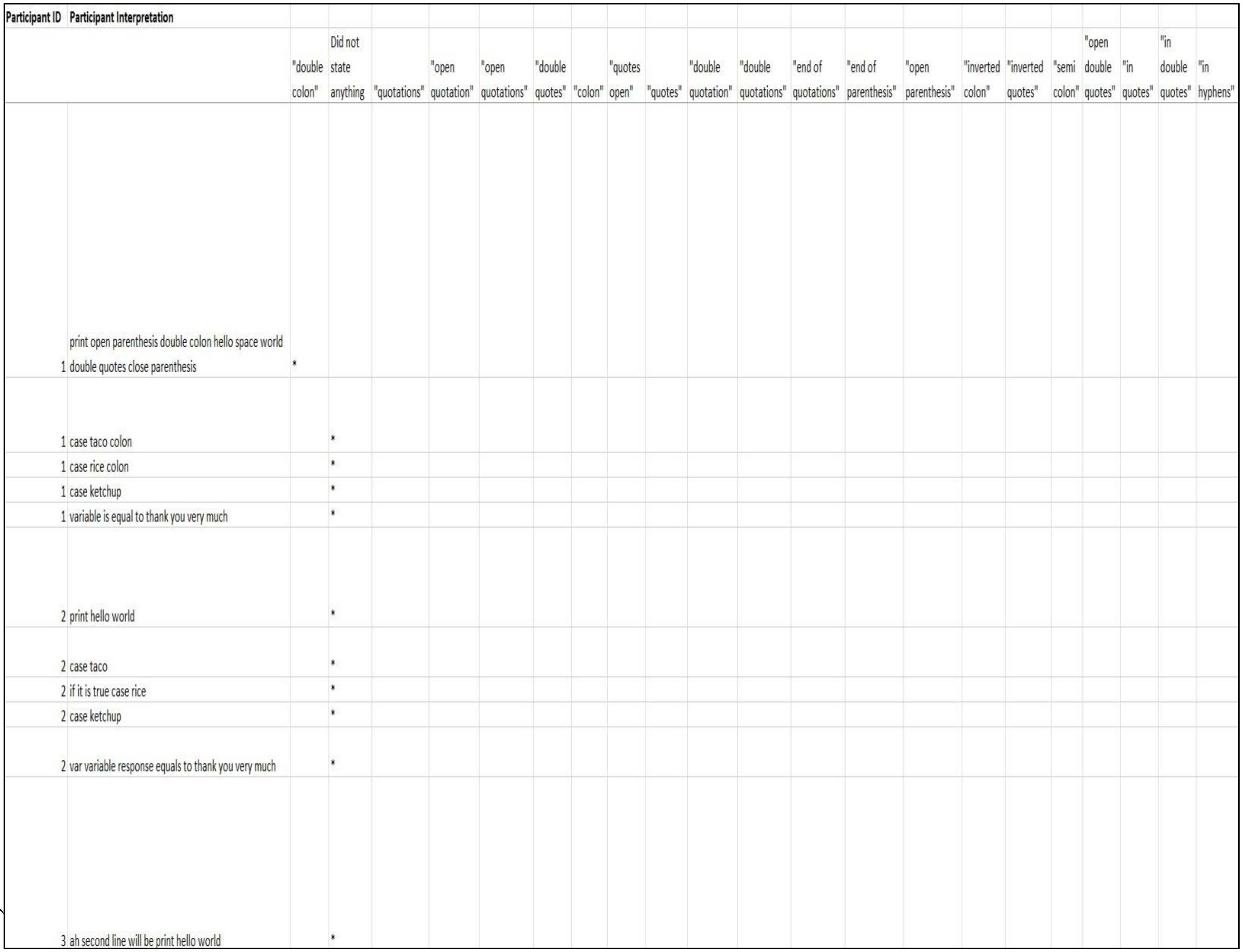


Participant ID	Participant Interpretation									
		"open parenthesis"	Did not state anything	"bracket open"	"braces open"	"parenthesis"	"parenthesis open"	"bracket"	"open brackets"	"open brace"
14	switch parenthesis food close parenthesis semi um what do you call this what ever you think um open parenthesis					*				
15	print hello world		*							
15	switch food flower bracket open		*							
16	print hello world		*							
16	switch food		*							
18	print um hello world		*							
18	switch of food open parenthesis		*							
19	print hello world		*							
19	switch food		*							
20	print hello world		*							
20	switch food		*							
21	print hello world		*							
21	switch food		*							
22	print parenthesis open double quotes hello world end double quotes parenthesis						*			
22	switch parenthesis open food close parenthesis open parenthesis						*			
23	print hello world		*							
23	switch case uh of food		*							
24	print hello world		*							
24	and after switch case we use food as a keyword in there		*							
25	print open parenthesis double quotes hello world close the uh parenthesis	*								
25	um in switch condition open parenthesis food um open curly braces	*								
26	print hello world		*							
26	take a switch case uh take an input food as an variable and um		*							

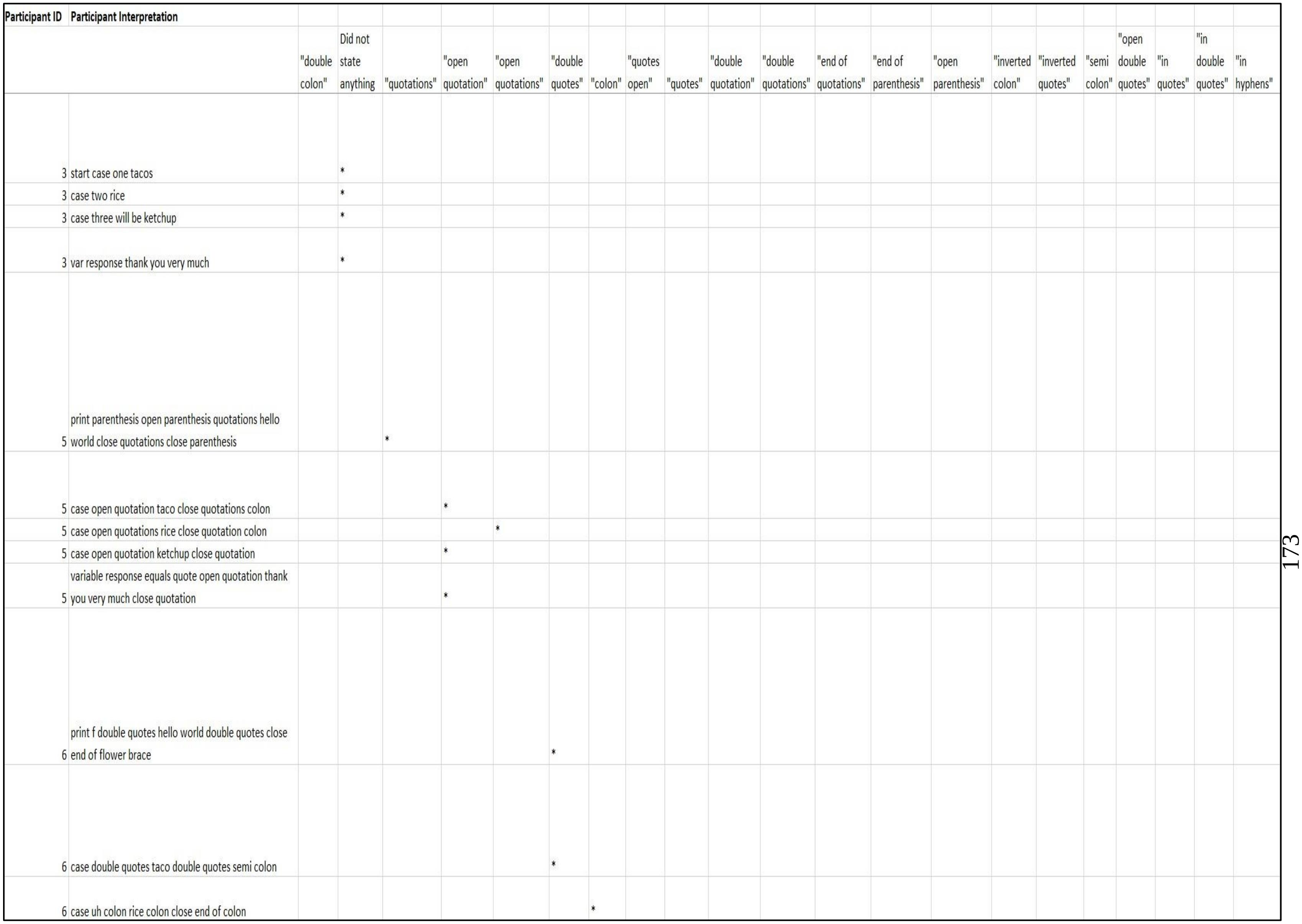


Participant ID	Participant Interpretation									
		"open parenthesis"	Did not state anything	"bracket open"	"braces open"	"parenthesis"	"parenthesis open"	"bracket"	"open brackets"	"open brace"
27	print hello world end double quotes		*							
27	switch food o eh open curly braces		*							
28	print uh brackets uh a pause double quotes hello world double quotes bracket		*							
28	next line switch bracket food bracket space curly braces							*		
29	print hello world		*							
29	switch food		*							
30	print o open brackets double quotes hello world								*	
30	switch open brace food open brace									*
31	print hello world		*							
31	switch food		*							
32	print hello world		*							
32	switch food		*							
33	print hello world		*							
33	switch food		*							
34	print hello world		*							
34	switch open brackets food close brackets open parenthesis								*	
35	print hello world		*							
35	add a switch statement		*							
36	print hello world		*							
36	switch of food		*							
37	print hello world		*							
37	switch uh with the case of food		*							
38	print hello world		*							
38	switch food		*							
	% of Participant That Stated This	**19.44444444**	**65.27778**	**2.77778**	**2.7778**	**1.388888889**	**2.777777778**	**1.388889**	**2.777778**	**1.389**
	# of Particpants That Stated This	**14**	**47**	**2**	**2**	**1**	**2**	**1**	**2**	**1**

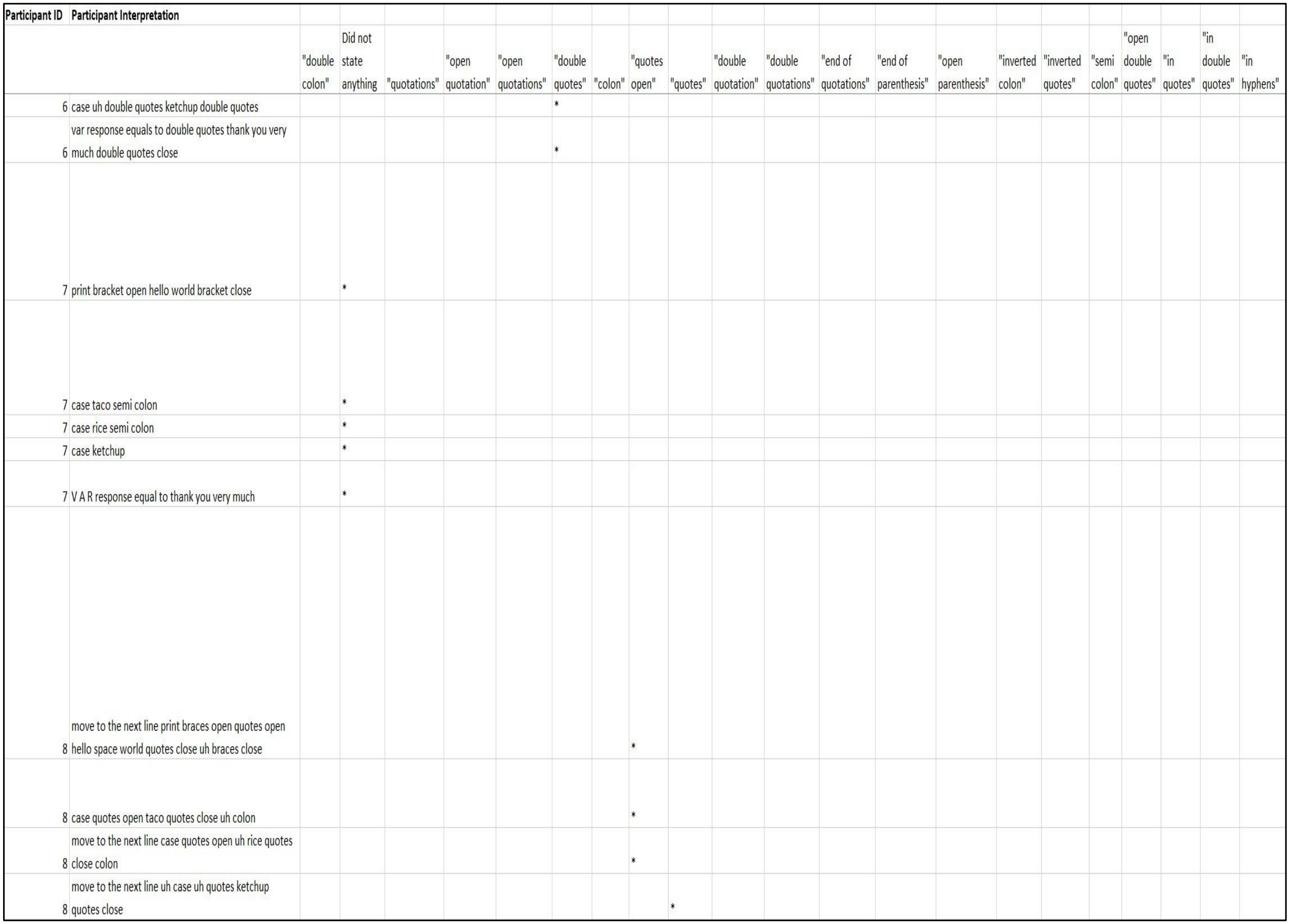
E.9) First “



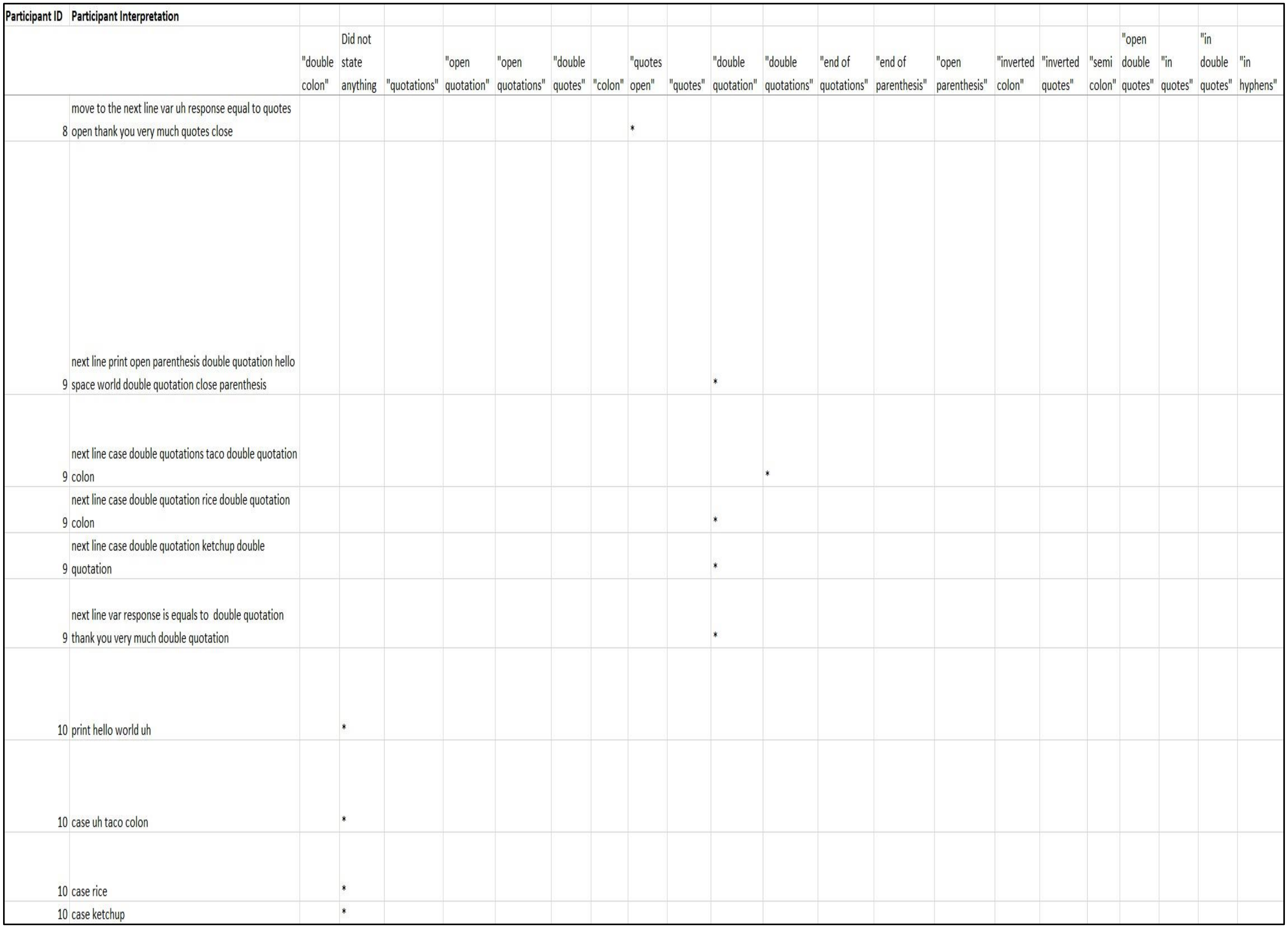
Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
1	print open parenthesis double colon hello space world double quotes close parenthesis	*																				
1	case taco colon		*																			
1	case rice colon		*																			
1	case ketchup		*																			
1	variable is equal to thank you very much		*																			
2	print hello world		*																			
2	case taco		*																			
2	if it is true case rice		*																			
2	case ketchup		*																			
2	var variable response equals to thank you very much		*																			
3	ah second line will be print hello world		*																			



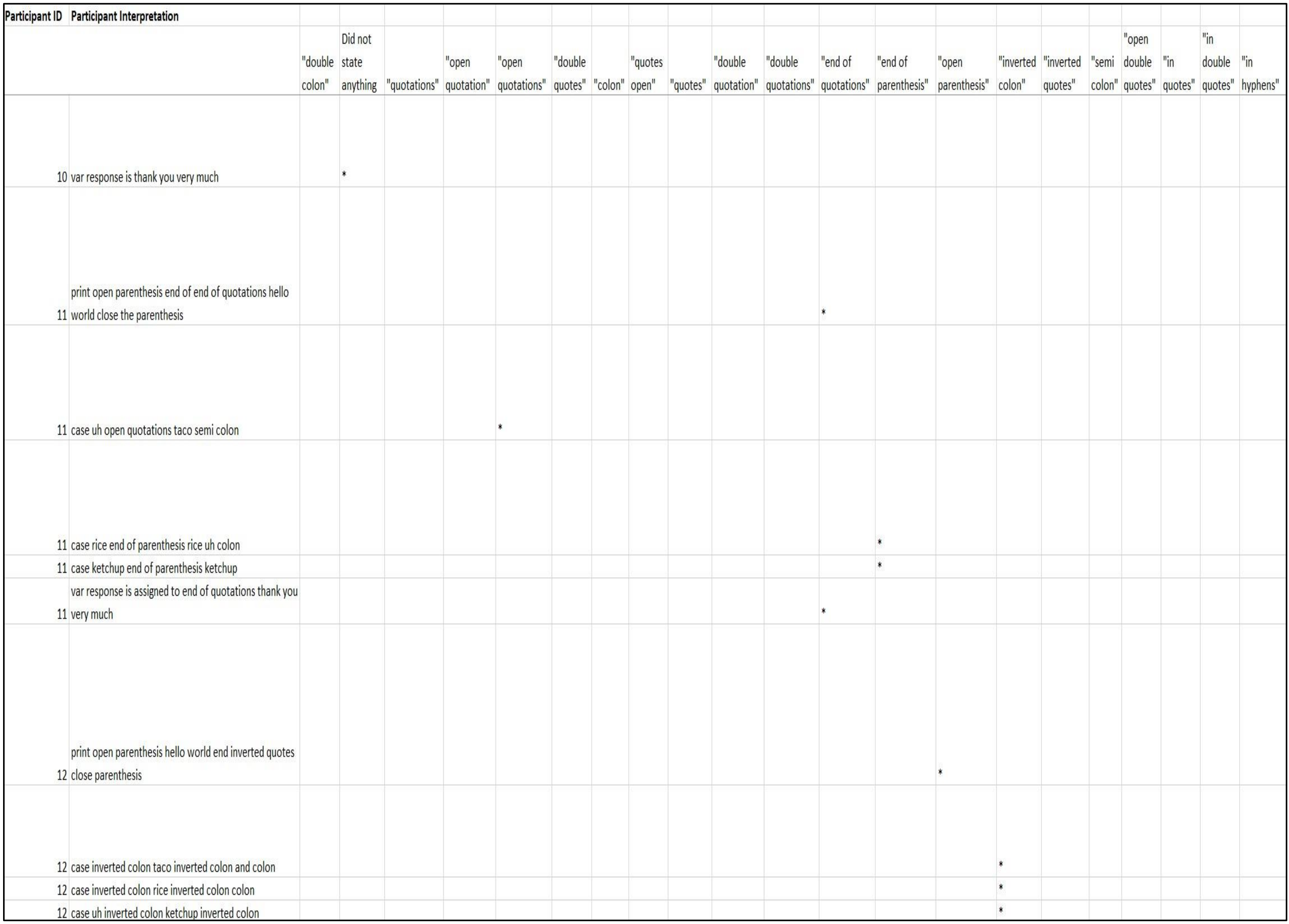
Participant ID	Participant Interpretation																					
		"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
3	start case one tacos		*																			
3	case two rice		*																			
3	case three will be ketchup		*																			
3	var response thank you very much		*																			
5	print parenthesis open parenthesis quotations hello world close quotations close parenthesis			*																		
5	case open quotation taco close quotations colon				*																	
5	case open quotations rice close quotation colon					*																
5	case open quotation ketchup close quotation				*																	
5	variable response equals quote open quotation thank you very much close quotation				*																	
6	print f double quotes hello world double quotes close end of flower brace						*															
6	case double quotes taco double quotes semi colon						*															
6	case uh colon rice colon close end of colon							*														



Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
6	case uh double quotes ketchup double quotes						*															
6	var response equals to double quotes thank you very much double quotes close						*															
7	print bracket open hello world bracket close		*																			
7	case taco semi colon		*																			
7	case rice semi colon		*																			
7	case ketchup		*																			
7	V A R response equal to thank you very much		*																			
8	move to the next line print braces open quotes open hello space world quotes close uh braces close								*													
8	case quotes open taco quotes close uh colon								*													
8	move to the next line case quotes open uh rice quotes close colon								*													
8	move to the next line uh case uh quotes ketchup quotes close									*												

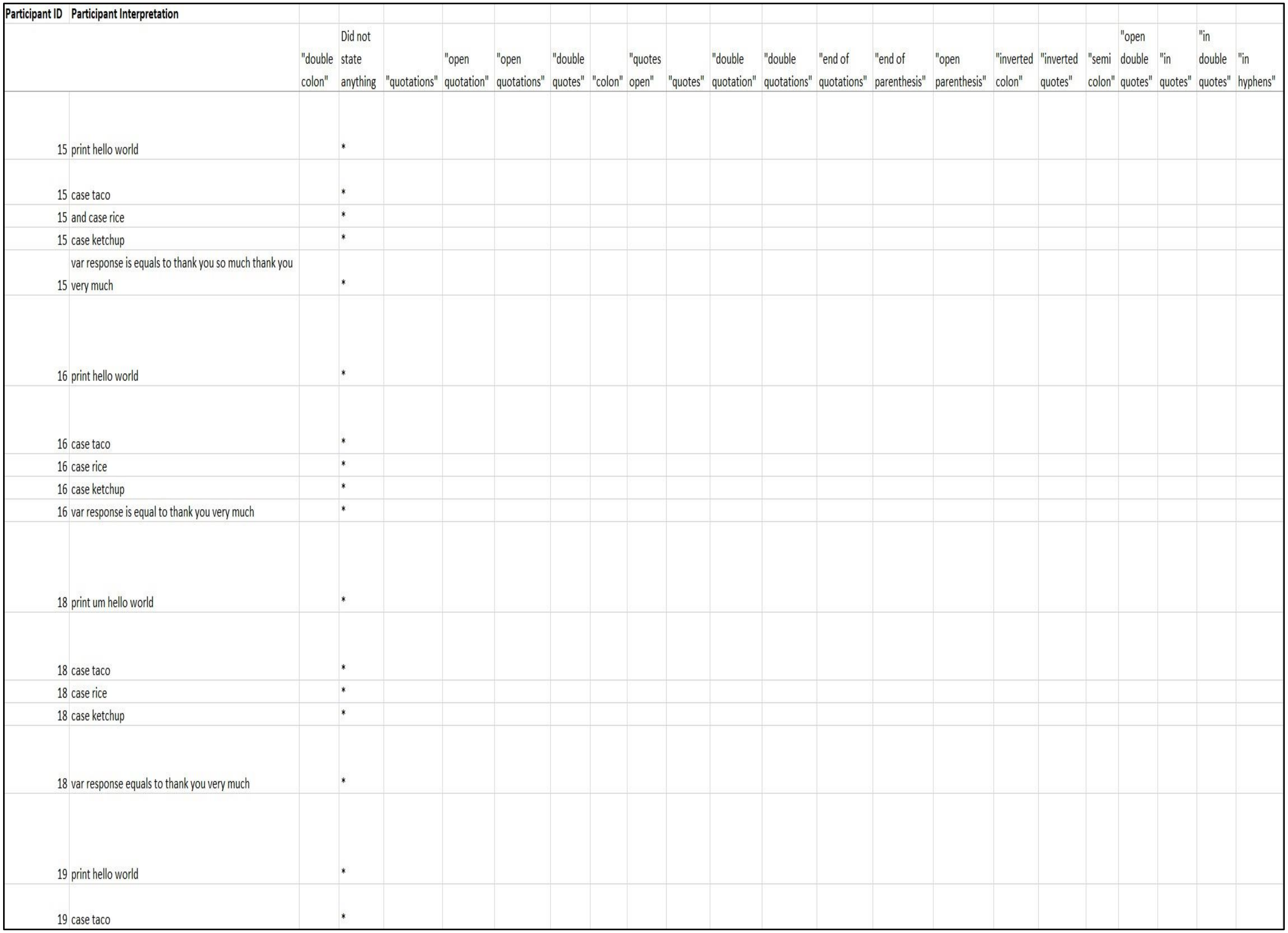


Participant ID	Participant Interpretation																					
		"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
8	move to the next line var uh response equal to quotes open thank you very much quotes close								*													
9	next line print open parenthesis double quotation hello space world double quotation close parenthesis										*											
9	next line case double quotations taco double quotation colon											*										
9	next line case double quotation rice double quotation colon										*											
9	next line case double quotation ketchup double quotation										*											
9	next line var response is equals to double quotation thank you very much double quotation										*											
10	print hello world uh		*																			
10	case uh taco colon		*																			
10	case rice		*																			
10	case ketchup		*																			

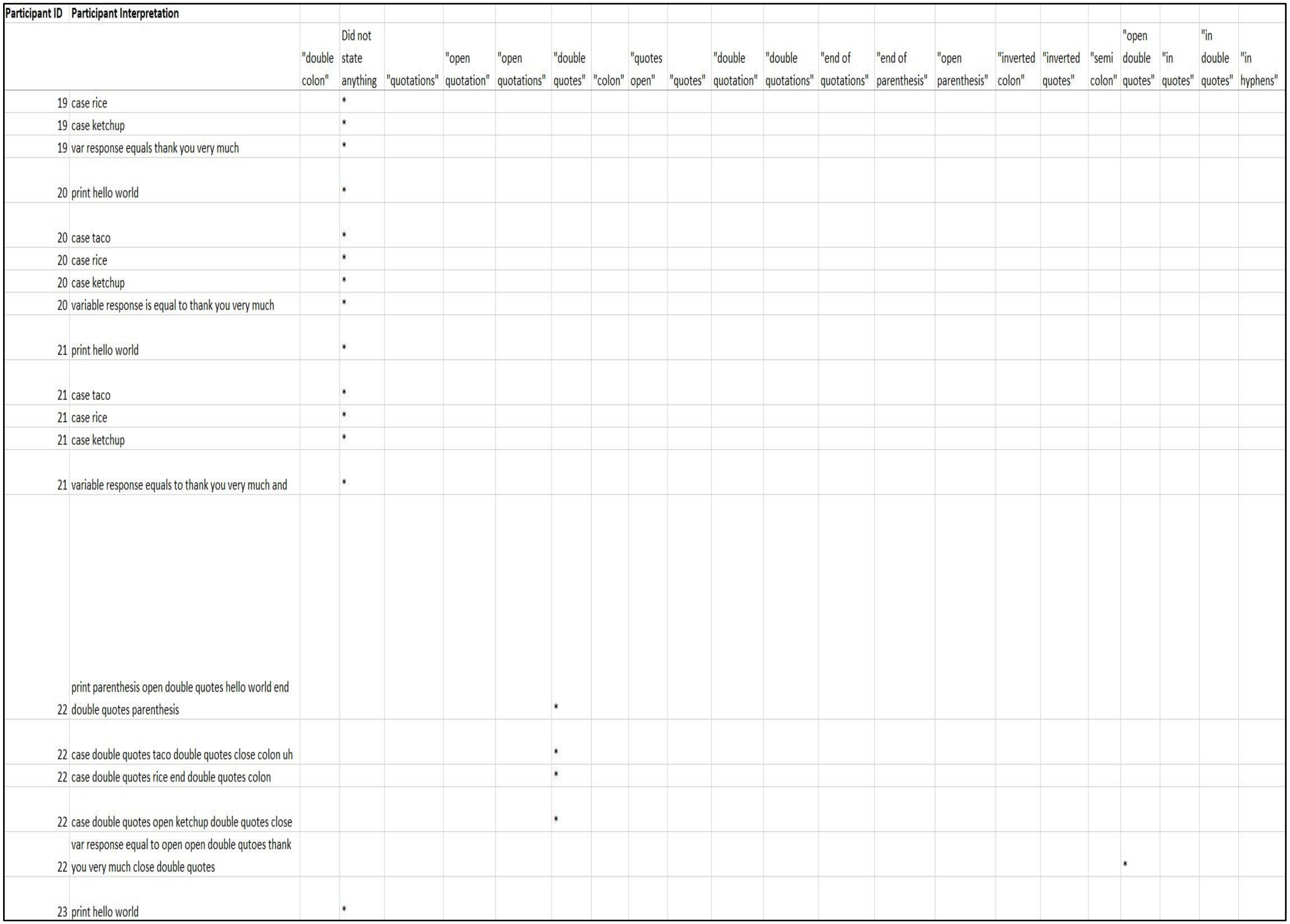


Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
10	var response is thank you very much		*																			
11	print open parenthesis end of end of quotations hello world close the parenthesis												*									
11	case uh open quotations taco semi colon					*																
11	case rice end of parenthesis rice uh colon													*								
11	case ketchup end of parenthesis ketchup													*								
11	var response is assigned to end of quotations thank you very much												*									
12	print open parenthesis hello world end inverted quotes close parenthesis														*							
12	case inverted colon taco inverted colon and colon															*						
12	case inverted colon rice inverted colon colon															*						
12	case uh inverted colon ketchup inverted colon															*						

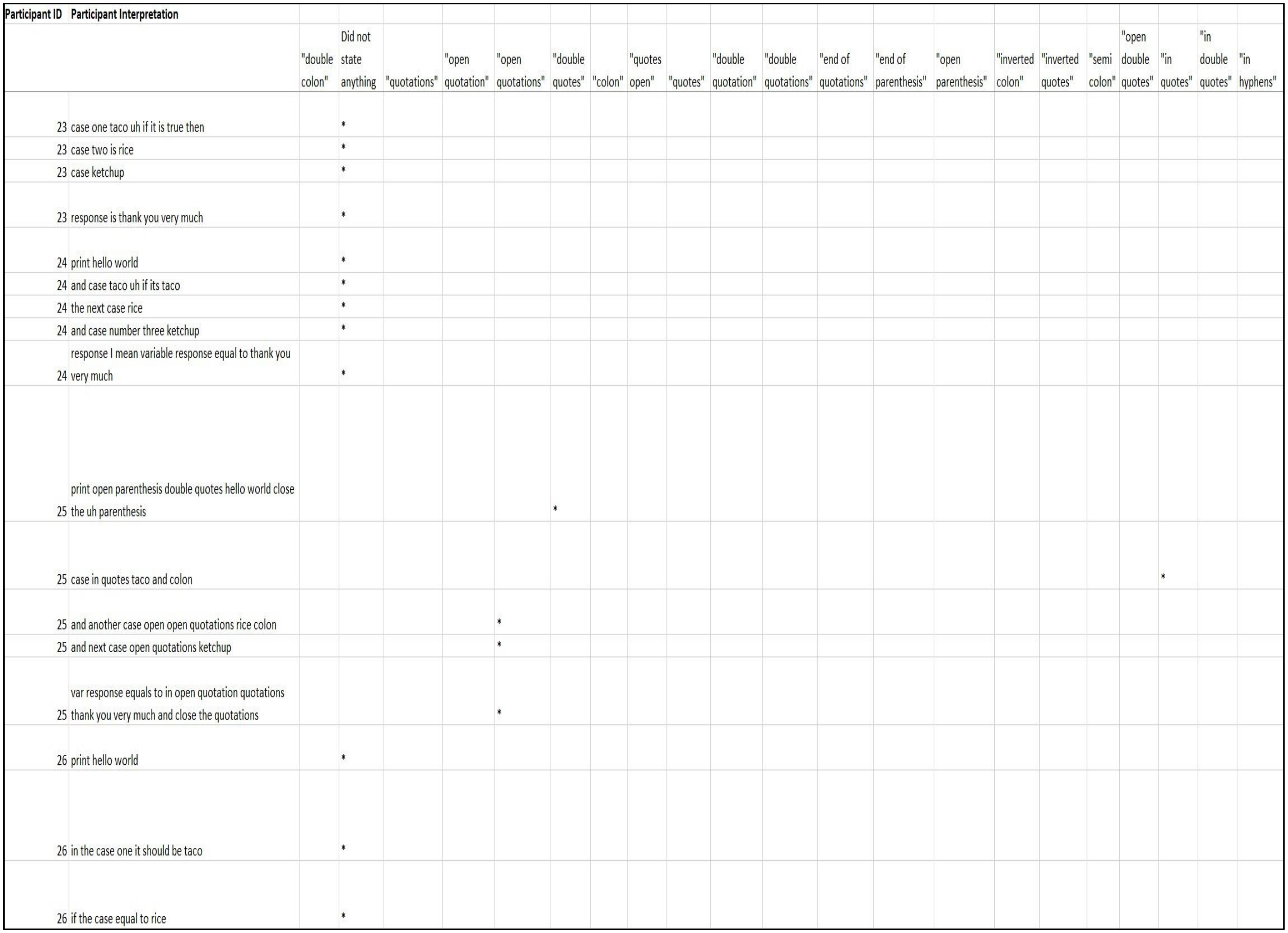
Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
12	var response equals inverted quotes thank you very much inverted quotes																*					
13	print open parenthesis double quotes hello world close double quotes close parenthesis						*															
13	case double quotes taco colon						*															
13	case double quotes rice colon						*															
13	case double quotes ketchup						*															
13	var response equals double quotes thank you very much						*															
14	print ah semi colon hello world close semi colon																	*				
14	case taco ah semi ah two dots		*																			
14	next case uh rice		*																			
14	next case ketchup		*																			
14	variable response equals thank you very much		*																			



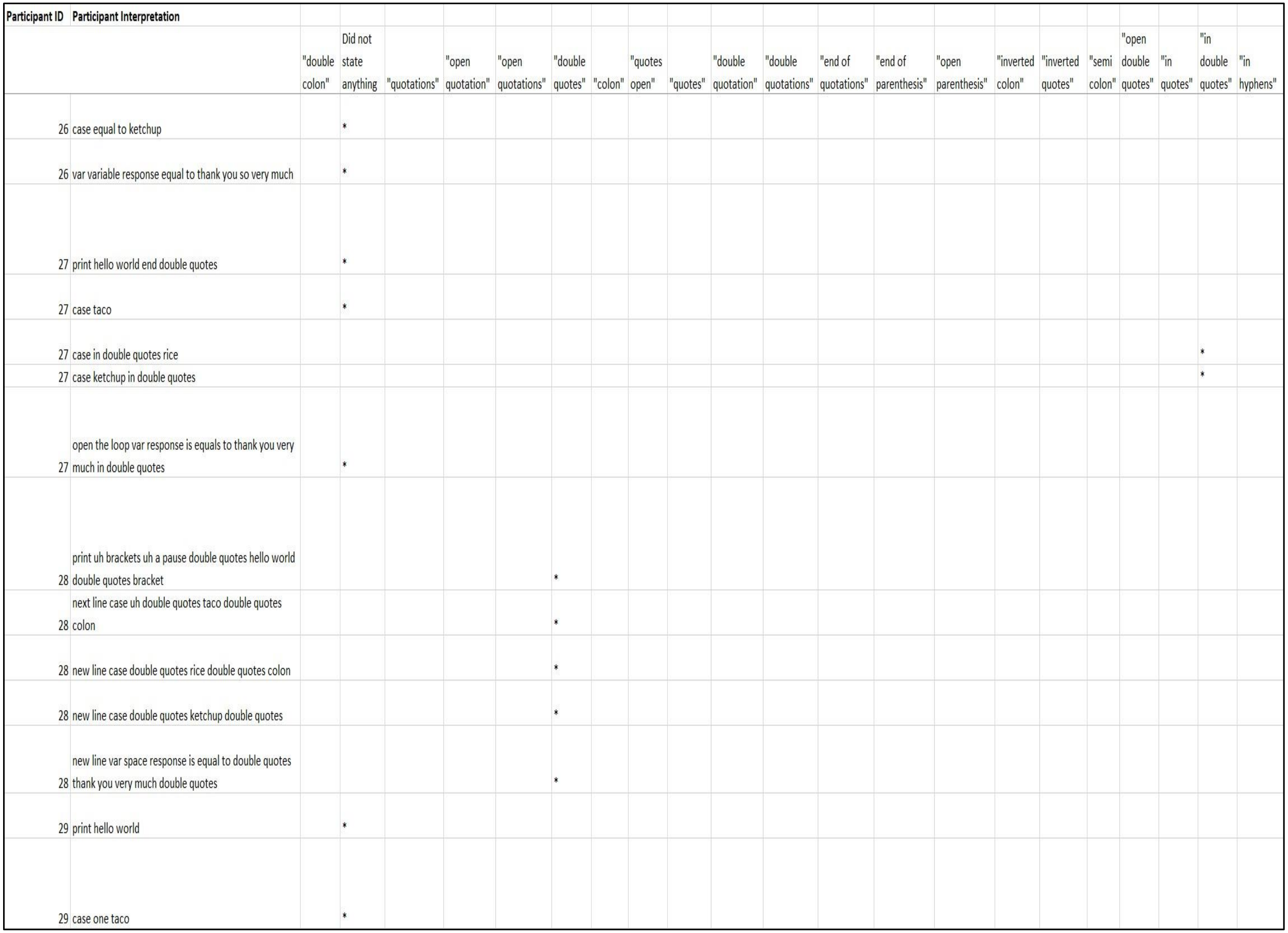
Participant ID	Participant Interpretation																					
		"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
15	print hello world		*																			
15	case taco		*																			
15	and case rice		*																			
15	case ketchup		*																			
15	var response is equals to thank you so much thank you very much		*																			
16	print hello world		*																			
16	case taco		*																			
16	case rice		*																			
16	case ketchup		*																			
16	var response is equal to thank you very much		*																			
18	print um hello world		*																			
18	case taco		*																			
18	case rice		*																			
18	case ketchup		*																			
18	var response equals to thank you very much		*																			
19	print hello world		*																			
19	case taco		*																			



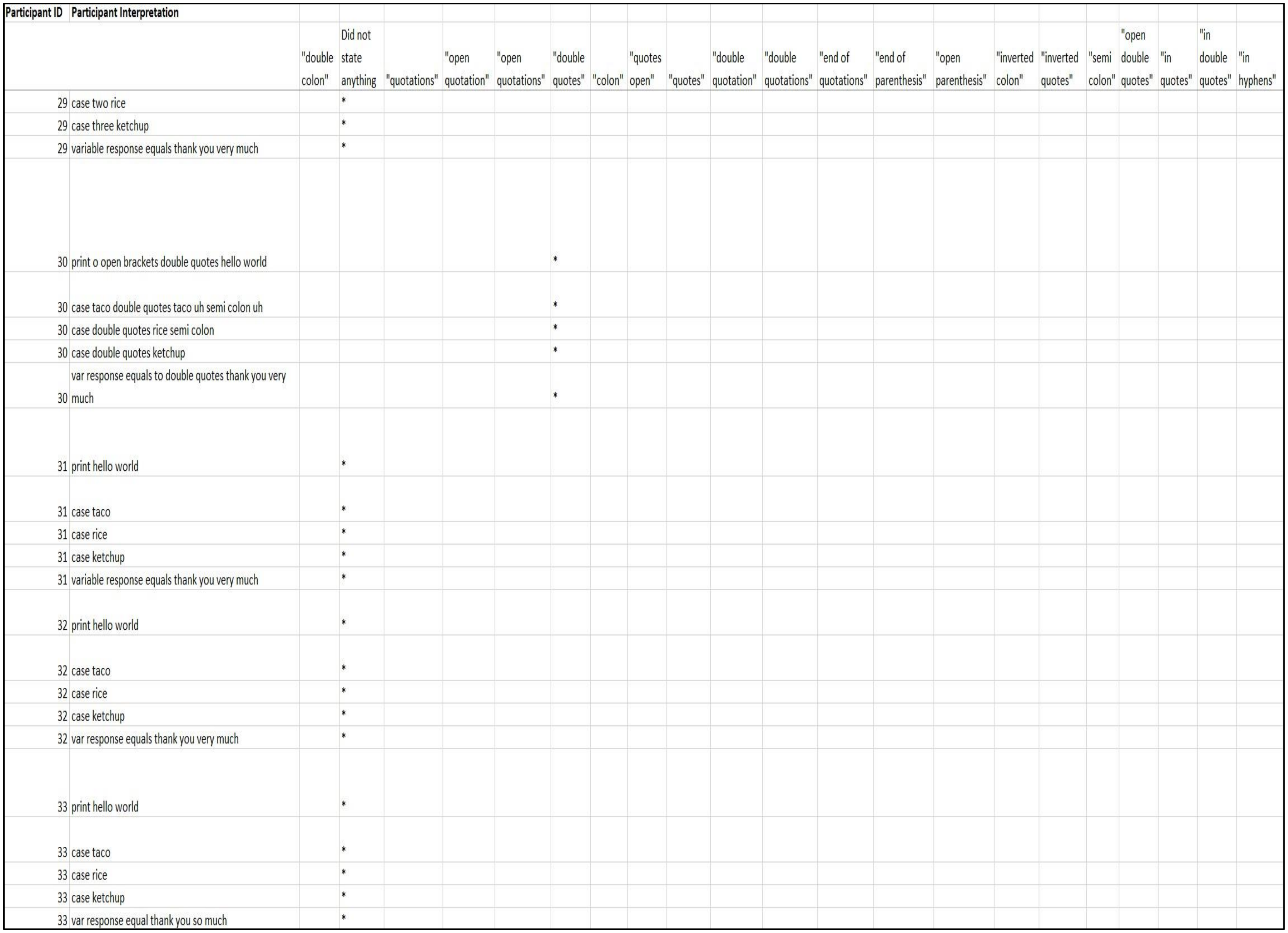
Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
19	case rice		*																			
19	case ketchup		*																			
19	var response equals thank you very much		*																			
20	print hello world		*																			
20	case taco		*																			
20	case rice		*																			
20	case ketchup		*																			
20	variable response is equal to thank you very much		*																			
21	print hello world		*																			
21	case taco		*																			
21	case rice		*																			
21	case ketchup		*																			
21	variable response equals to thank you very much and		*																			
22	print parenthesis open double quotes hello world end double quotes parenthesis						*															
22	case double quotes taco double quotes close colon uh						*															
22	case double quotes rice end double quotes colon						*															
22	case double quotes open ketchup double quotes close						*															
22	var response equal to open open double qutoes thank you very much close double quotes																		*			
23	print hello world		*																			



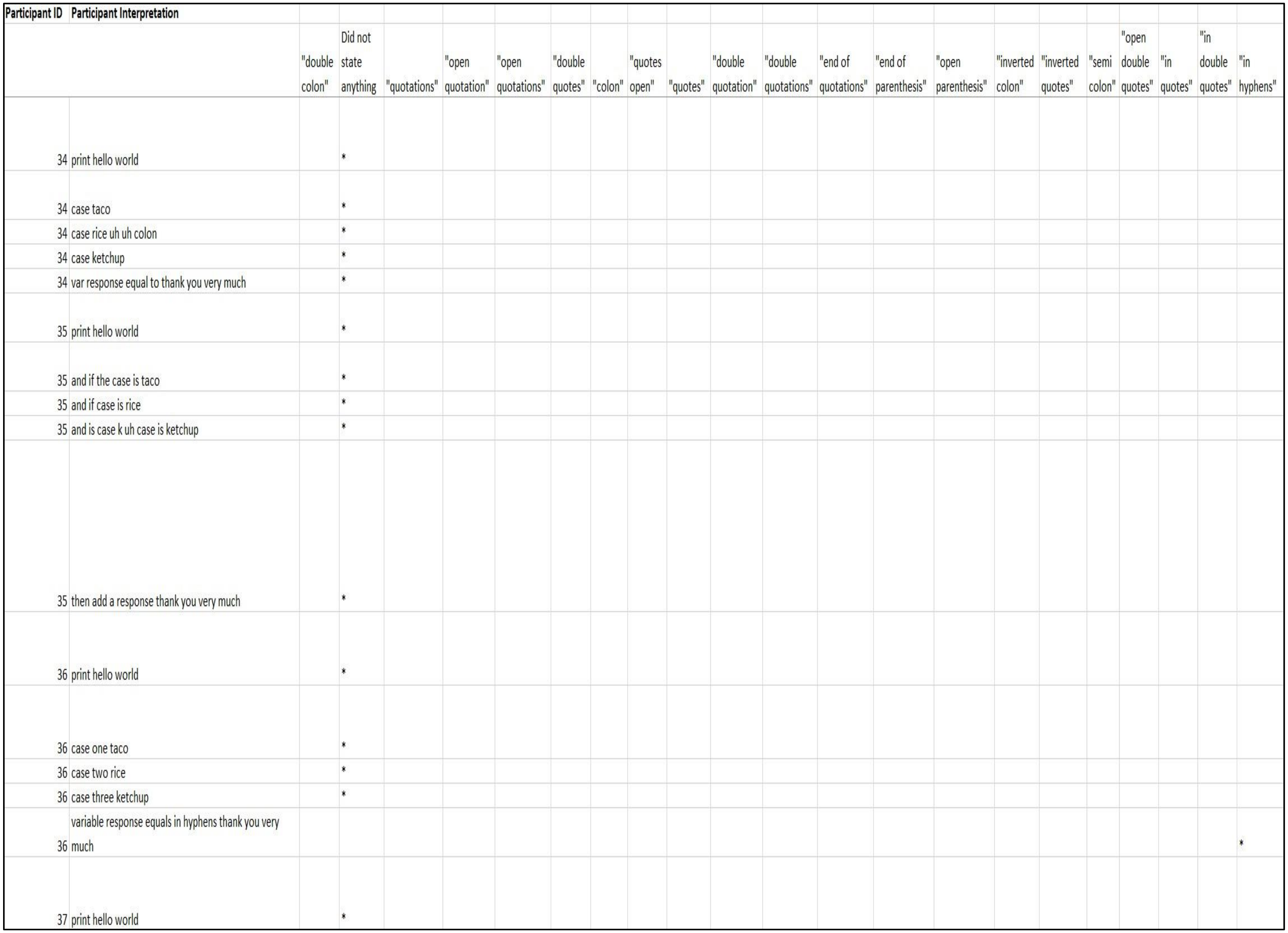
Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
23	case one taco uh if it is true then		*																			
23	case two is rice		*																			
23	case ketchup		*																			
23	response is thank you very much		*																			
24	print hello world		*																			
24	and case taco uh if its taco		*																			
24	the next case rice		*																			
24	and case number three ketchup		*																			
24	response I mean variable response equal to thank you very much		*																			
25	print open parenthesis double quotes hello world close the uh parenthesis						*															
25	case in quotes taco and colon																			*		
25	and another case open open quotations rice colon					*																
25	and next case open quotations ketchup					*																
25	var response equals to in open quotation quotations thank you very much and close the quotations					*																
26	print hello world		*																			
26	in the case one it should be taco		*																			
26	if the case equal to rice		*																			



Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
26	case equal to ketchup		*																			
26	var variable response equal to thank you so very much		*																			
27	print hello world end double quotes		*																			
27	case taco		*																			
27	case in double quotes rice																				*	
27	case ketchup in double quotes																				*	
27	open the loop var response is equals to thank you very much in double quotes		*																			
28	print uh brackets uh a pause double quotes hello world double quotes bracket						*															
28	next line case uh double quotes taco double quotes colon						*															
28	new line case double quotes rice double quotes colon						*															
28	new line case double quotes ketchup double quotes						*															
28	new line var space response is equal to double quotes thank you very much double quotes						*															
29	print hello world		*																			
29	case one taco		*																			



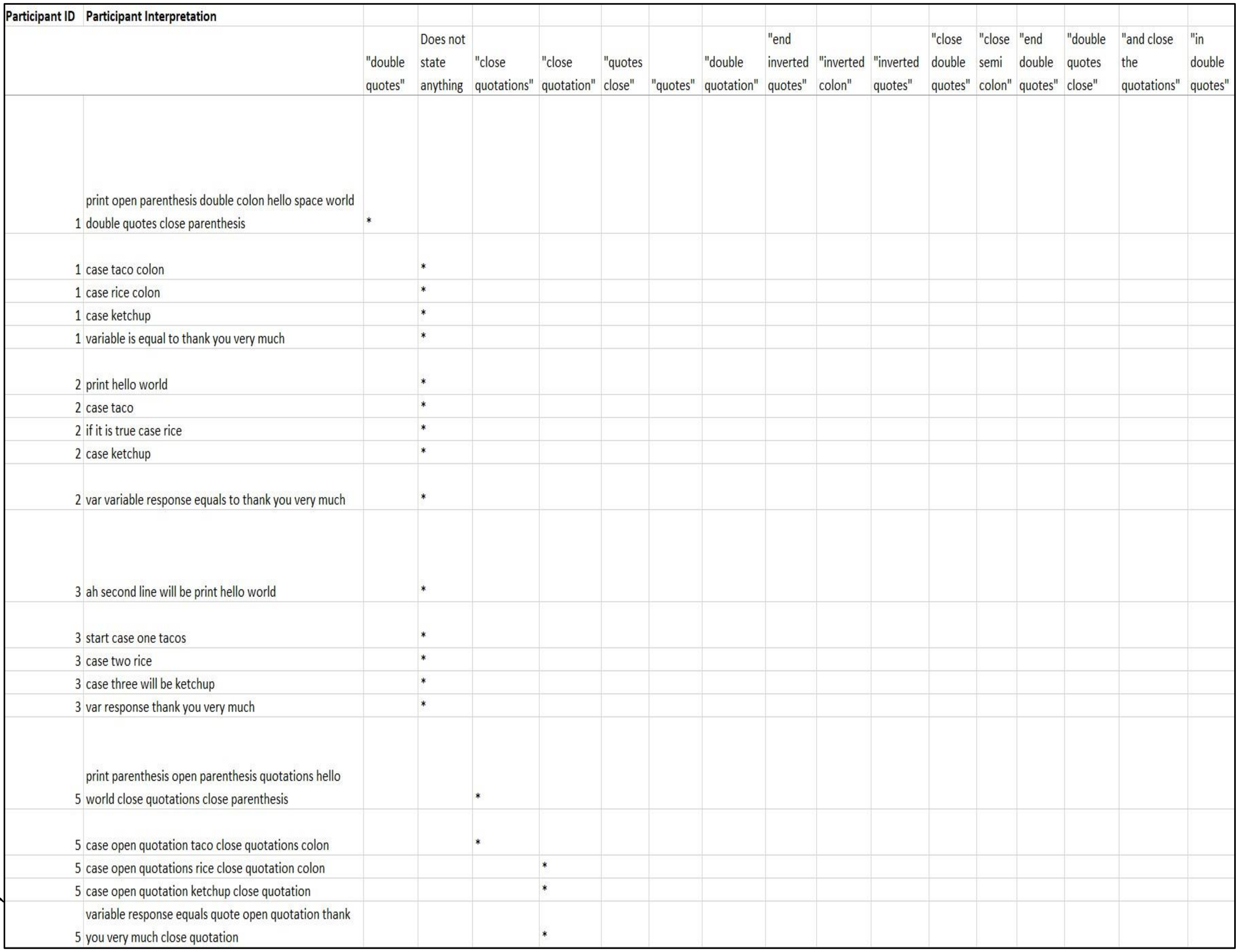
Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
29	case two rice		*																			
29	case three ketchup		*																			
29	variable response equals thank you very much		*																			
30	print o open brackets double quotes hello world						*															
30	case taco double quotes taco uh semi colon uh						*															
30	case double quotes rice semi colon						*															
30	case double quotes ketchup						*															
30	var response equals to double quotes thank you very much						*															
31	print hello world		*																			
31	case taco		*																			
31	case rice		*																			
31	case ketchup		*																			
31	variable response equals thank you very much		*																			
32	print hello world		*																			
32	case taco		*																			
32	case rice		*																			
32	case ketchup		*																			
32	var response equals thank you very much		*																			
33	print hello world		*																			
33	case taco		*																			
33	case rice		*																			
33	case ketchup		*																			
33	var response equal thank you so much		*																			



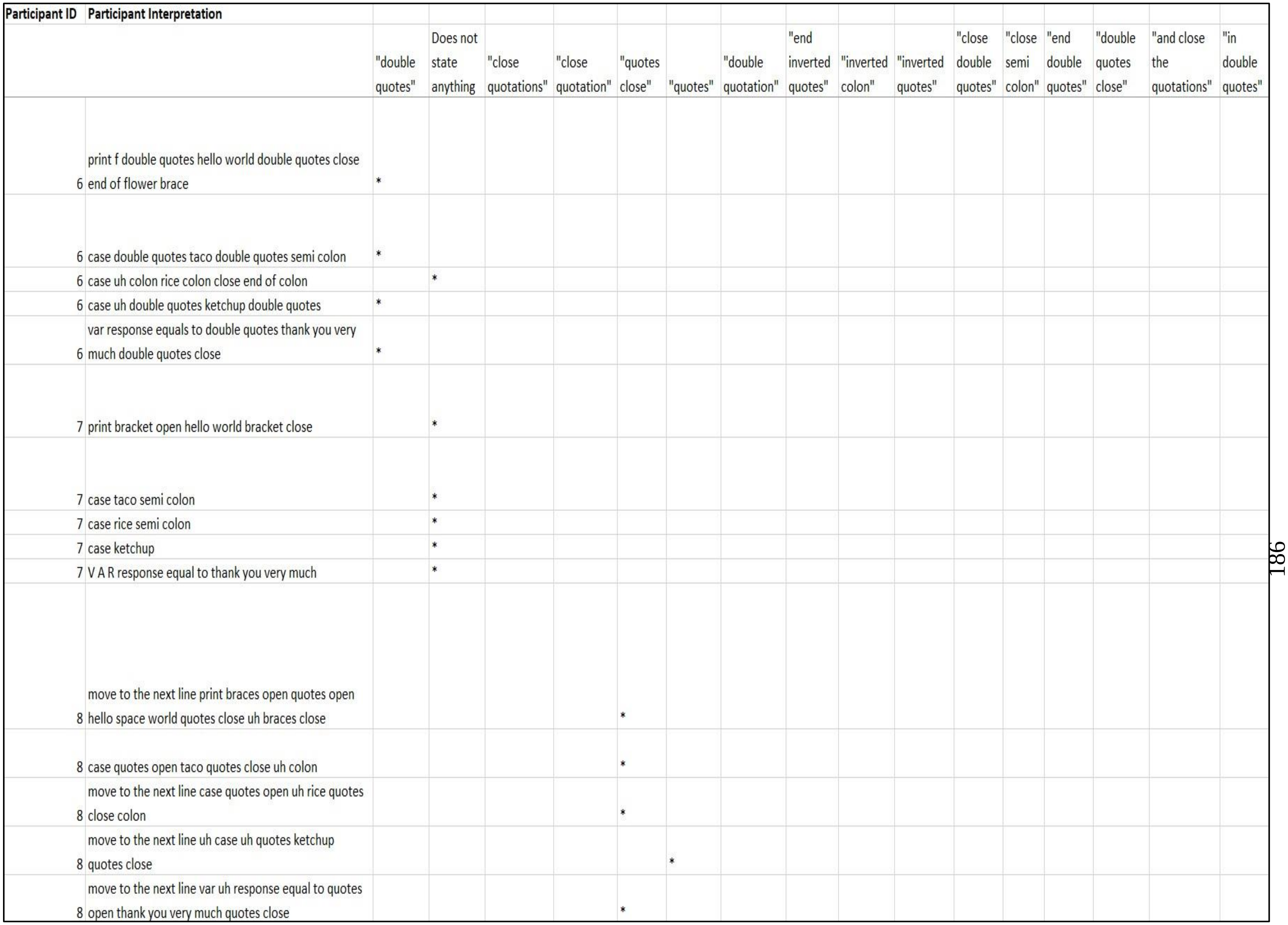
Participant ID	Participant Interpretation	"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
34	print hello world		*																			
34	case taco		*																			
34	case rice uh uh colon		*																			
34	case ketchup		*																			
34	var response equal to thank you very much		*																			
35	print hello world		*																			
35	and if the case is taco		*																			
35	and if case is rice		*																			
35	and is case k uh case is ketchup		*																			
35	then add a response thank you very much		*																			
36	print hello world		*																			
36	case one taco		*																			
36	case two rice		*																			
36	case three ketchup		*																			
36	variable response equals in hyphens thank you very much																					*
37	print hello world		*																			

Participant ID	Participant Interpretation																						
		"double colon"	Did not state anything	"quotations"	"open quotation"	"open quotations"	"double quotes"	"colon"	"quotes open"	"quotes"	"double quotation"	"double quotations"	"end of quotations"	"end of parenthesis"	"open parenthesis"	"inverted colon"	"inverted quotes"	"semi colon"	"open double quotes"	"in quotes"	"in double quotes"	"in hyphens"
37	and then case one taco and case taco		*																			
37	case rice		*																			
37	case ketchup		*																			
37	variable response equal to thank you very much		*																			
38	print hello world		*																			
38	case taco		*																			
38	case rice		*																			
38	case ketchup		*																			
38	initialize response equal to thank you very much		*																			
	% of Occurences This Was Stated	**0.55556**	**66.66667**	**0.55555556**	**1.6666667**	**2.77777778**	**13.3333**	**0.5556**	**2.22222**	**0.55556**	**2.2222222**	**0.55555556**	**1.11111111**	**1.111111111**	**0.555555556**	**1.666667**	**0.555556**	**0.556**	**0.55556**	**0.55556**	**1.11111**	**0.555556**
	# of Occurences This Was Stated	**1**	**120**	**1**	**3**	**5**	**24**	**1**	**4**	**1**	**4**	**1**	**2**	**2**	**1**	**3**	**1**	**1**	**1**	**1**	**2**	**1**

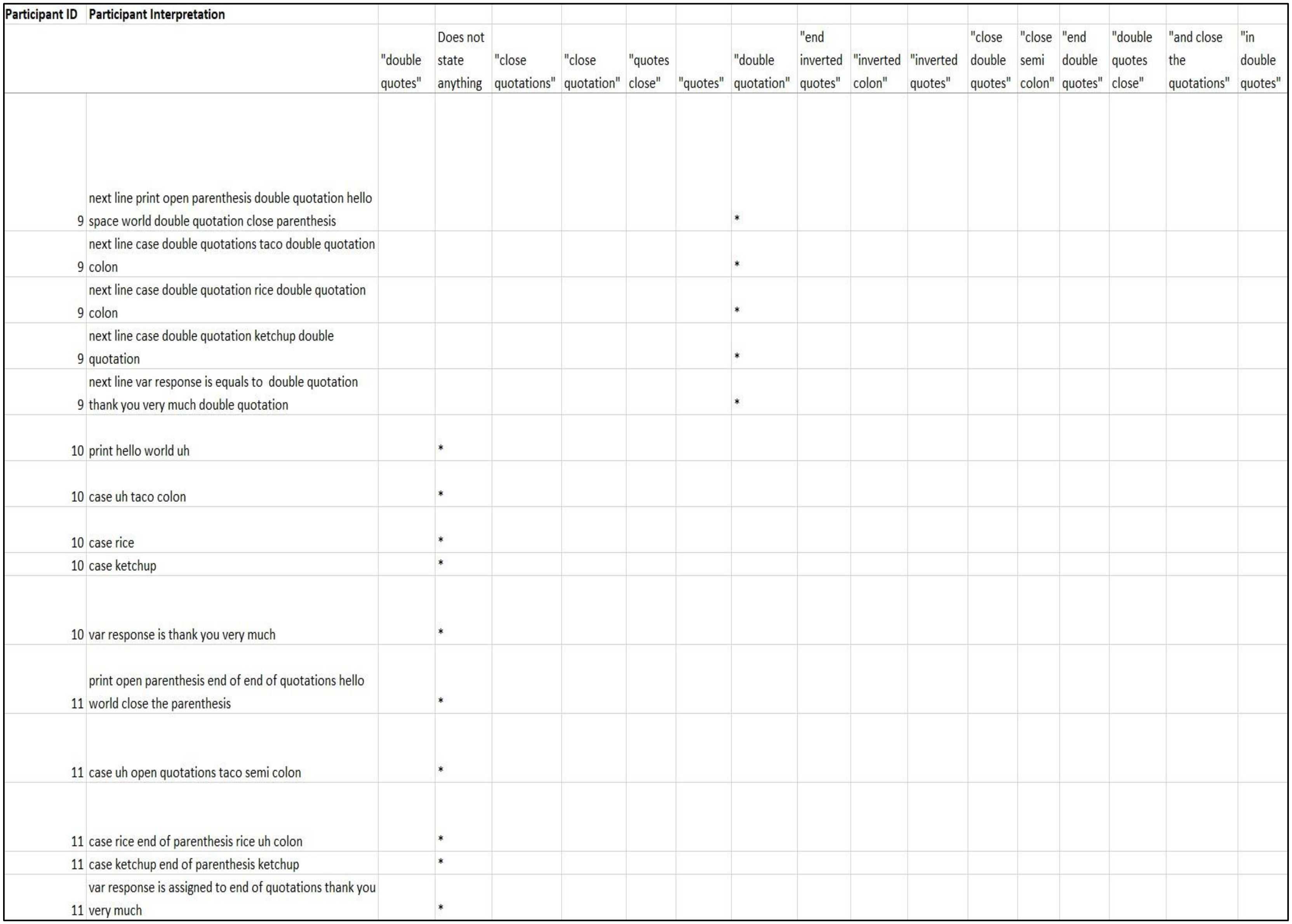
E.10) Second “



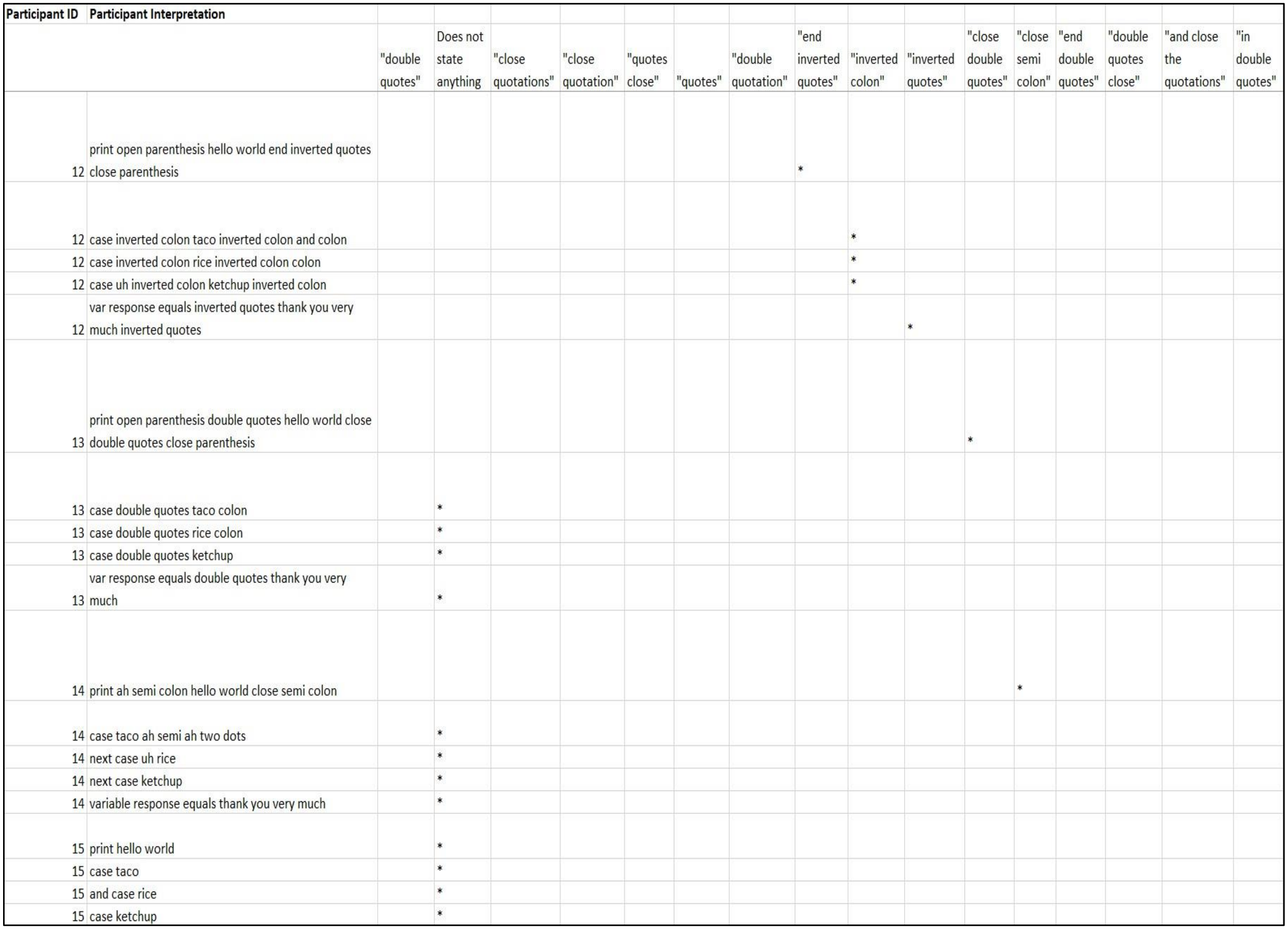
Participant ID	Participant Interpretation	"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
1	print open parenthesis double colon hello space world double quotes close parenthesis	*															
1	case taco colon		*														
1	case rice colon		*														
1	case ketchup		*														
1	variable is equal to thank you very much		*														
2	print hello world		*														
2	case taco		*														
2	if it is true case rice		*														
2	case ketchup		*														
2	var variable response equals to thank you very much		*														
3	ah second line will be print hello world		*														
3	start case one tacos		*														
3	case two rice		*														
3	case three will be ketchup		*														
3	var response thank you very much		*														
5	print parenthesis open parenthesis quotations hello world close quotations close parenthesis			*													
5	case open quotation taco close quotations colon			*													
5	case open quotations rice close quotation colon				*												
5	case open quotation ketchup close quotation				*												
5	variable response equals quote open quotation thank you very much close quotation				*												



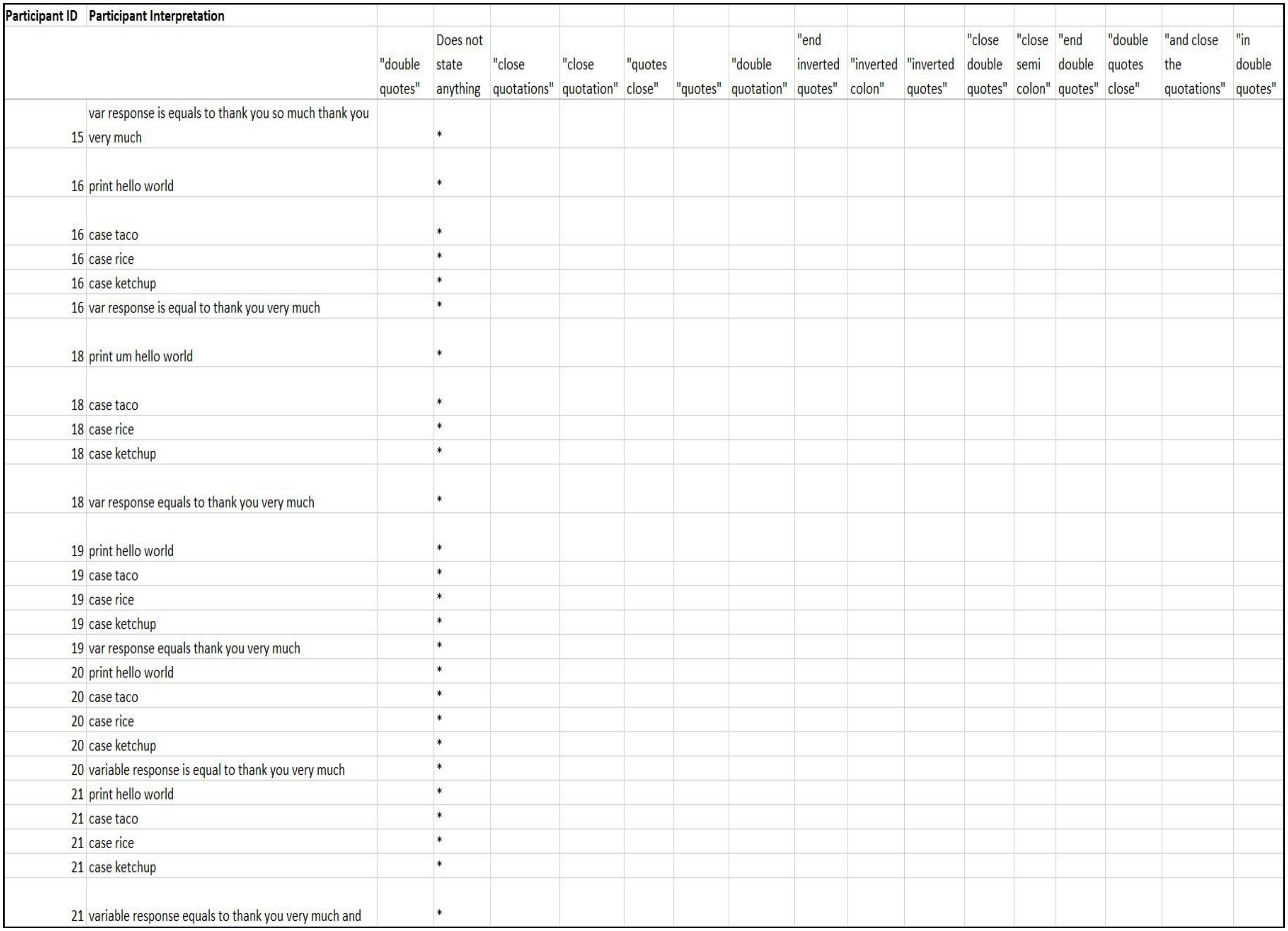
Participant ID	Participant Interpretation	"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
6	print f double quotes hello world double quotes close end of flower brace	*															
6	case double quotes taco double quotes semi colon	*															
6	case uh colon rice colon close end of colon		*														
6	case uh double quotes ketchup double quotes	*															
6	var response equals to double quotes thank you very much double quotes close	*															
7	print bracket open hello world bracket close		*														
7	case taco semi colon		*														
7	case rice semi colon		*														
7	case ketchup		*														
7	V A R response equal to thank you very much		*														
8	move to the next line print braces open quotes open hello space world quotes close uh braces close					*											
8	case quotes open taco quotes close uh colon					*											
8	move to the next line case quotes open uh rice quotes close colon					*											
8	move to the next line uh case uh quotes ketchup quotes close						*										
8	move to the next line var uh response equal to quotes open thank you very much quotes close					*											



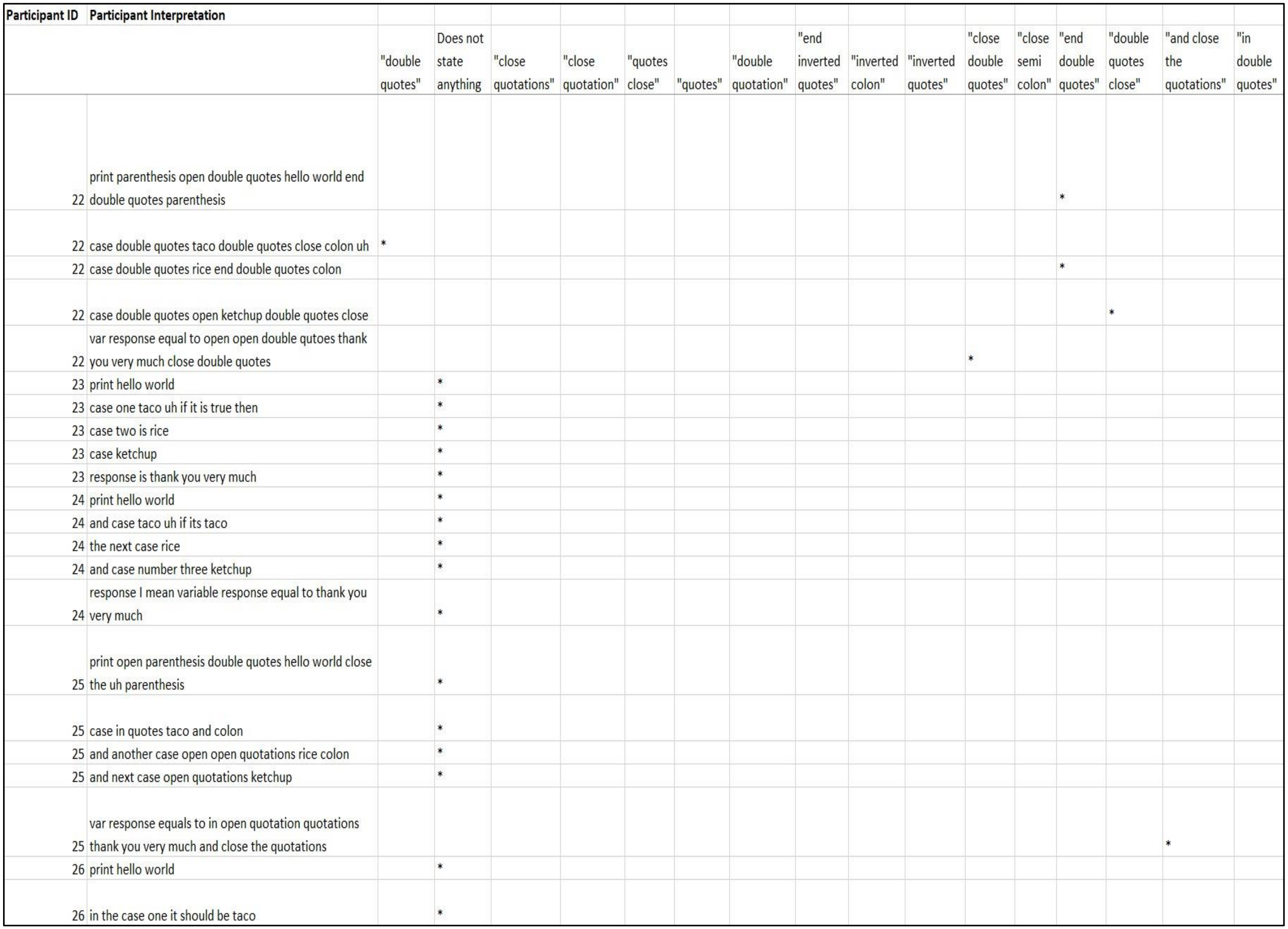
Participant ID	Participant Interpretation																
		"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
9	next line print open parenthesis double quotation hello space world double quotation close parenthesis							*									
9	next line case double quotations taco double quotation colon							*									
9	next line case double quotation rice double quotation colon							*									
9	next line case double quotation ketchup double quotation							*									
9	next line var response is equals to double quotation thank you very much double quotation							*									
10	print hello world uh		*														
10	case uh taco colon		*														
10	case rice		*														
10	case ketchup		*														
10	var response is thank you very much		*														
11	print open parenthesis end of end of quotations hello world close the parenthesis		*														
11	case uh open quotations taco semi colon		*														
11	case rice end of parenthesis rice uh colon		*														
11	case ketchup end of parenthesis ketchup		*														
11	var response is assigned to end of quotations thank you very much		*														



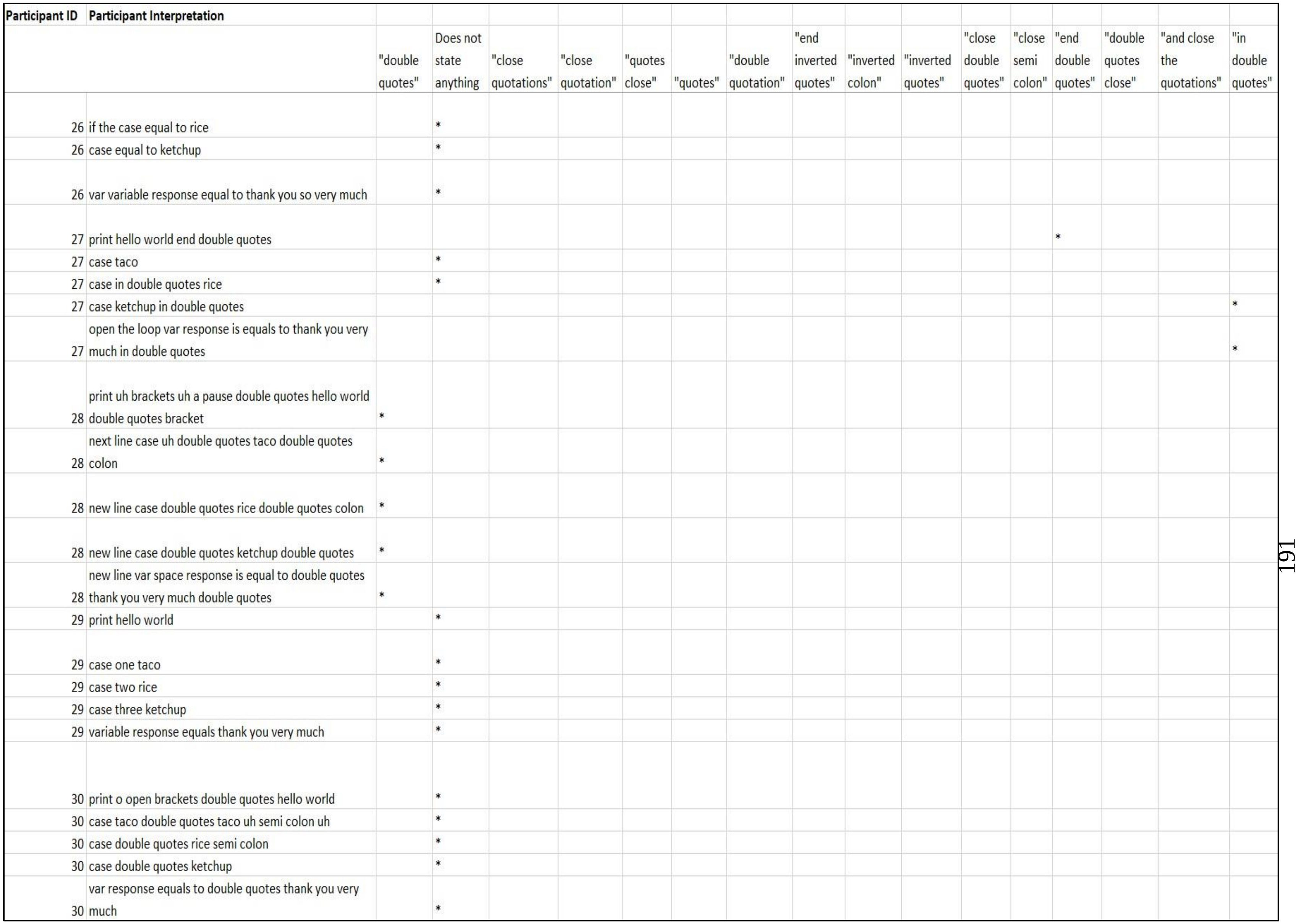
Participant ID	Participant Interpretation	"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
12	print open parenthesis hello world end inverted quotes close parenthesis								*								
12	case inverted colon taco inverted colon and colon									*							
12	case inverted colon rice inverted colon colon									*							
12	case uh inverted colon ketchup inverted colon									*							
12	var response equals inverted quotes thank you very much inverted quotes										*						
13	print open parenthesis double quotes hello world close double quotes close parenthesis											*					
13	case double quotes taco colon		*														
13	case double quotes rice colon		*														
13	case double quotes ketchup		*														
13	var response equals double quotes thank you very much		*														
14	print ah semi colon hello world close semi colon												*				
14	case taco ah semi ah two dots		*														
14	next case uh rice		*														
14	next case ketchup		*														
14	variable response equals thank you very much		*														
15	print hello world		*														
15	case taco		*														
15	and case rice		*														
15	case ketchup		*														



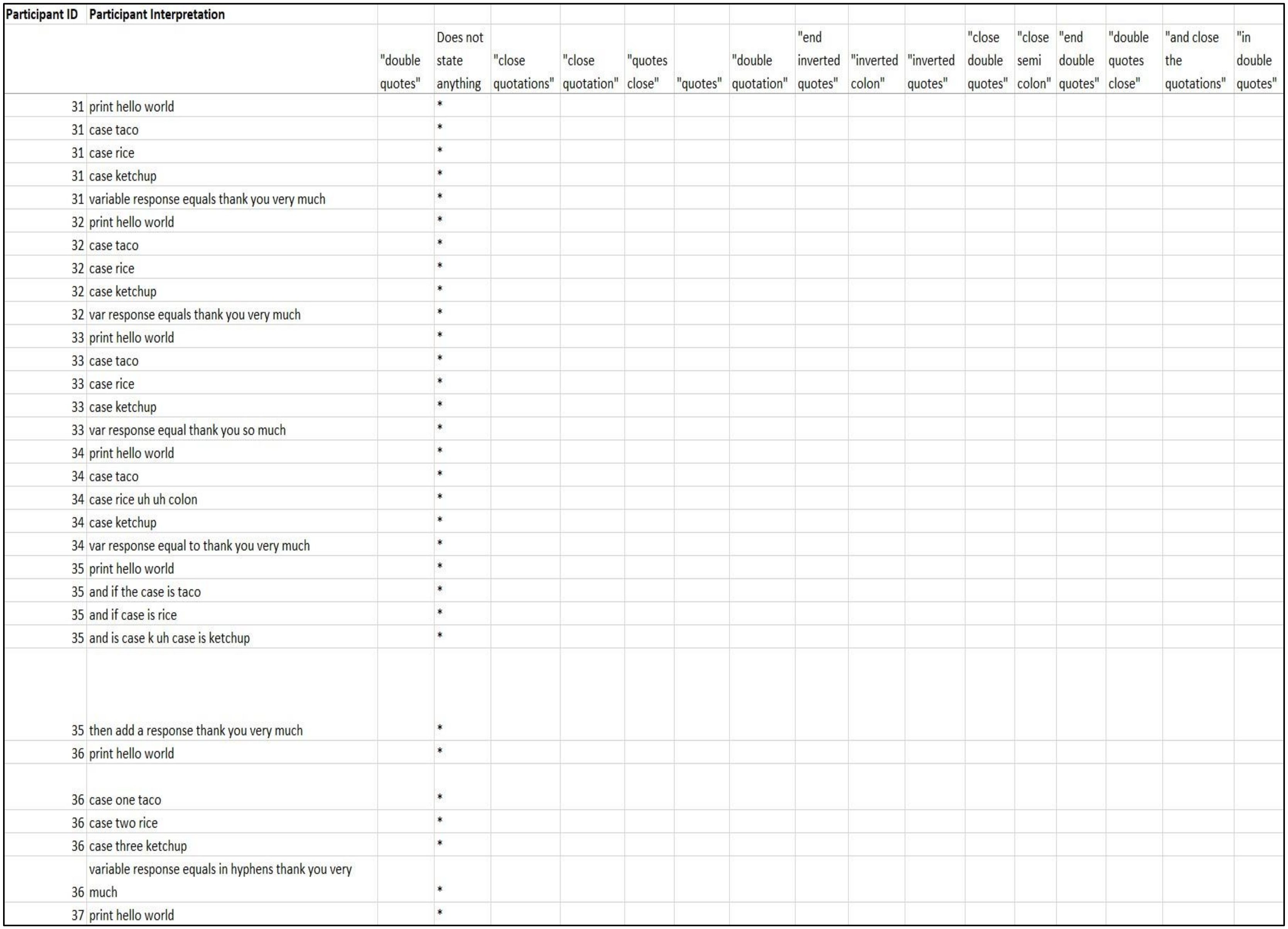
Participant ID	Participant Interpretation																
		"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
15	var response is equals to thank you so much thank you very much		*														
16	print hello world		*														
16	case taco		*														
16	case rice		*														
16	case ketchup		*														
16	var response is equal to thank you very much		*														
18	print um hello world		*														
18	case taco		*														
18	case rice		*														
18	case ketchup		*														
18	var response equals to thank you very much		*														
19	print hello world		*														
19	case taco		*														
19	case rice		*														
19	case ketchup		*														
19	var response equals thank you very much		*														
20	print hello world		*														
20	case taco		*														
20	case rice		*														
20	case ketchup		*														
20	variable response is equal to thank you very much		*														
21	print hello world		*														
21	case taco		*														
21	case rice		*														
21	case ketchup		*														
21	variable response equals to thank you very much and		*														



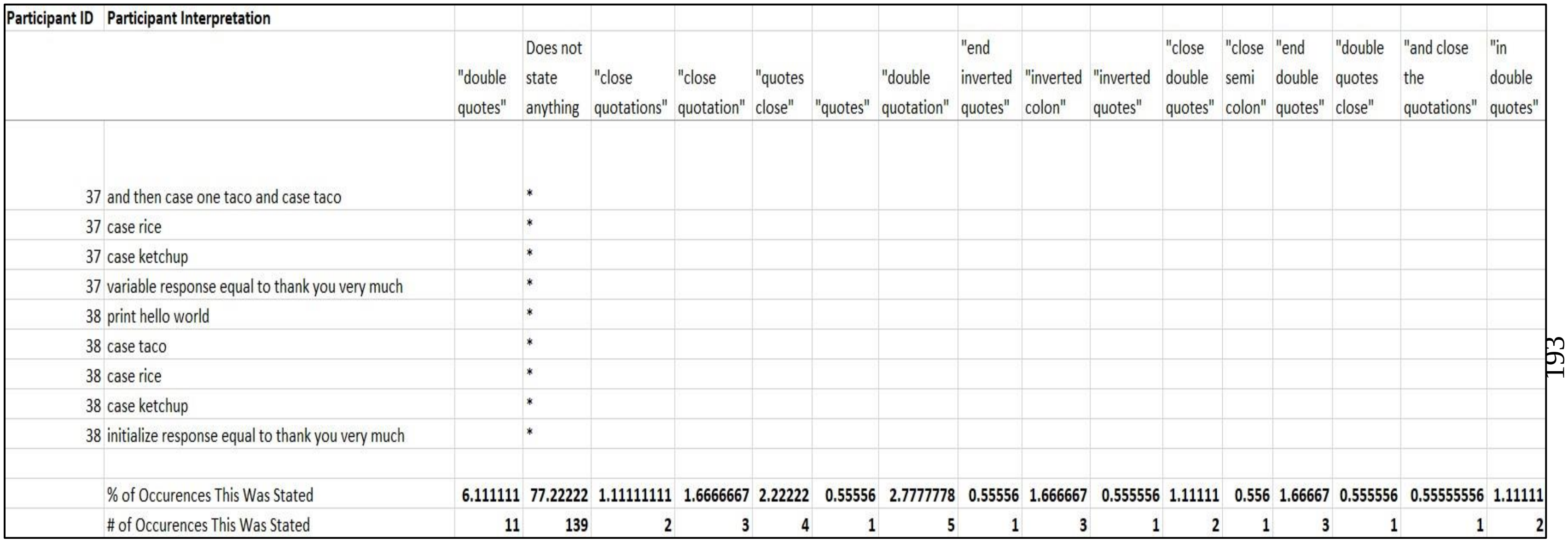
Participant ID	Participant Interpretation																
		"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
22	print parenthesis open double quotes hello world end double quotes parenthesis													*			
22	case double quotes taco double quotes close colon uh	*															
22	case double quotes rice end double quotes colon													*			
22	case double quotes open ketchup double quotes close														*		
22	var response equal to open open double qutoes thank you very much close double quotes											*					
23	print hello world		*														
23	case one taco uh if it is true then		*														
23	case two is rice		*														
23	case ketchup		*														
23	response is thank you very much		*														
24	print hello world		*														
24	and case taco uh if its taco		*														
24	the next case rice		*														
24	and case number three ketchup		*														
24	response I mean variable response equal to thank you very much		*														
25	print open parenthesis double quotes hello world close the uh parenthesis		*														
25	case in quotes taco and colon		*														
25	and another case open open quotations rice colon		*														
25	and next case open quotations ketchup		*														
25	var response equals to in open quotation quotations thank you very much and close the quotations															*	
26	print hello world		*														
26	in the case one it should be taco		*														



Participant ID	Participant Interpretation	"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
26	if the case equal to rice		*														
26	case equal to ketchup		*														
26	var variable response equal to thank you so very much		*														
27	print hello world end double quotes													*			
27	case taco		*														
27	case in double quotes rice		*														
27	case ketchup in double quotes																*
27	open the loop var response is equals to thank you very much in double quotes																*
28	print uh brackets uh a pause double quotes hello world double quotes bracket	*															
28	next line case uh double quotes taco double quotes colon	*															
28	new line case double quotes rice double quotes colon	*															
28	new line case double quotes ketchup double quotes	*															
28	new line var space response is equal to double quotes thank you very much double quotes	*															
29	print hello world		*														
29	case one taco		*														
29	case two rice		*														
29	case three ketchup		*														
29	variable response equals thank you very much		*														
30	print o open brackets double quotes hello world		*														
30	case taco double quotes taco uh semi colon uh		*														
30	case double quotes rice semi colon		*														
30	case double quotes ketchup		*														
30	var response equals to double quotes thank you very much		*														

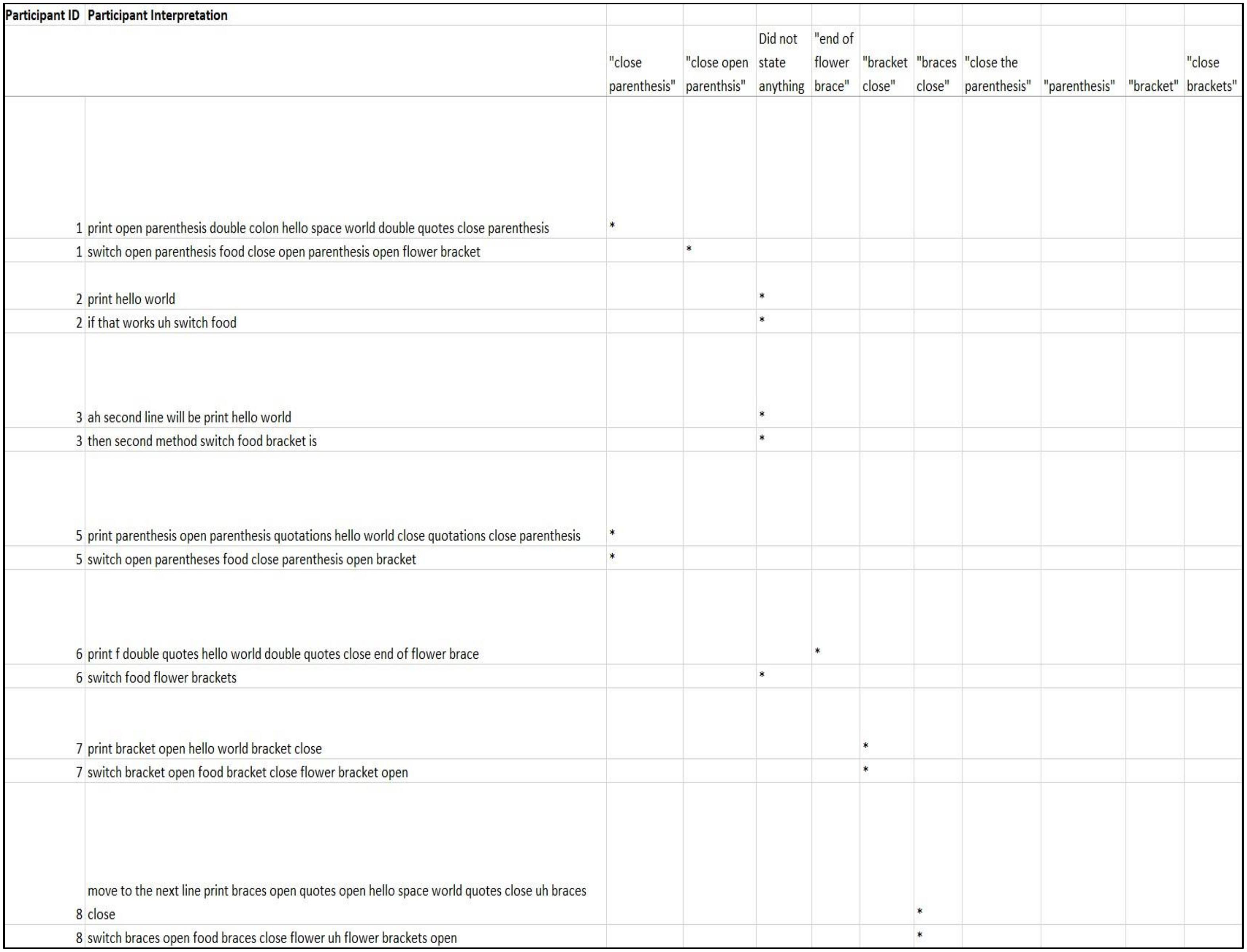


Participant ID	Participant Interpretation																
		"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
31	print hello world		*														
31	case taco		*														
31	case rice		*														
31	case ketchup		*														
31	variable response equals thank you very much		*														
32	print hello world		*														
32	case taco		*														
32	case rice		*														
32	case ketchup		*														
32	var response equals thank you very much		*														
33	print hello world		*														
33	case taco		*														
33	case rice		*														
33	case ketchup		*														
33	var response equal thank you so much		*														
34	print hello world		*														
34	case taco		*														
34	case rice uh uh colon		*														
34	case ketchup		*														
34	var response equal to thank you very much		*														
35	print hello world		*														
35	and if the case is taco		*														
35	and if case is rice		*														
35	and is case k uh case is ketchup		*														
35	then add a response thank you very much		*														
36	print hello world		*														
36	case one taco		*														
36	case two rice		*														
36	case three ketchup		*														
36	variable response equals in hyphens thank you very much		*														
37	print hello world		*														

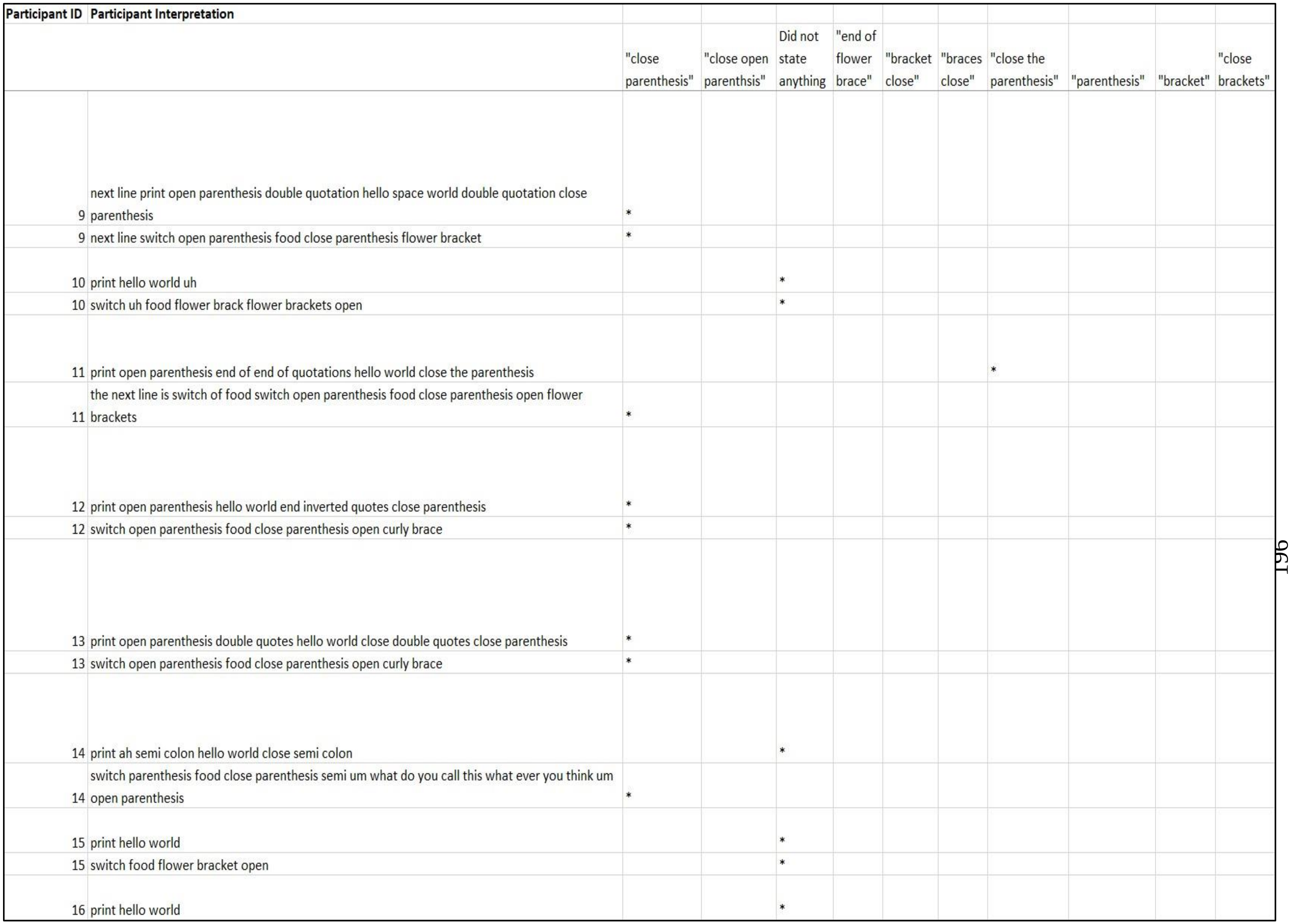


Participant ID	Participant Interpretation																
		"double quotes"	Does not state anything	"close quotations"	"close quotation"	"quotes close"	"quotes"	"double quotation"	"end inverted quotes"	"inverted colon"	"inverted quotes"	"close double quotes"	"close semi colon"	"end double quotes"	"double quotes close"	"and close the quotations"	"in double quotes"
37	and then case one taco and case taco		*														
37	case rice		*														
37	case ketchup		*														
37	variable response equal to thank you very much		*														
38	print hello world		*														
38	case taco		*														
38	case rice		*														
38	case ketchup		*														
38	initialize response equal to thank you very much		*														
	% of Occurences This Was Stated	**6.111111**	**77.22222**	**1.11111111**	**1.6666667**	**2.22222**	**0.55556**	**2.7777778**	**0.55556**	**1.666667**	**0.555556**	**1.11111**	**0.556**	**1.66667**	**0.555556**	**0.55555556**	**1.11111**
	# of Occurences This Was Stated	**11**	**139**	**2**	**3**	**4**	**1**	**5**	**1**	**3**	**1**	**2**	**1**	**3**	**1**	**1**	**2**

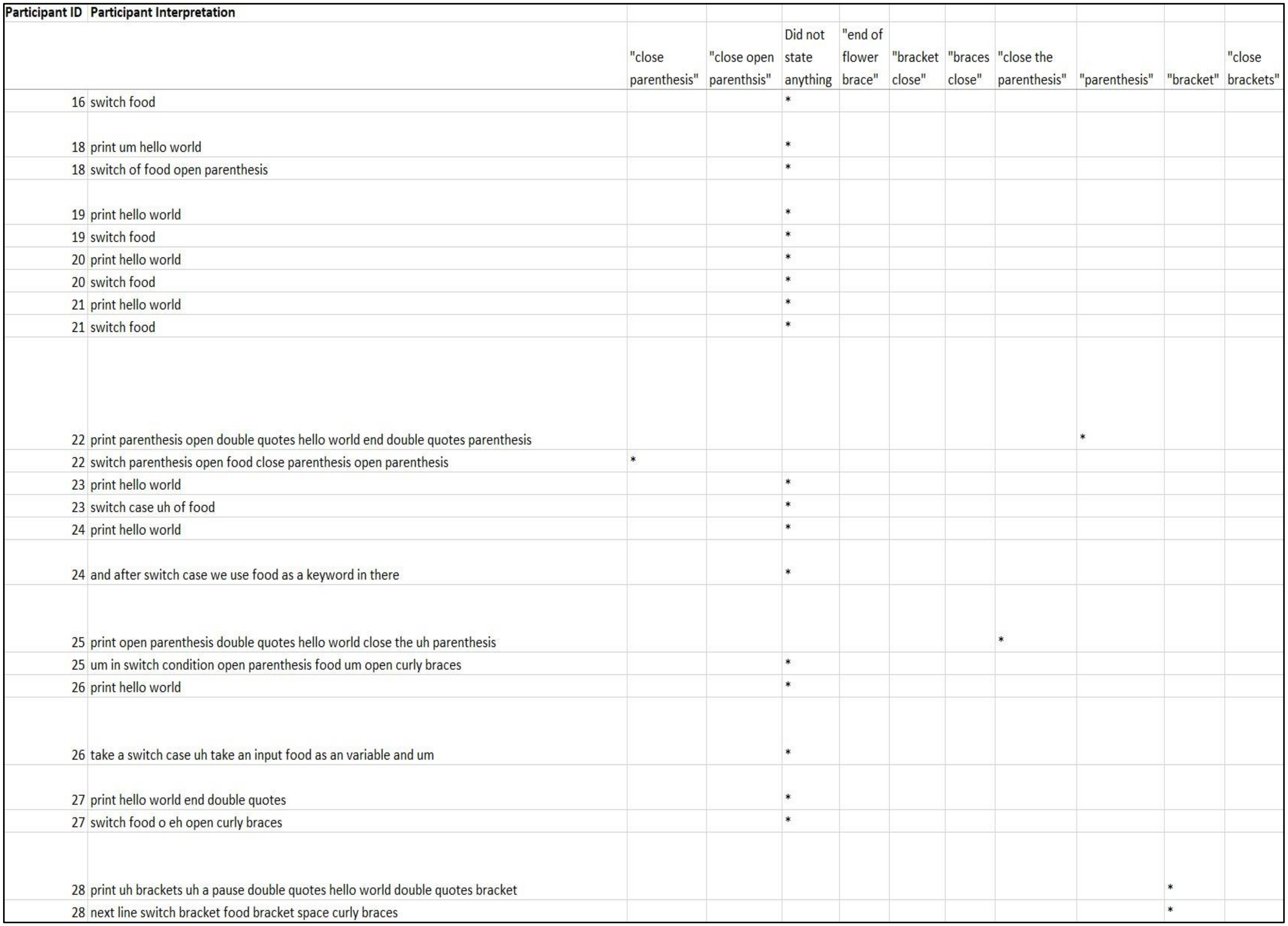
E.11))



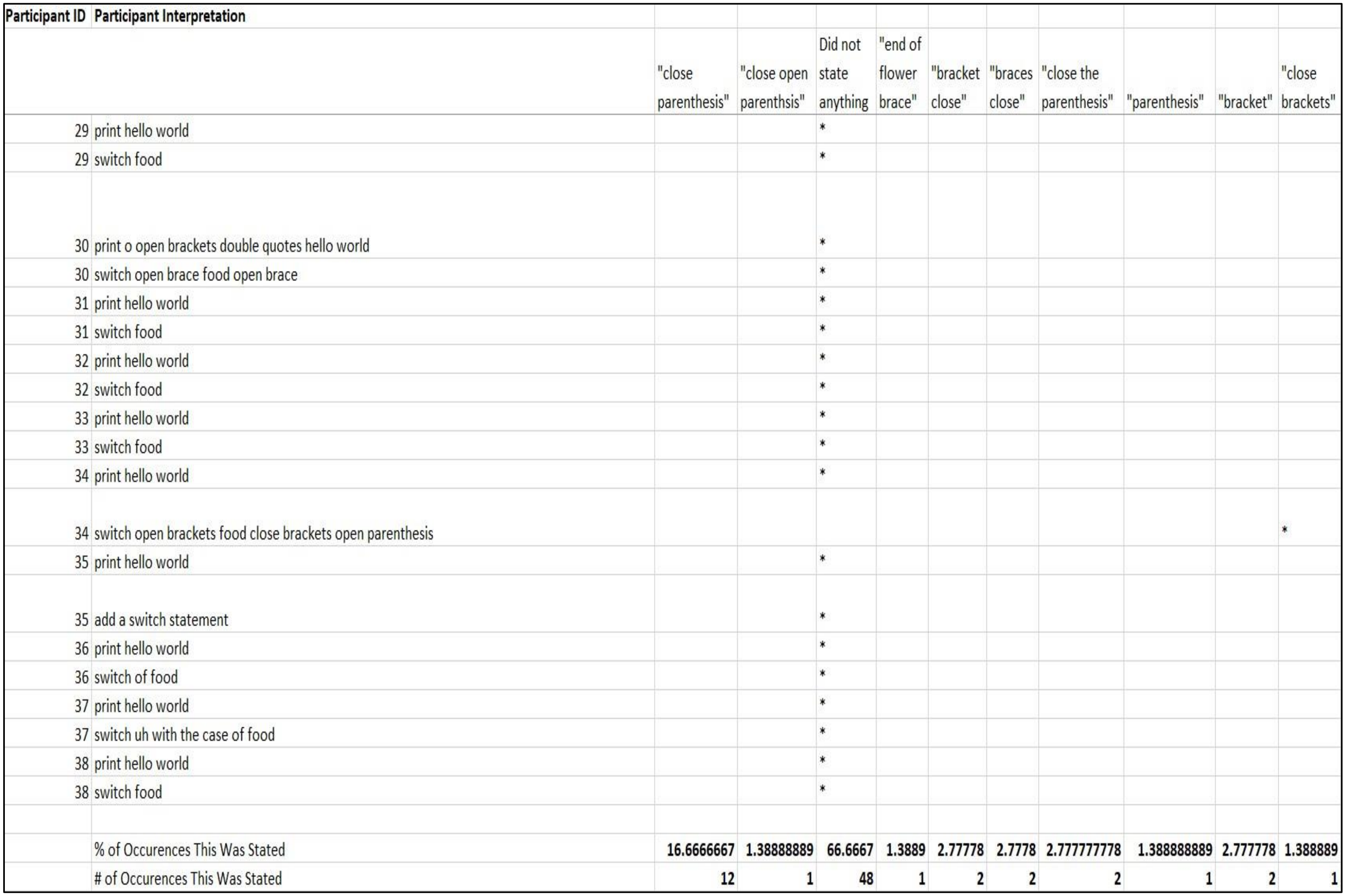
Participant ID	Participant Interpretation										
		"close parenthesis"	"close open parenthsis"	Did not state anything	"end of flower brace"	"bracket close"	"braces close"	"close the parenthesis"	"parenthesis"	"bracket"	"close brackets"
1	print open parenthesis double colon hello space world double quotes close parenthesis	*									
1	switch open parenthesis food close open parenthesis open flower bracket		*								
2	print hello world			*							
2	if that works uh switch food			*							
3	ah second line will be print hello world			*							
3	then second method switch food bracket is			*							
5	print parenthesis open parenthesis quotations hello world close quotations close parenthesis	*									
5	switch open parentheses food close parenthesis open bracket	*									
6	print f double quotes hello world double quotes close end of flower brace				*						
6	switch food flower brackets			*							
7	print bracket open hello world bracket close					*					
7	switch bracket open food bracket close flower bracket open					*					
8	move to the next line print braces open quotes open hello space world quotes close uh braces close						*				
8	switch braces open food braces close flower uh flower brackets open						*				



Participant ID	Participant Interpretation										
		"close parenthesis"	"close open parenthsis"	Did not state anything	"end of flower brace"	"bracket close"	"braces close"	"close the parenthesis"	"parenthesis"	"bracket"	"close brackets"
9	next line print open parenthesis double quotation hello space world double quotation close parenthesis	*									
9	next line switch open parenthesis food close parenthesis flower bracket	*									
10	print hello world uh			*							
10	switch uh food flower brack flower brackets open			*							
11	print open parenthesis end of end of quotations hello world close the parenthesis							*			
11	the next line is switch of food switch open parenthesis food close parenthesis open flower brackets	*									
12	print open parenthesis hello world end inverted quotes close parenthesis	*									
12	switch open parenthesis food close parenthesis open curly brace	*									
13	print open parenthesis double quotes hello world close double quotes close parenthesis	*									
13	switch open parenthesis food close parenthesis open curly brace	*									
14	print ah semi colon hello world close semi colon			*							
14	switch parenthesis food close parenthesis semi um what do you call this what ever you think um open parenthesis	*									
15	print hello world			*							
15	switch food flower bracket open			*							
16	print hello world			*							

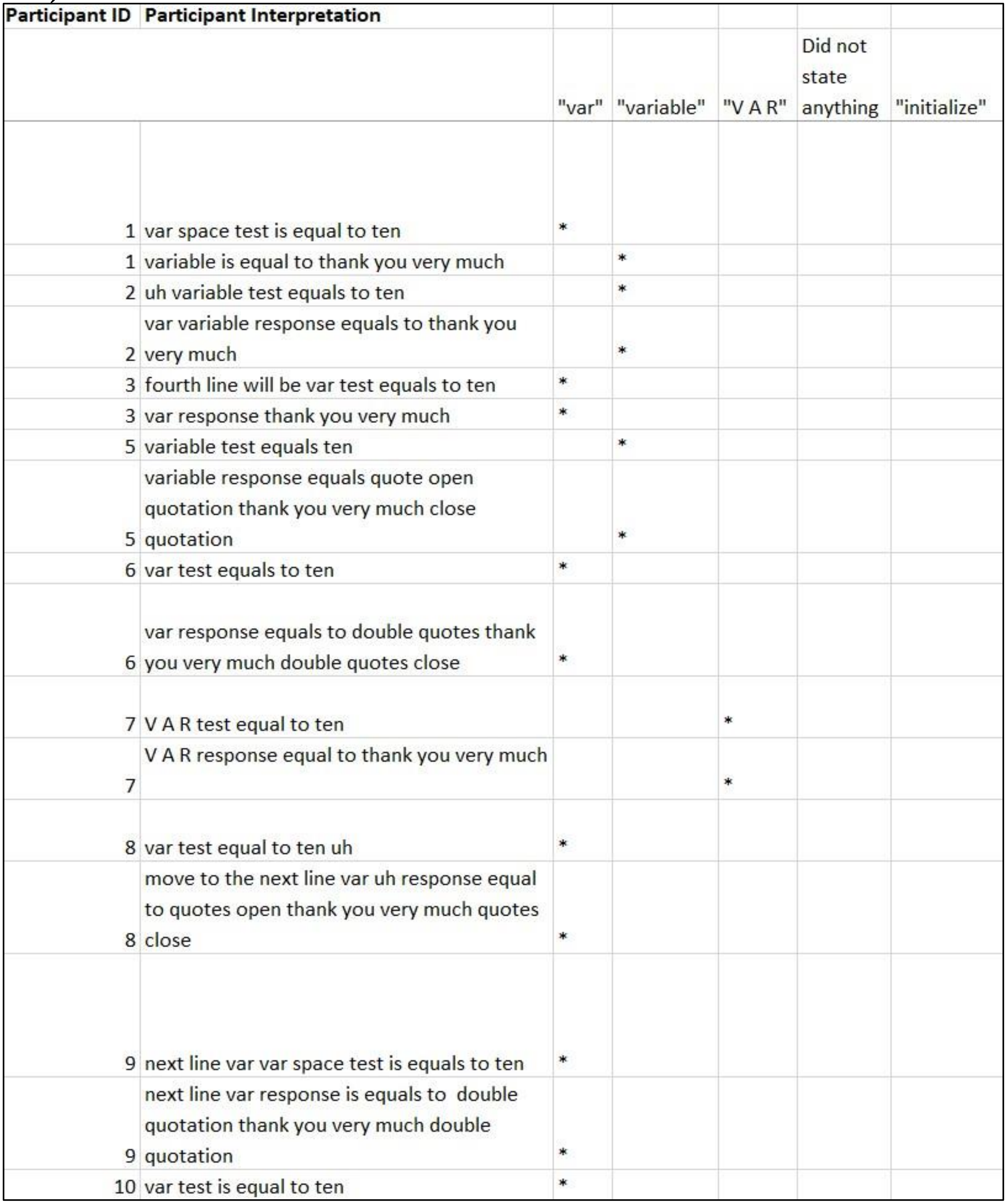


Participant ID	Participant Interpretation										
		"close parenthesis"	"close open parenthsis"	Did not state anything	"end of flower brace"	"bracket close"	"braces close"	"close the parenthesis"	"parenthesis"	"bracket"	"close brackets"
16	switch food			*							
18	print um hello world			*							
18	switch of food open parenthesis			*							
19	print hello world			*							
19	switch food			*							
20	print hello world			*							
20	switch food			*							
21	print hello world			*							
21	switch food			*							
22	print parenthesis open double quotes hello world end double quotes parenthesis								*		
22	switch parenthesis open food close parenthesis open parenthesis	*									
23	print hello world			*							
23	switch case uh of food			*							
24	print hello world			*							
24	and after switch case we use food as a keyword in there			*							
25	print open parenthesis double quotes hello world close the uh parenthesis							*			
25	um in switch condition open parenthesis food um open curly braces			*							
26	print hello world			*							
26	take a switch case uh take an input food as an variable and um			*							
27	print hello world end double quotes			*							
27	switch food o eh open curly braces			*							
28	print uh brackets uh a pause double quotes hello world double quotes bracket									*	
28	next line switch bracket food bracket space curly braces									*	

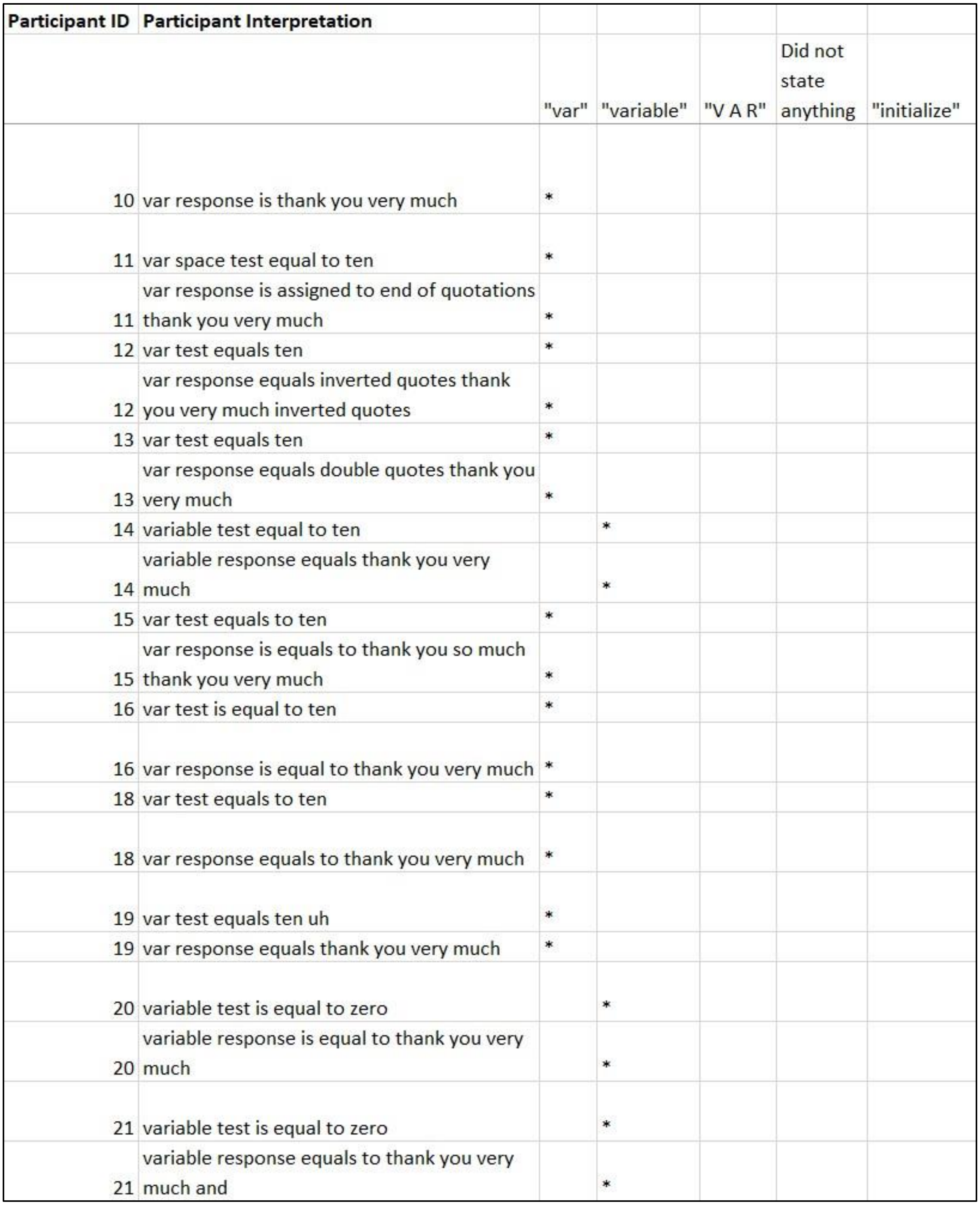


Participant ID	Participant Interpretation	"close parenthesis"	"close open parenthsis"	Did not state anything	"end of flower brace"	"bracket close"	"braces close"	"close the parenthesis"	"parenthesis"	"bracket"	"close brackets"
29	print hello world			*							
29	switch food			*							
30	print o open brackets double quotes hello world			*							
30	switch open brace food open brace			*							
31	print hello world			*							
31	switch food			*							
32	print hello world			*							
32	switch food			*							
33	print hello world			*							
33	switch food			*							
34	print hello world			*							
34	switch open brackets food close brackets open parenthesis										*
35	print hello world			*							
35	add a switch statement			*							
36	print hello world			*							
36	switch of food			*							
37	print hello world			*							
37	switch uh with the case of food			*							
38	print hello world			*							
38	switch food			*							
	% of Occurences This Was Stated	**16.6666667**	**1.38888889**	**66.6667**	**1.3889**	**2.77778**	**2.7778**	**2.777777778**	**1.388888889**	**2.777778**	**1.388889**
	# of Occurences This Was Stated	**12**	**1**	**48**	**1**	**2**	**2**	**2**	**1**	**2**	**1**

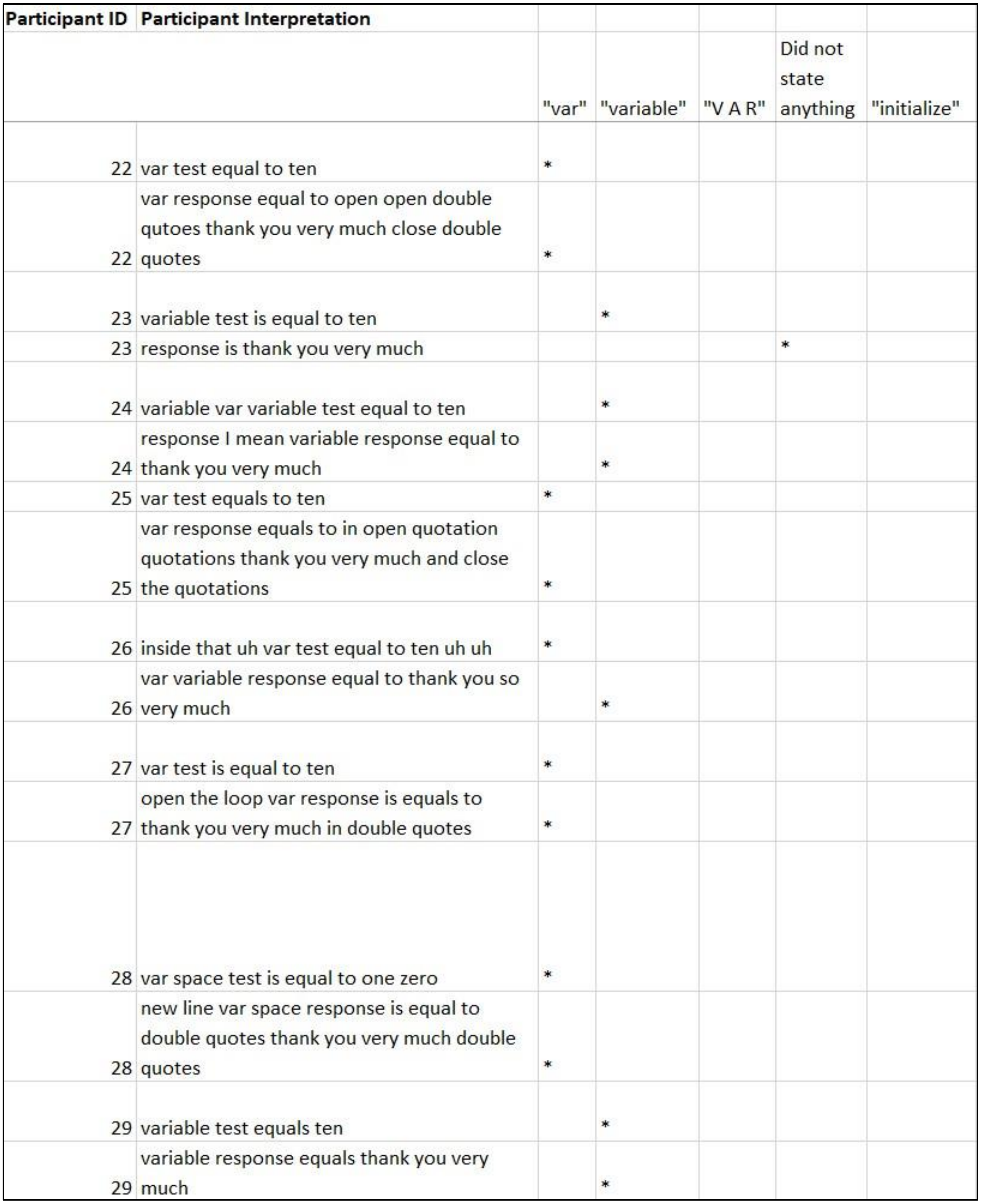
E.12) var



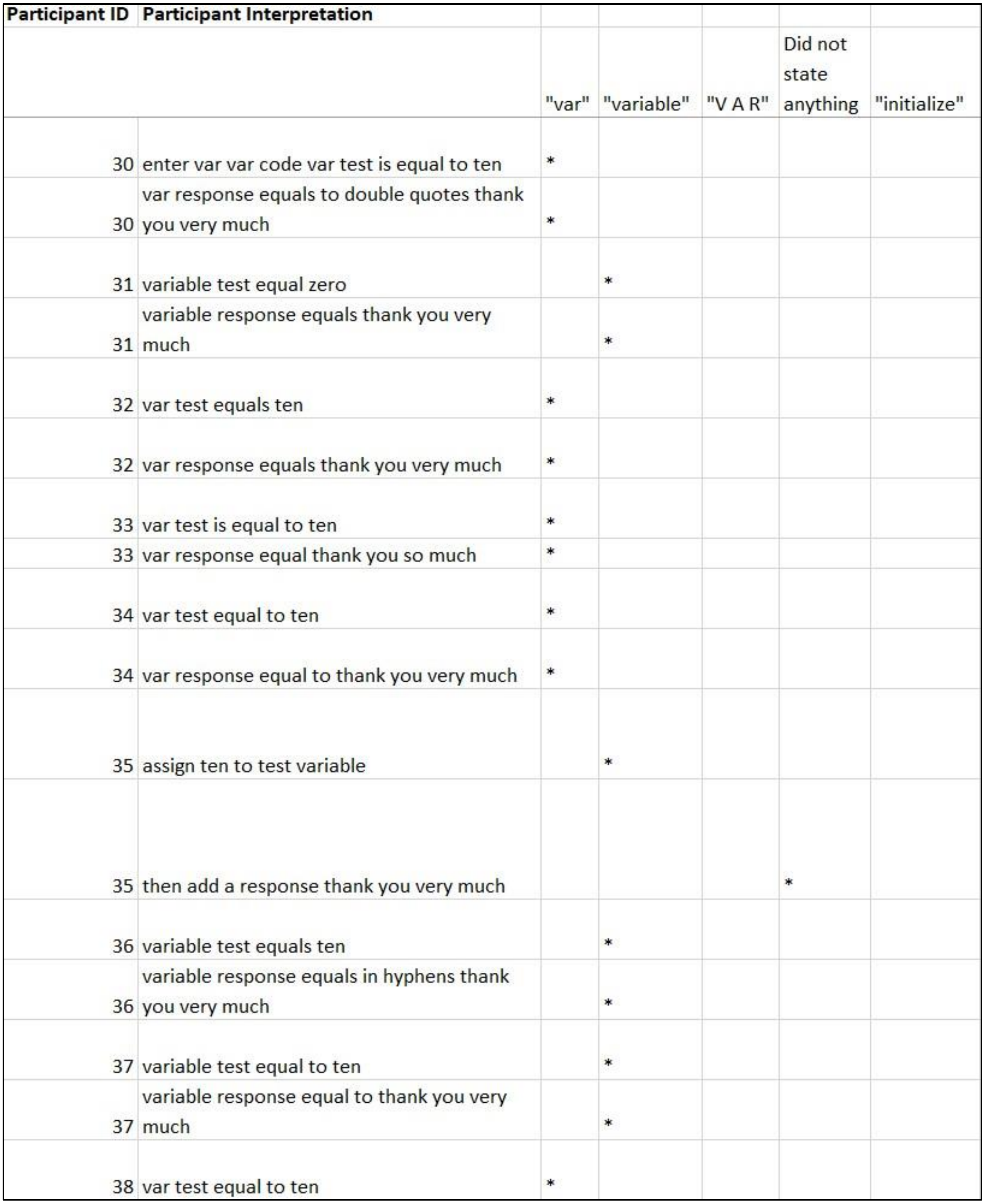
Participant ID	Participant Interpretation					
		"var"	"variable"	"V A R"	Did not state anything	"initialize"
1	var space test is equal to ten	*				
1	variable is equal to thank you very much		*			
2	uh variable test equals to ten		*			
2	var variable response equals to thank you very much		*			
3	fourth line will be var test equals to ten	*				
3	var response thank you very much	*				
5	variable test equals ten		*			
5	variable response equals quote open quotation thank you very much close quotation		*			
6	var test equals to ten	*				
6	var response equals to double quotes thank you very much double quotes close	*				
7	V A R test equal to ten			*		
7	V A R response equal to thank you very much			*		
8	var test equal to ten uh	*				
8	move to the next line var uh response equal to quotes open thank you very much quotes close	*				
9	next line var var space test is equals to ten	*				
9	next line var response is equals to double quotation thank you very much double quotation	*				
10	var test is equal to ten	*				



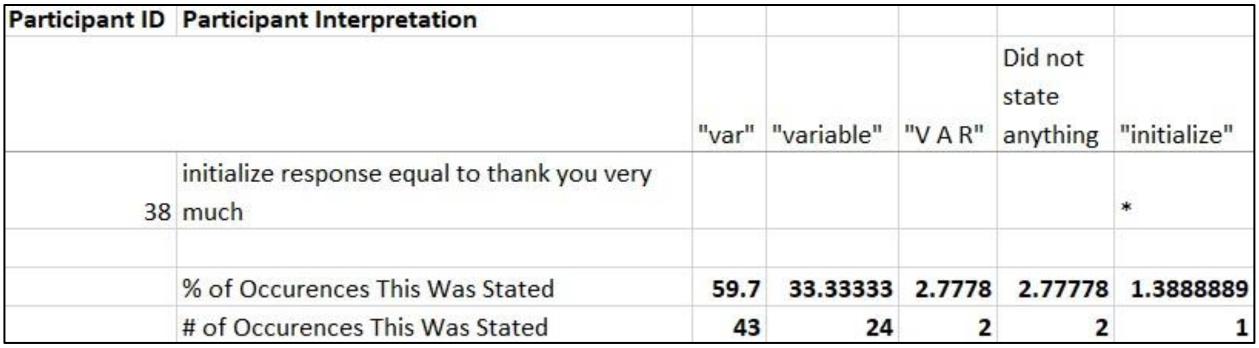
Participant ID	Participant Interpretation					
		"var"	"variable"	"V A R"	Did not state anything	"initialize"
10	var response is thank you very much	*				
11	var space test equal to ten	*				
11	var response is assigned to end of quotations thank you very much	*				
12	var test equals ten	*				
12	var response equals inverted quotes thank you very much inverted quotes	*				
13	var test equals ten	*				
13	var response equals double quotes thank you very much	*				
14	variable test equal to ten		*			
14	variable response equals thank you very much		*			
15	var test equals to ten	*				
15	var response is equals to thank you so much thank you very much	*				
16	var test is equal to ten	*				
16	var response is equal to thank you very much	*				
18	var test equals to ten	*				
18	var response equals to thank you very much	*				
19	var test equals ten uh	*				
19	var response equals thank you very much	*				
20	variable test is equal to zero		*			
20	variable response is equal to thank you very much		*			
21	variable test is equal to zero		*			
21	variable response equals to thank you very much and		*			



Participant ID	Participant Interpretation					
		"var"	"variable"	"V A R"	Did not state anything	"initialize"
22	var test equal to ten	*				
22	var response equal to open open double qutoes thank you very much close double quotes	*				
23	variable test is equal to ten		*			
23	response is thank you very much				*	
24	variable var variable test equal to ten		*			
24	response I mean variable response equal to thank you very much		*			
25	var test equals to ten	*				
25	var response equals to in open quotation quotations thank you very much and close the quotations	*				
26	inside that uh var test equal to ten uh uh	*				
26	var variable response equal to thank you so very much		*			
27	var test is equal to ten	*				
27	open the loop var response is equals to thank you very much in double quotes	*				
28	var space test is equal to one zero	*				
28	new line var space response is equal to double quotes thank you very much double quotes	*				
29	variable test equals ten		*			
29	variable response equals thank you very much		*			

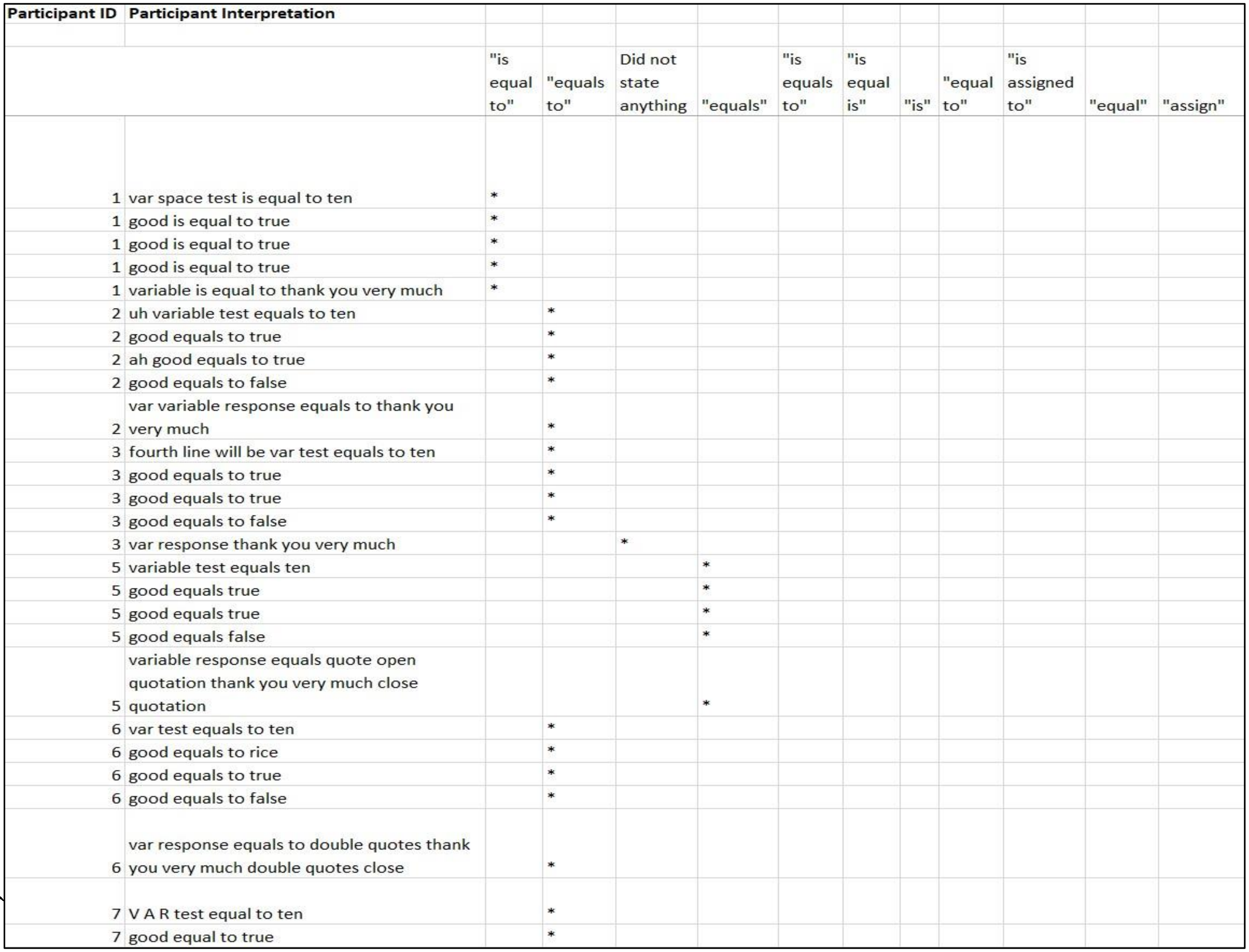


Participant ID	Participant Interpretation					
		"var"	"variable"	"V A R"	Did not state anything	"initialize"
30	enter var var code var test is equal to ten	*				
30	var response equals to double quotes thank you very much	*				
31	variable test equal zero		*			
31	variable response equals thank you very much		*			
32	var test equals ten	*				
32	var response equals thank you very much	*				
33	var test is equal to ten	*				
33	var response equal thank you so much	*				
34	var test equal to ten	*				
34	var response equal to thank you very much	*				
35	assign ten to test variable		*			
35	then add a response thank you very much				*	
36	variable test equals ten		*			
36	variable response equals in hyphens thank you very much		*			
37	variable test equal to ten		*			
37	variable response equal to thank you very much		*			
38	var test equal to ten	*				

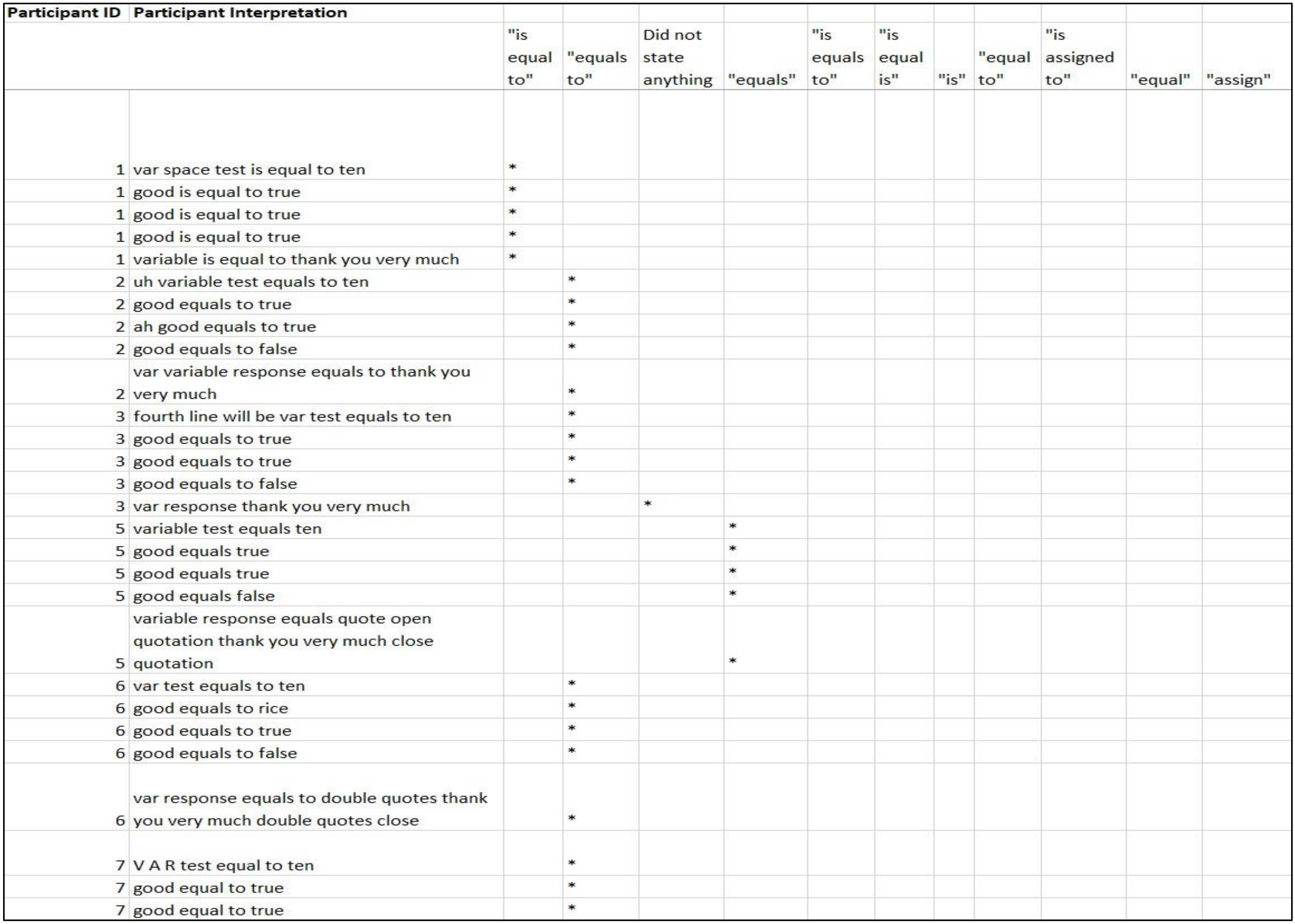


Participant ID	**Participant Interpretation**					
		"var"	"variable"	"V A R"	Did not state anything	"initialize"
38	initialize response equal to thank you very much					*
	% of Occurences This Was Stated	**59.7**	**33.33333**	**2.7778**	**2.77778**	**1.3888889**
	# of Occurences This Was Stated	**43**	**24**	**2**	**2**	**1**

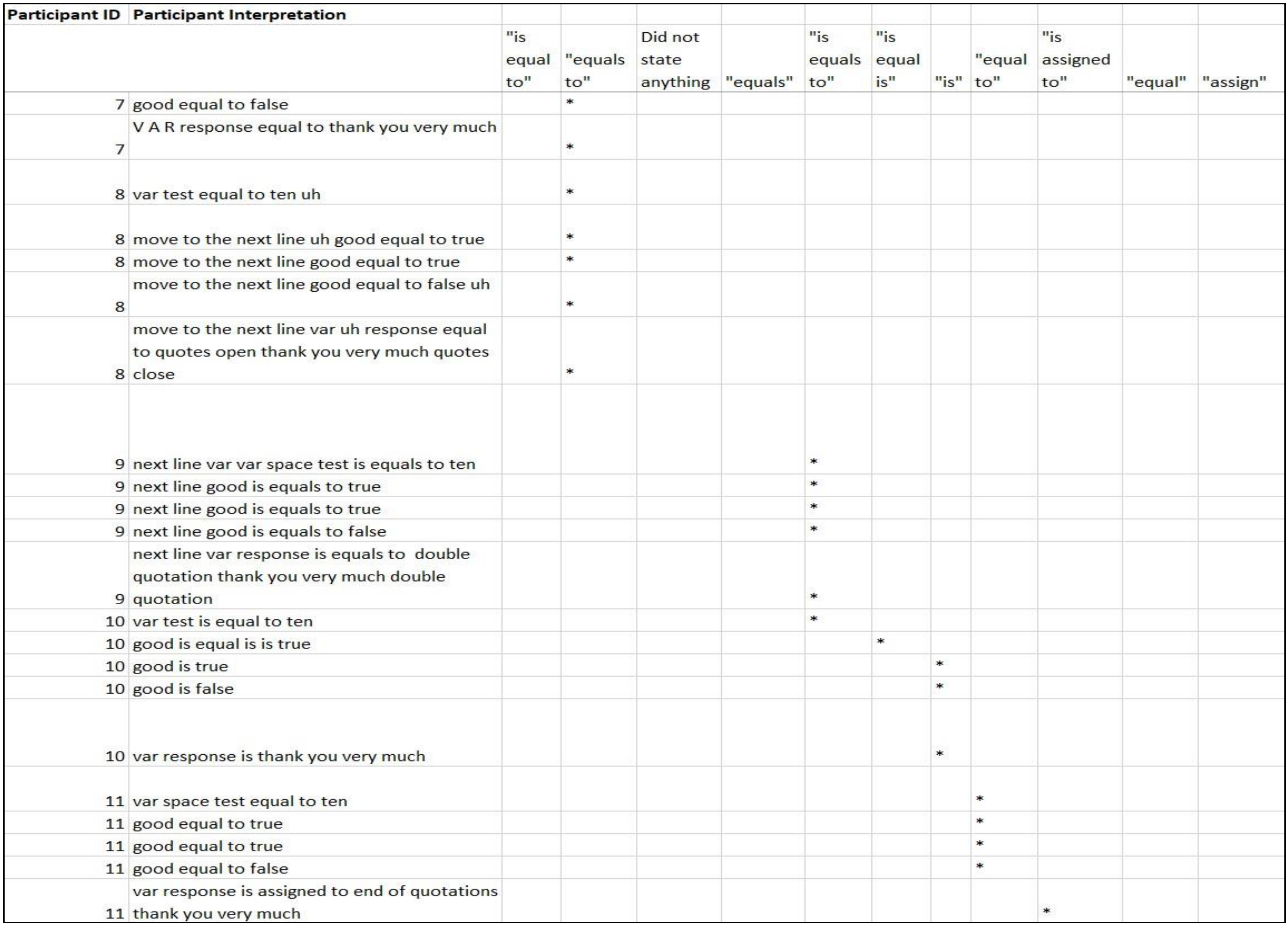
E.13) =



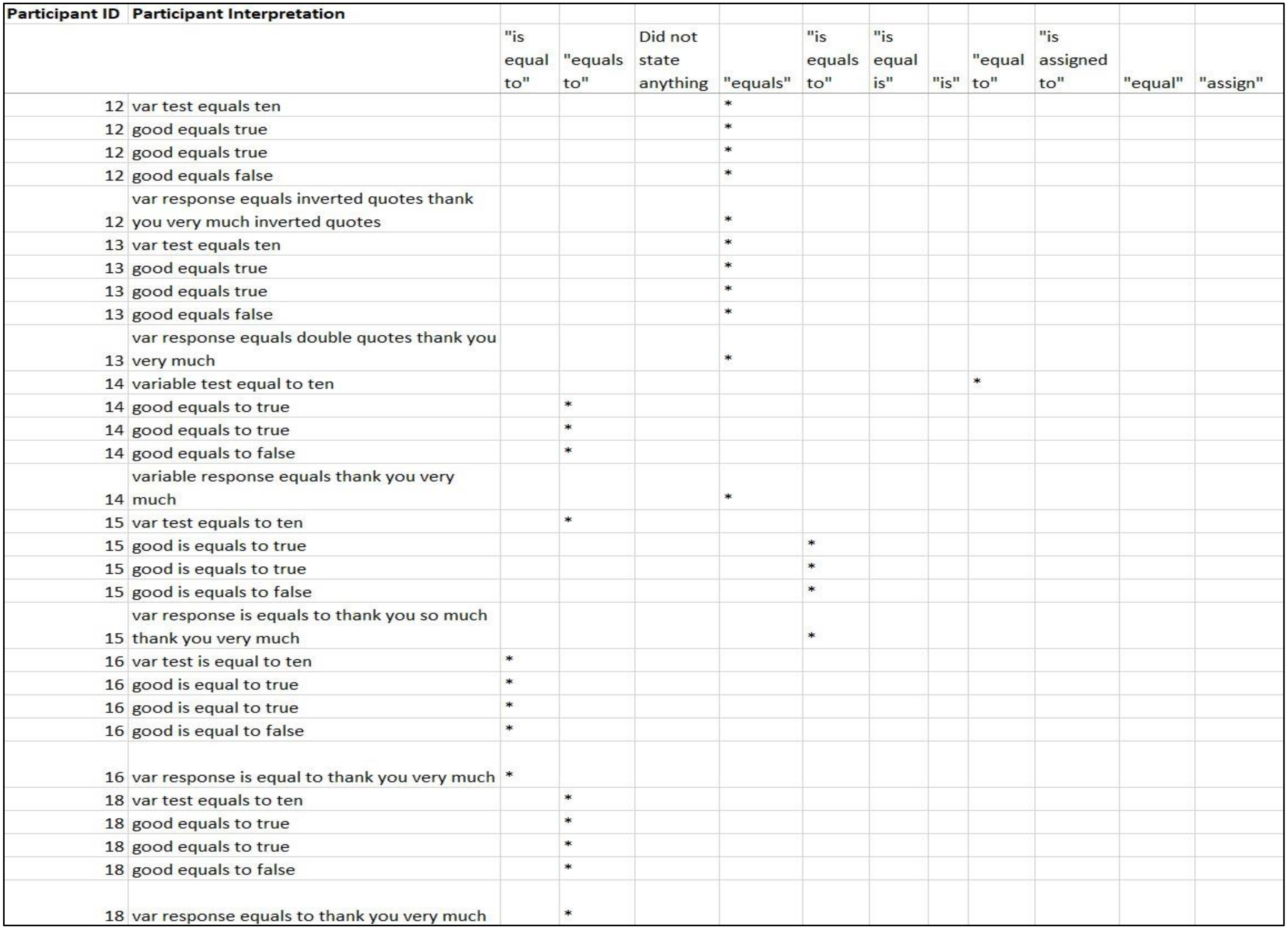
Participant ID	Participant Interpretation											
		"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
1	var space test is equal to ten	*										
1	good is equal to true	*										
1	good is equal to true	*										
1	good is equal to true	*										
1	variable is equal to thank you very much	*										
2	uh variable test equals to ten		*									
2	good equals to true		*									
2	ah good equals to true		*									
2	good equals to false		*									
2	var variable response equals to thank you very much		*									
3	fourth line will be var test equals to ten		*									
3	good equals to true		*									
3	good equals to true		*									
3	good equals to false		*									
3	var response thank you very much			*								
5	variable test equals ten				*							
5	good equals true				*							
5	good equals true				*							
5	good equals false				*							
5	variable response equals quote open quotation thank you very much close quotation				*							
6	var test equals to ten		*									
6	good equals to rice		*									
6	good equals to true		*									
6	good equals to false		*									
6	var response equals to double quotes thank you very much double quotes close		*									
7	V A R test equal to ten		*									
7	good equal to true		*									



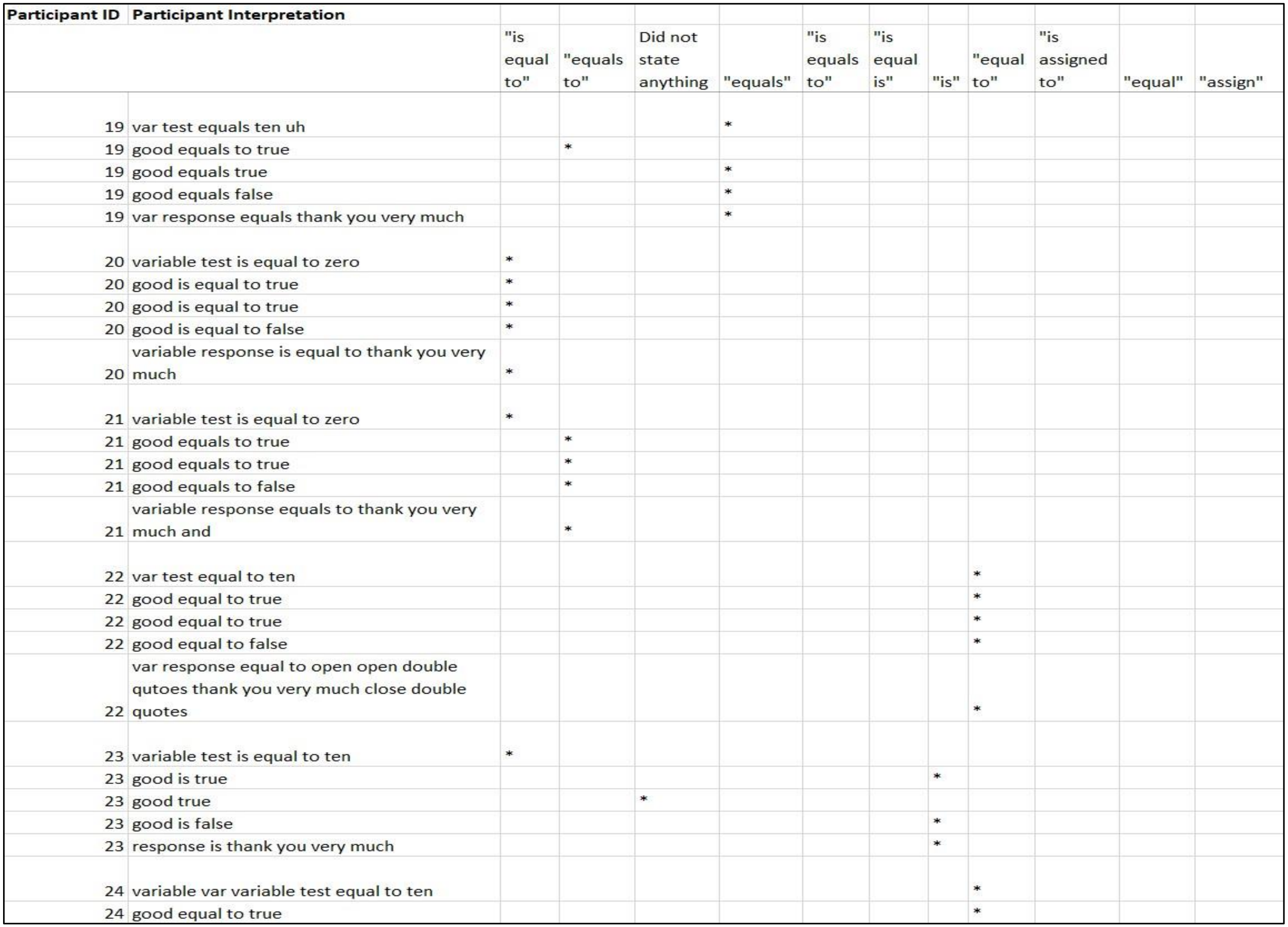
Participant ID	Participant Interpretation											
		"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
1	var space test is equal to ten	*										
1	good is equal to true	*										
1	good is equal to true	*										
1	good is equal to true	*										
1	variable is equal to thank you very much	*										
2	uh variable test equals to ten		*									
2	good equals to true		*									
2	ah good equals to true		*									
2	good equals to false		*									
2	var variable response equals to thank you very much		*									
3	fourth line will be var test equals to ten		*									
3	good equals to true		*									
3	good equals to true		*									
3	good equals to false		*									
3	var response thank you very much			*								
5	variable test equals ten				*							
5	good equals true				*							
5	good equals true				*							
5	good equals false				*							
5	variable response equals quote open quotation thank you very much close quotation				*							
6	var test equals to ten		*									
6	good equals to rice		*									
6	good equals to true		*									
6	good equals to false		*									
6	var response equals to double quotes thank you very much double quotes close		*									
7	V A R test equal to ten		*									
7	good equal to true		*									
7	good equal to true		*									



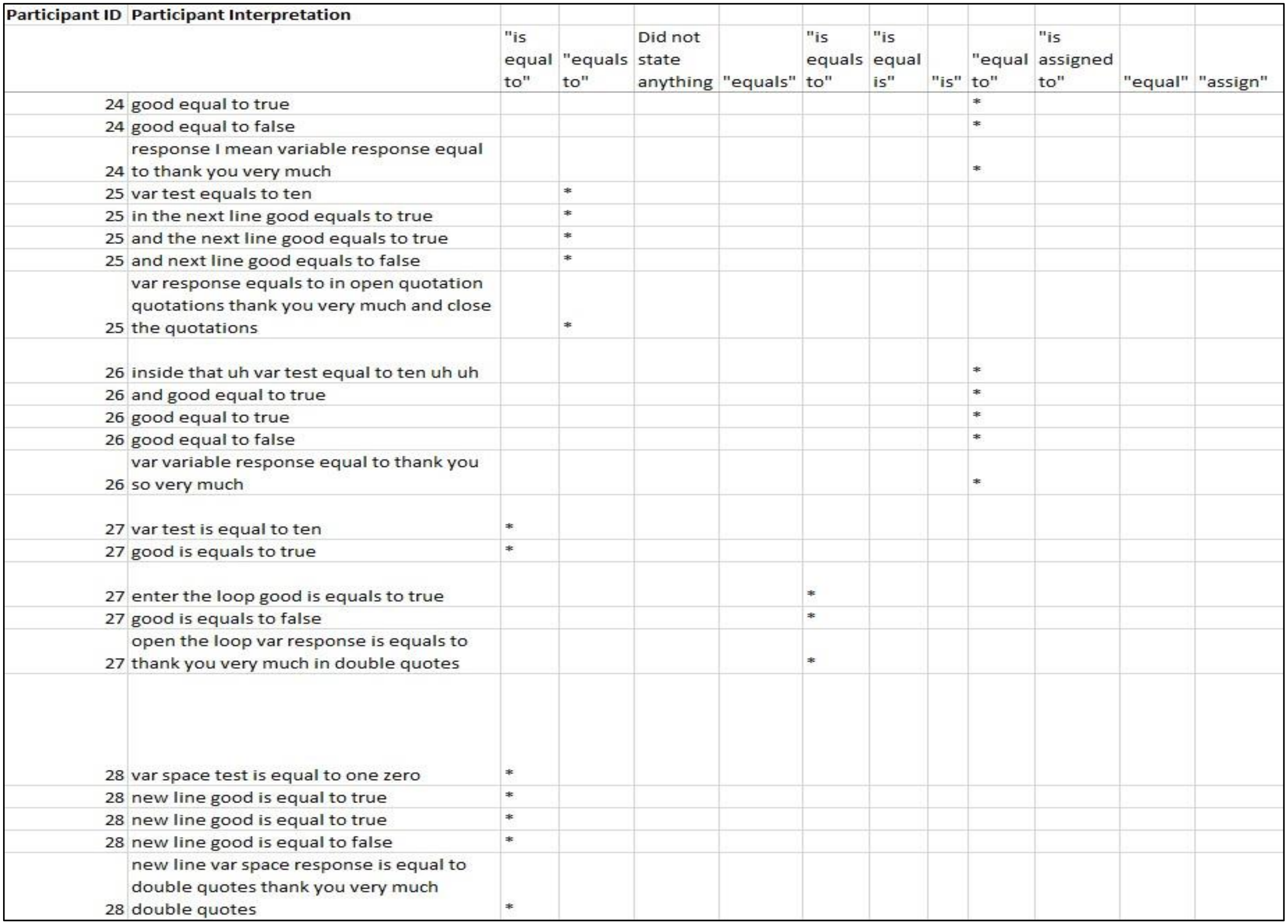
Participant ID	Participant Interpretation	"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
7	good equal to false		*									
7	V A R response equal to thank you very much		*									
8	var test equal to ten uh		*									
8	move to the next line uh good equal to true		*									
8	move to the next line good equal to true		*									
8	move to the next line good equal to false uh		*									
8	move to the next line var uh response equal to quotes open thank you very much quotes close		*									
9	next line var var space test is equals to ten					*						
9	next line good is equals to true					*						
9	next line good is equals to true					*						
9	next line good is equals to false					*						
9	next line var response is equals to double quotation thank you very much double quotation					*						
10	var test is equal to ten					*						
10	good is equal is is true						*					
10	good is true							*				
10	good is false							*				
10	var response is thank you very much							*				
11	var space test equal to ten								*			
11	good equal to true								*			
11	good equal to true								*			
11	good equal to false								*			
11	var response is assigned to end of quotations thank you very much									*		



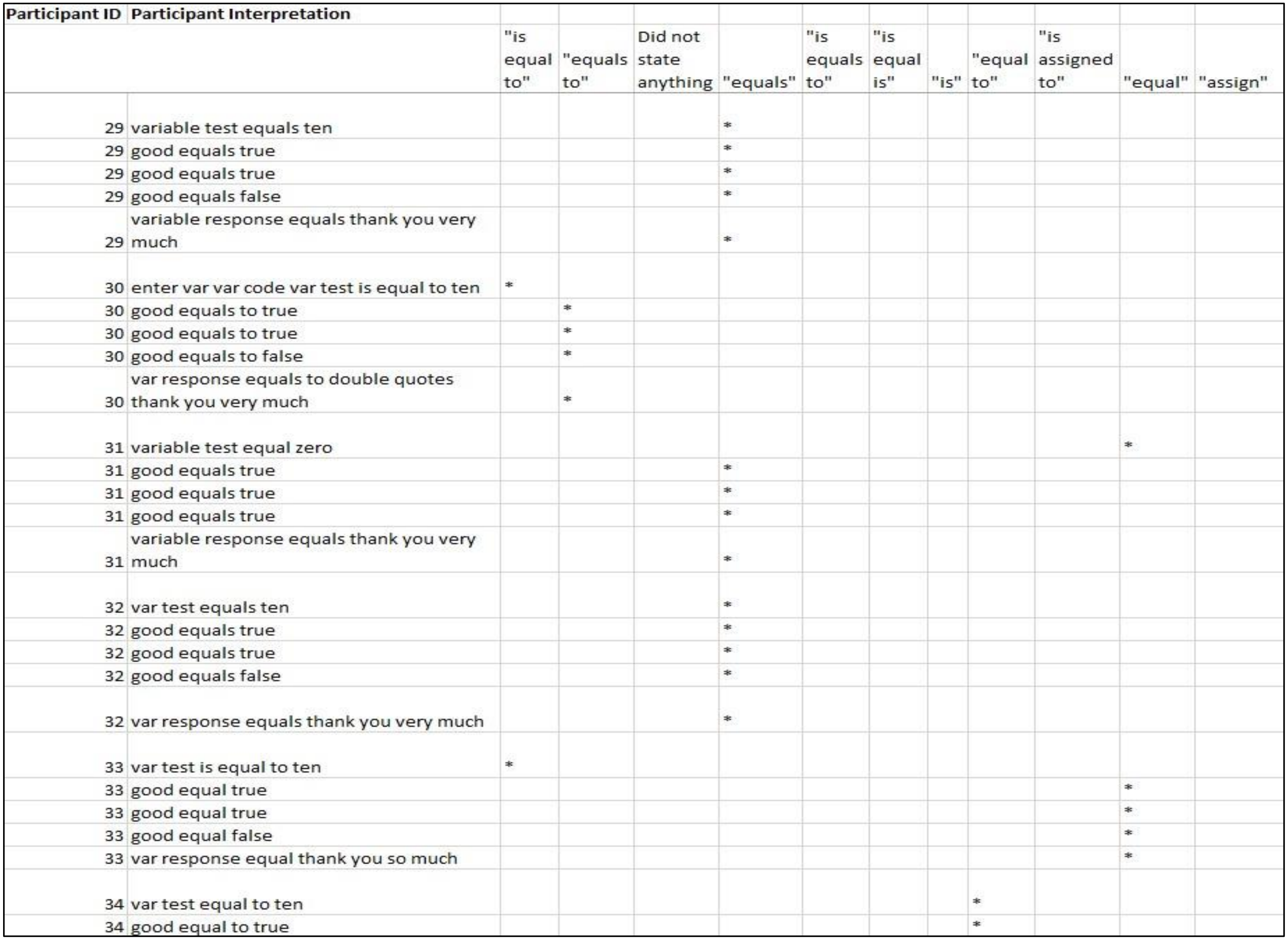
Participant ID	Participant Interpretation											
		"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
12	var test equals ten				*							
12	good equals true				*							
12	good equals true				*							
12	good equals false				*							
12	var response equals inverted quotes thank you very much inverted quotes				*							
13	var test equals ten				*							
13	good equals true				*							
13	good equals true				*							
13	good equals false				*							
13	var response equals double quotes thank you very much				*							
14	variable test equal to ten								*			
14	good equals to true		*									
14	good equals to true		*									
14	good equals to false		*									
14	variable response equals thank you very much				*							
15	var test equals to ten		*									
15	good is equals to true					*						
15	good is equals to true					*						
15	good is equals to false					*						
15	var response is equals to thank you so much thank you very much					*						
16	var test is equal to ten	*										
16	good is equal to true	*										
16	good is equal to true	*										
16	good is equal to false	*										
16	var response is equal to thank you very much	*										
18	var test equals to ten		*									
18	good equals to true		*									
18	good equals to true		*									
18	good equals to false		*									
18	var response equals to thank you very much		*									



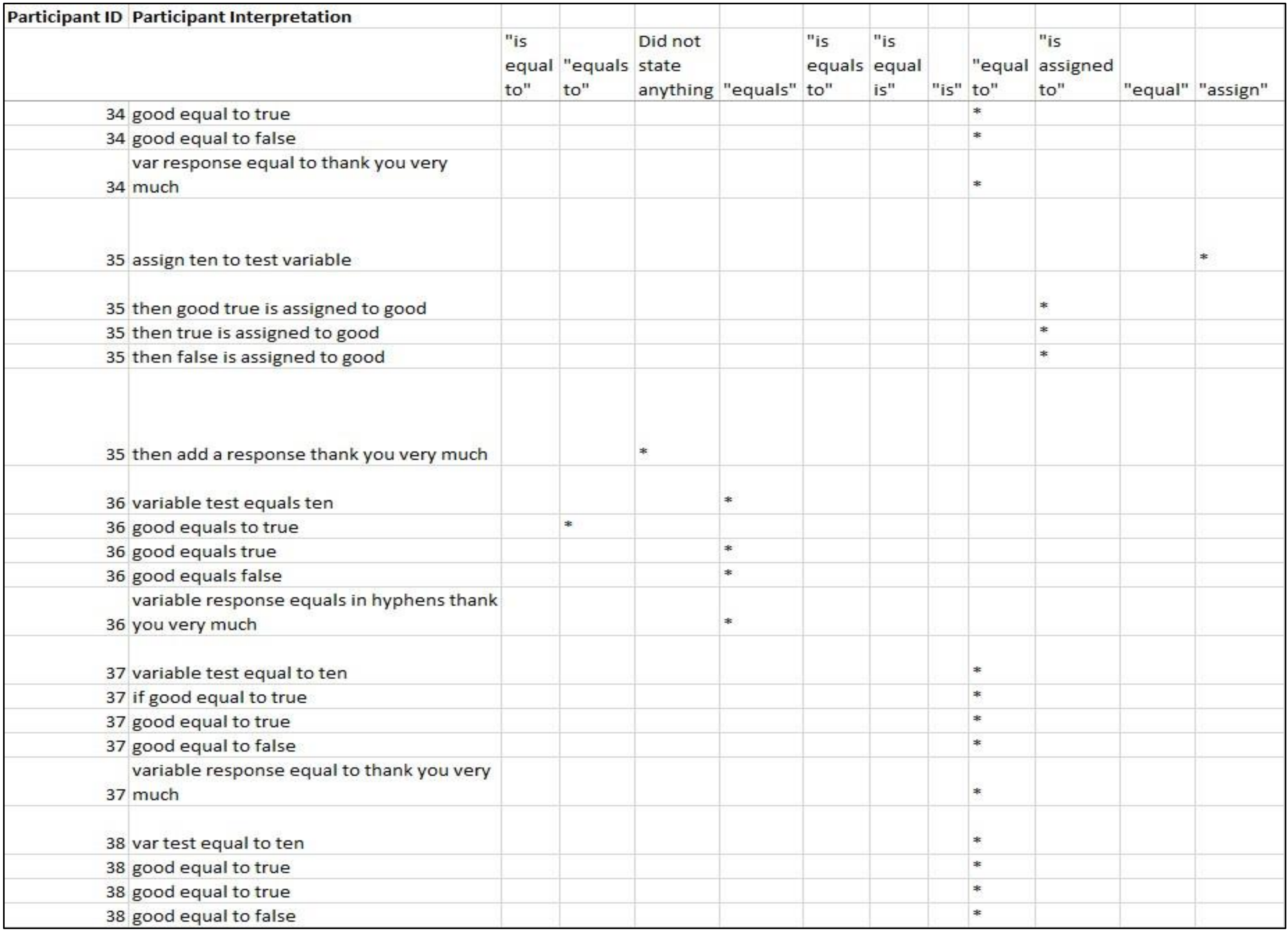
Participant ID	Participant Interpretation											
		"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
19	var test equals ten uh				*							
19	good equals to true		*									
19	good equals true				*							
19	good equals false				*							
19	var response equals thank you very much				*							
20	variable test is equal to zero	*										
20	good is equal to true	*										
20	good is equal to true	*										
20	good is equal to false	*										
20	variable response is equal to thank you very much	*										
21	variable test is equal to zero	*										
21	good equals to true		*									
21	good equals to true		*									
21	good equals to false		*									
21	variable response equals to thank you very much and		*									
22	var test equal to ten								*			
22	good equal to true								*			
22	good equal to true								*			
22	good equal to false								*			
22	var response equal to open open double qutoes thank you very much close double quotes								*			
23	variable test is equal to ten	*										
23	good is true							*				
23	good true			*								
23	good is false							*				
23	response is thank you very much							*				
24	variable var variable test equal to ten								*			
24	good equal to true								*			



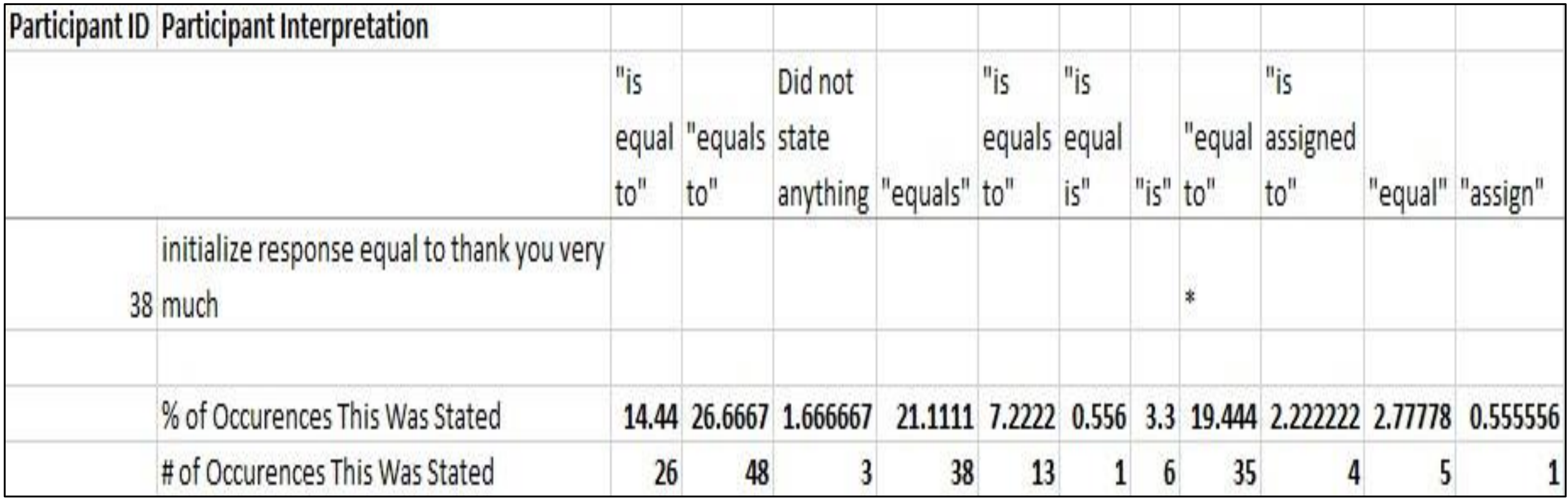
Participant ID	Participant Interpretation											
		"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
24	good equal to true								*			
24	good equal to false								*			
24	response I mean variable response equal to thank you very much								*			
25	var test equals to ten		*									
25	in the next line good equals to true		*									
25	and the next line good equals to true		*									
25	and next line good equals to false		*									
25	var response equals to in open quotation quotations thank you very much and close the quotations		*									
26	inside that uh var test equal to ten uh uh								*			
26	and good equal to true								*			
26	good equal to true								*			
26	good equal to false								*			
26	var variable response equal to thank you so very much								*			
27	var test is equal to ten	*										
27	good is equals to true	*										
27	enter the loop good is equals to true					*						
27	good is equals to false					*						
27	open the loop var response is equals to thank you very much in double quotes					*						
28	var space test is equal to one zero	*										
28	new line good is equal to true	*										
28	new line good is equal to true	*										
28	new line good is equal to false	*										
28	new line var space response is equal to double quotes thank you very much double quotes	*										



Participant ID	Participant Interpretation	"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
29	variable test equals ten				*							
29	good equals true				*							
29	good equals true				*							
29	good equals false				*							
29	variable response equals thank you very much				*							
30	enter var var code var test is equal to ten	*										
30	good equals to true		*									
30	good equals to true		*									
30	good equals to false		*									
30	var response equals to double quotes thank you very much		*									
31	variable test equal zero										*	
31	good equals true				*							
31	good equals true				*							
31	good equals true				*							
31	variable response equals thank you very much				*							
32	var test equals ten				*							
32	good equals true				*							
32	good equals true				*							
32	good equals false				*							
32	var response equals thank you very much				*							
33	var test is equal to ten	*										
33	good equal true										*	
33	good equal true										*	
33	good equal false										*	
33	var response equal thank you so much										*	
34	var test equal to ten								*			
34	good equal to true								*			

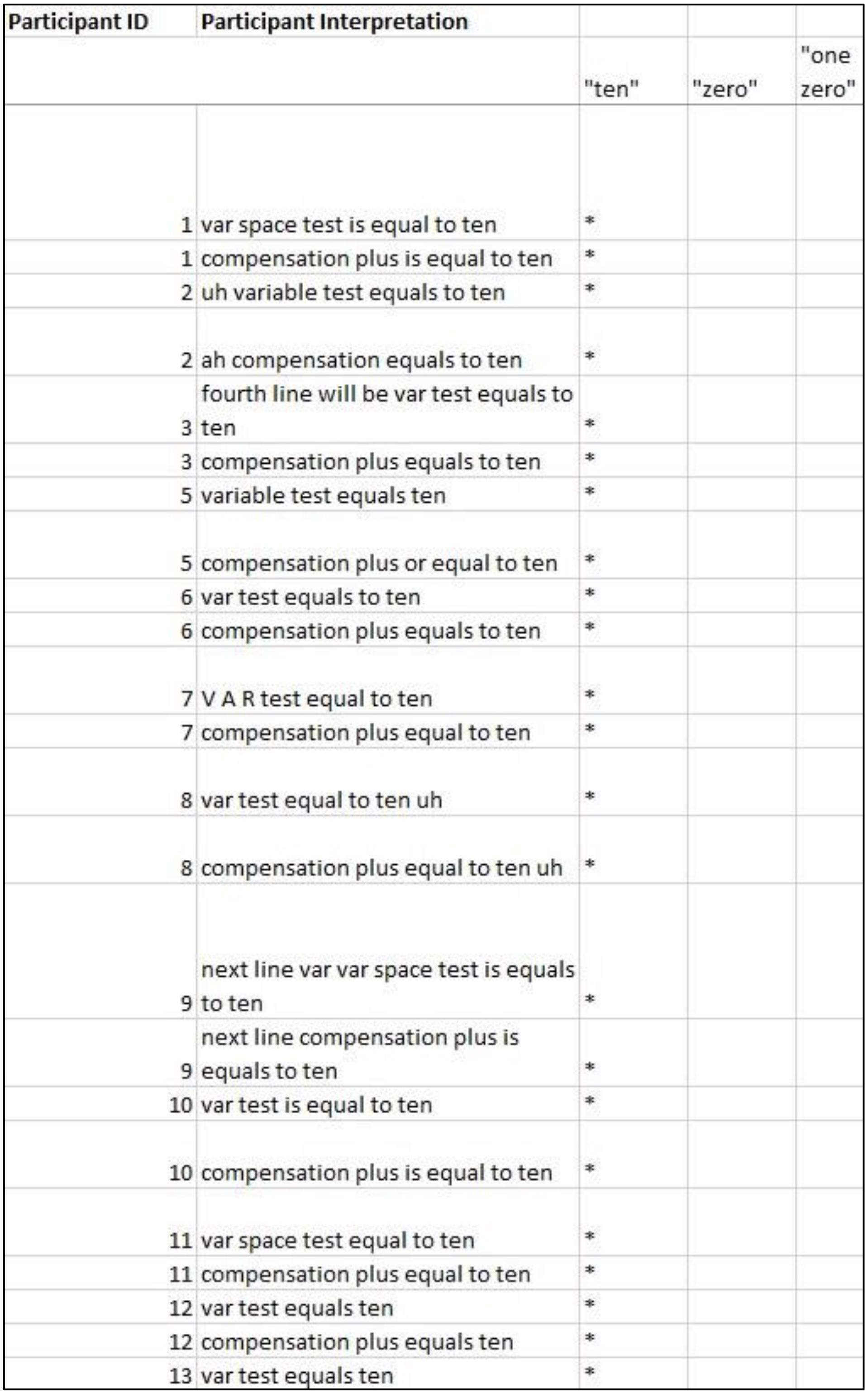


Participant ID	Participant Interpretation											
		"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
34	good equal to true								*			
34	good equal to false								*			
34	var response equal to thank you very much								*			
35	assign ten to test variable											*
35	then good true is assigned to good									*		
35	then true is assigned to good									*		
35	then false is assigned to good									*		
35	then add a response thank you very much			*								
36	variable test equals ten				*							
36	good equals to true		*									
36	good equals true				*							
36	good equals false				*							
36	variable response equals in hyphens thank you very much				*							
37	variable test equal to ten								*			
37	if good equal to true								*			
37	good equal to true								*			
37	good equal to false								*			
37	variable response equal to thank you very much								*			
38	var test equal to ten								*			
38	good equal to true								*			
38	good equal to true								*			
38	good equal to false								*			

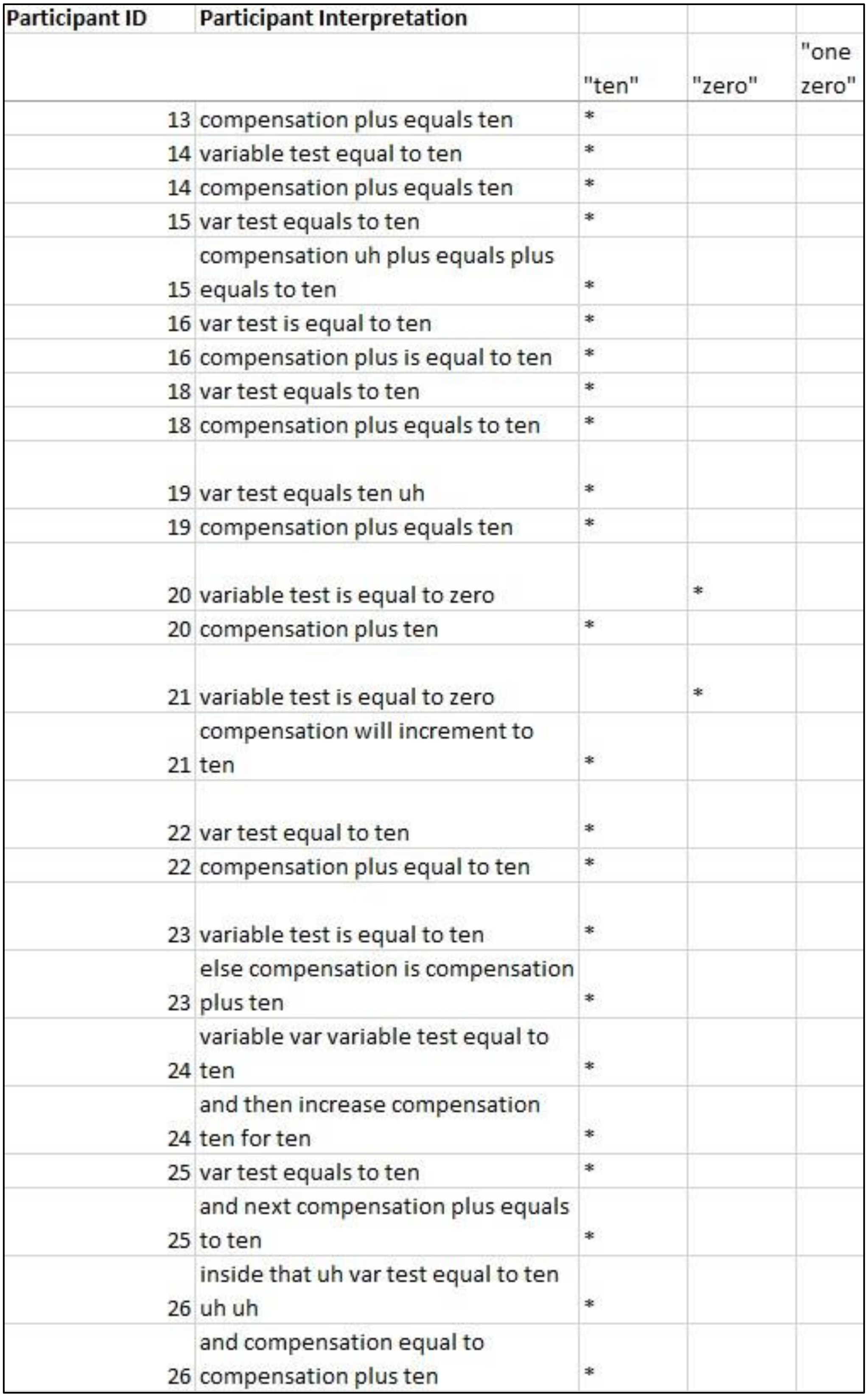


Participant ID	Participant Interpretation											
		"is equal to"	"equals to"	Did not state anything	"equals"	"is equals to"	"is equal is"	"is"	"equal to"	"is assigned to"	"equal"	"assign"
38	initialize response equal to thank you very much								*			
	% of Occurences This Was Stated	14.44	26.6667	1.666667	21.1111	7.2222	0.556	3.3	19.444	2.222222	2.77778	0.555556
	# of Occurences This Was Stated	26	48	3	38	13	1	6	35	4	5	1

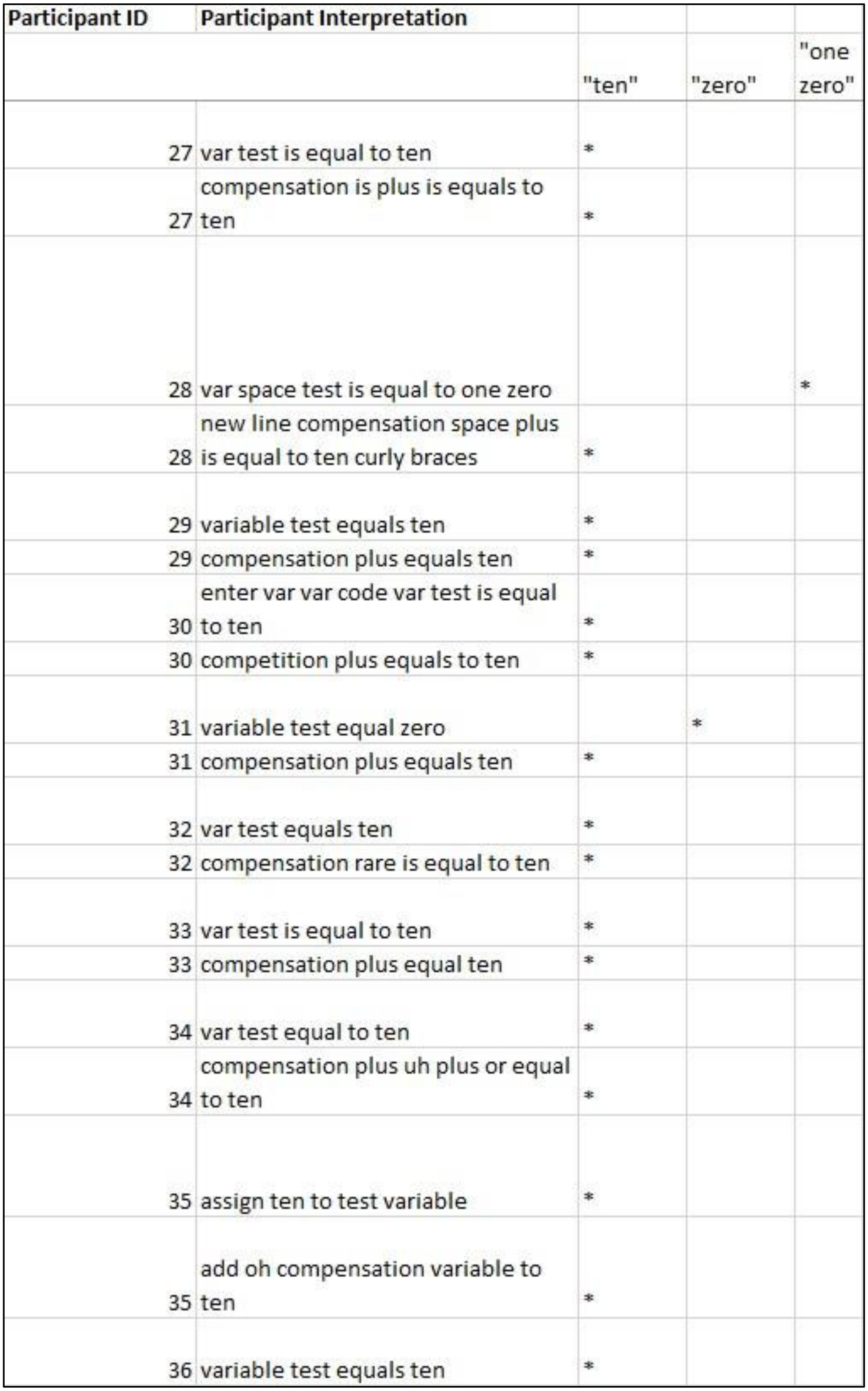
E.14) 10



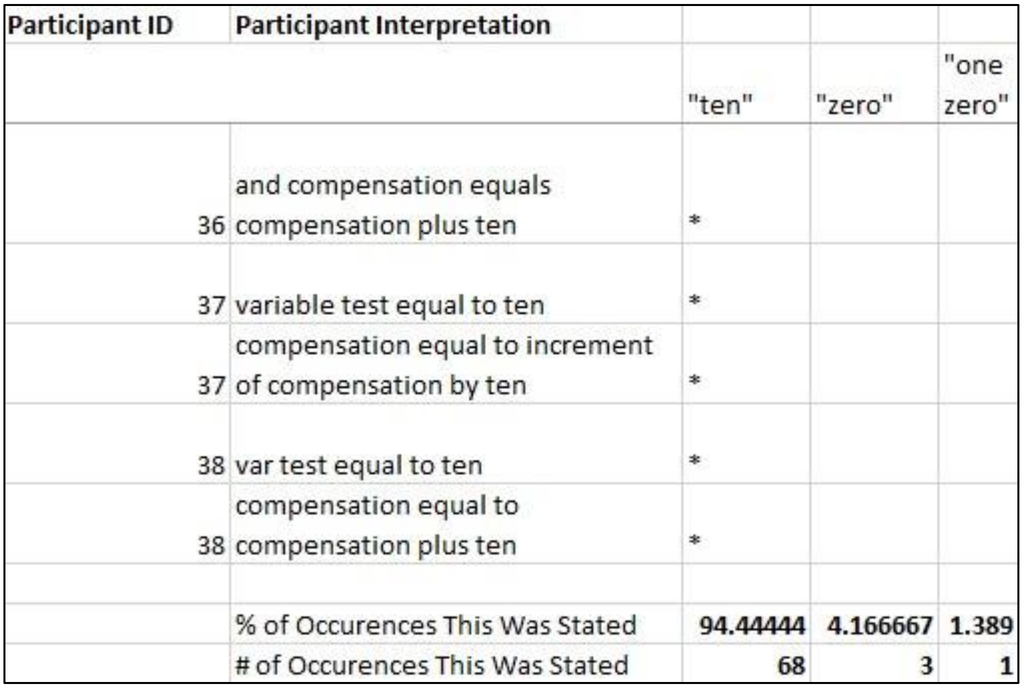
Participant ID	Participant Interpretation			
		"ten"	"zero"	"one zero"
1	var space test is equal to ten	*		
1	compensation plus is equal to ten	*		
2	uh variable test equals to ten	*		
2	ah compensation equals to ten	*		
3	fourth line will be var test equals to ten	*		
3	compensation plus equals to ten	*		
5	variable test equals ten	*		
5	compensation plus or equal to ten	*		
6	var test equals to ten	*		
6	compensation plus equals to ten	*		
7	V A R test equal to ten	*		
7	compensation plus equal to ten	*		
8	var test equal to ten uh	*		
8	compensation plus equal to ten uh	*		
9	next line var var space test is equals to ten	*		
9	next line compensation plus is equals to ten	*		
10	var test is equal to ten	*		
10	compensation plus is equal to ten	*		
11	var space test equal to ten	*		
11	compensation plus equal to ten	*		
12	var test equals ten	*		
12	compensation plus equals ten	*		
13	var test equals ten	*		



Participant ID	Participant Interpretation			
		"ten"	"zero"	"one zero"
13	compensation plus equals ten	*		
14	variable test equal to ten	*		
14	compensation plus equals ten	*		
15	var test equals to ten	*		
15	compensation uh plus equals plus equals to ten	*		
16	var test is equal to ten	*		
16	compensation plus is equal to ten	*		
18	var test equals to ten	*		
18	compensation plus equals to ten	*		
19	var test equals ten uh	*		
19	compensation plus equals ten	*		
20	variable test is equal to zero		*	
20	compensation plus ten	*		
21	variable test is equal to zero		*	
21	compensation will increment to ten	*		
22	var test equal to ten	*		
22	compensation plus equal to ten	*		
23	variable test is equal to ten	*		
23	else compensation is compensation plus ten	*		
24	variable var variable test equal to ten	*		
24	and then increase compensation ten for ten	*		
25	var test equals to ten	*		
25	and next compensation plus equals to ten	*		
26	inside that uh var test equal to ten uh uh	*		
26	and compensation equal to compensation plus ten	*		

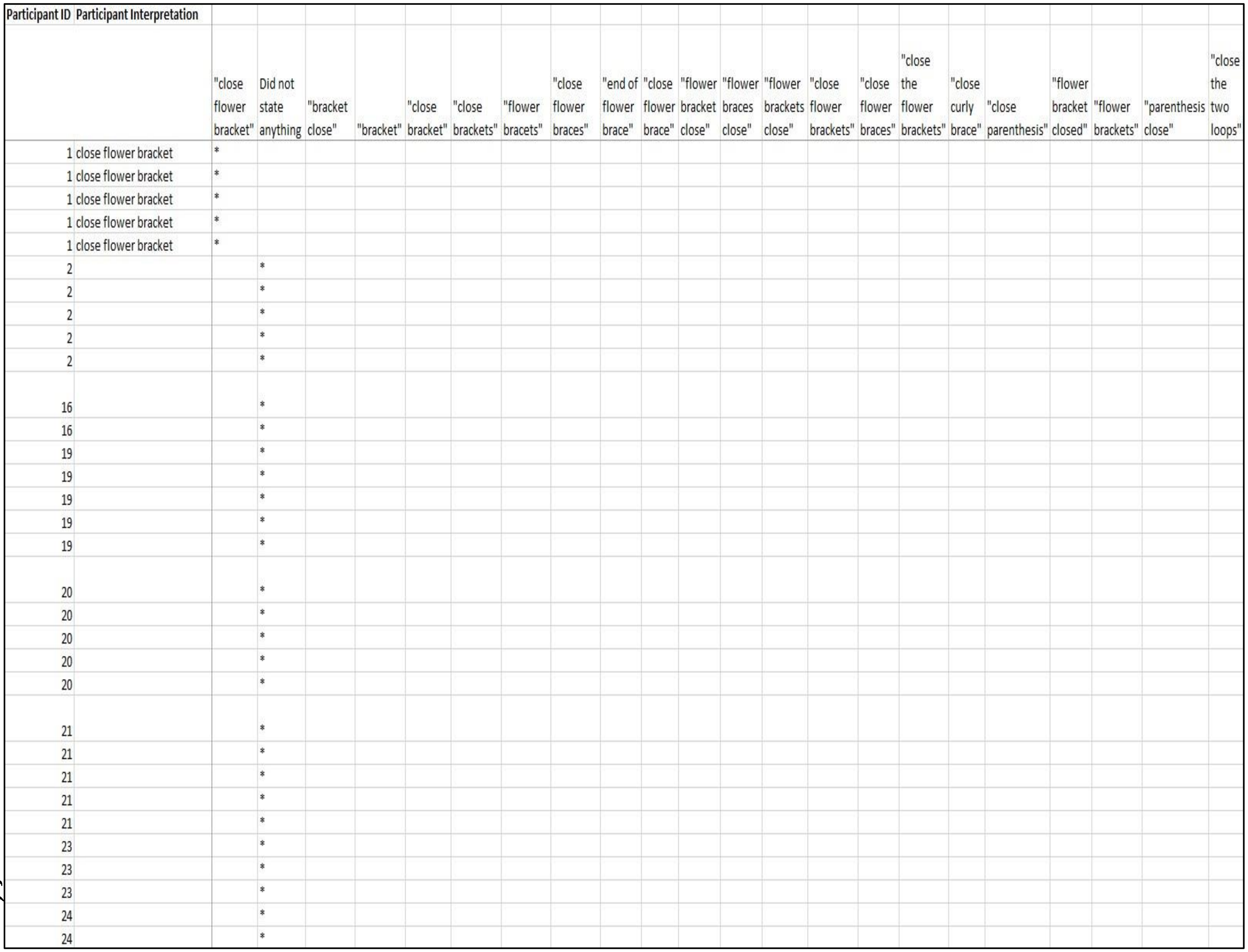


Participant ID	Participant Interpretation			
		"ten"	"zero"	"one zero"
27	var test is equal to ten	*		
27	compensation is plus is equals to ten	*		
28	var space test is equal to one zero			*
28	new line compensation space plus is equal to ten curly braces	*		
29	variable test equals ten	*		
29	compensation plus equals ten	*		
30	enter var var code var test is equal to ten	*		
30	competition plus equals to ten	*		
31	variable test equal zero		*	
31	compensation plus equals ten	*		
32	var test equals ten	*		
32	compensation rare is equal to ten	*		
33	var test is equal to ten	*		
33	compensation plus equal ten	*		
34	var test equal to ten	*		
34	compensation plus uh plus or equal to ten	*		
35	assign ten to test variable	*		
35	add oh compensation variable to ten	*		
36	variable test equals ten	*		



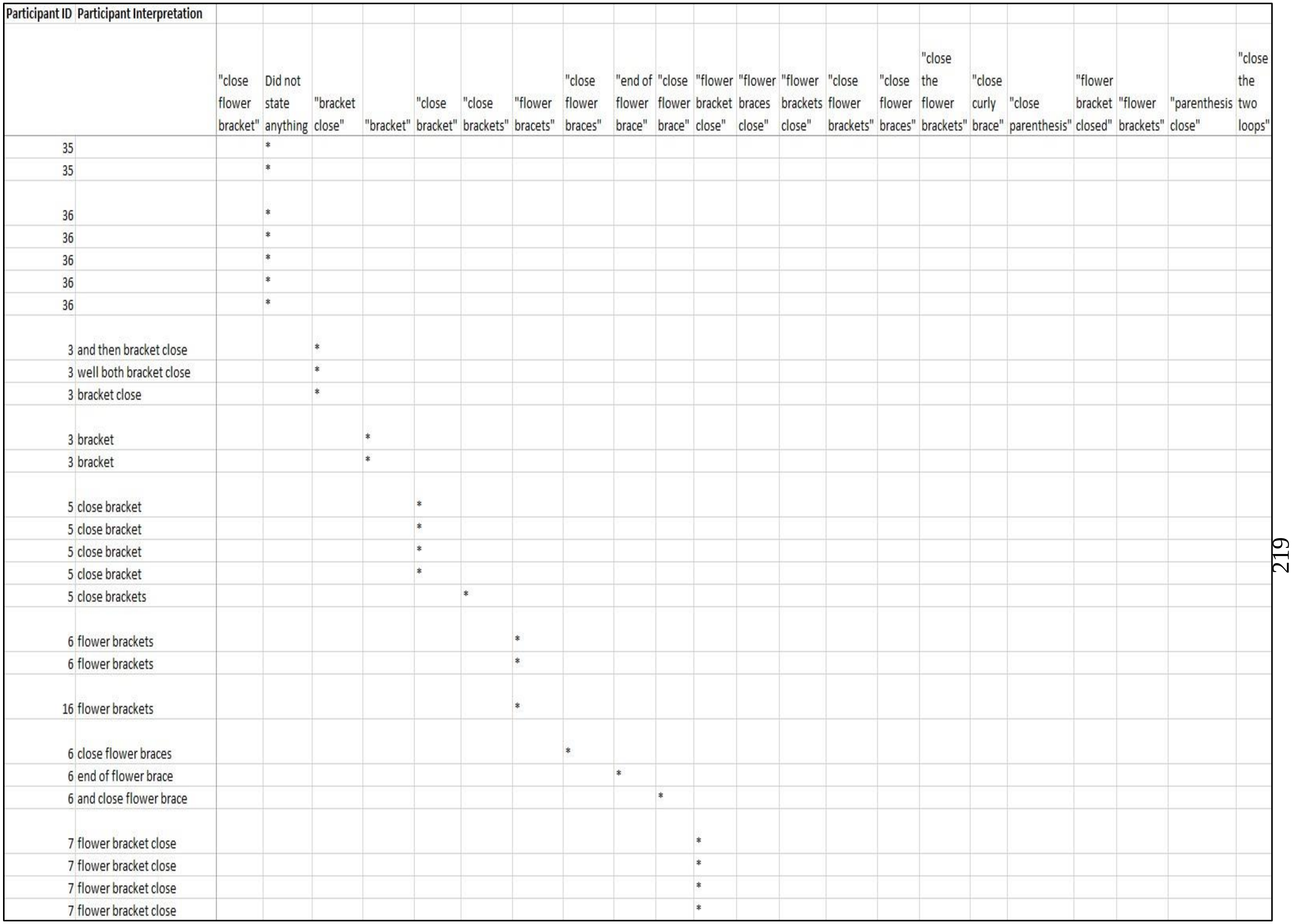
Participant ID	Participant Interpretation			
		"ten"	"zero"	"one zero"
36	and compensation equals compensation plus ten	*		
37	variable test equal to ten	*		
37	compensation equal to increment of compensation by ten	*		
38	var test equal to ten	*		
38	compensation equal to compensation plus ten	*		
	% of Occurences This Was Stated	**94.44444**	**4.166667**	**1.389**
	# of Occurences This Was Stated	**68**	**3**	**1**

E.15) }

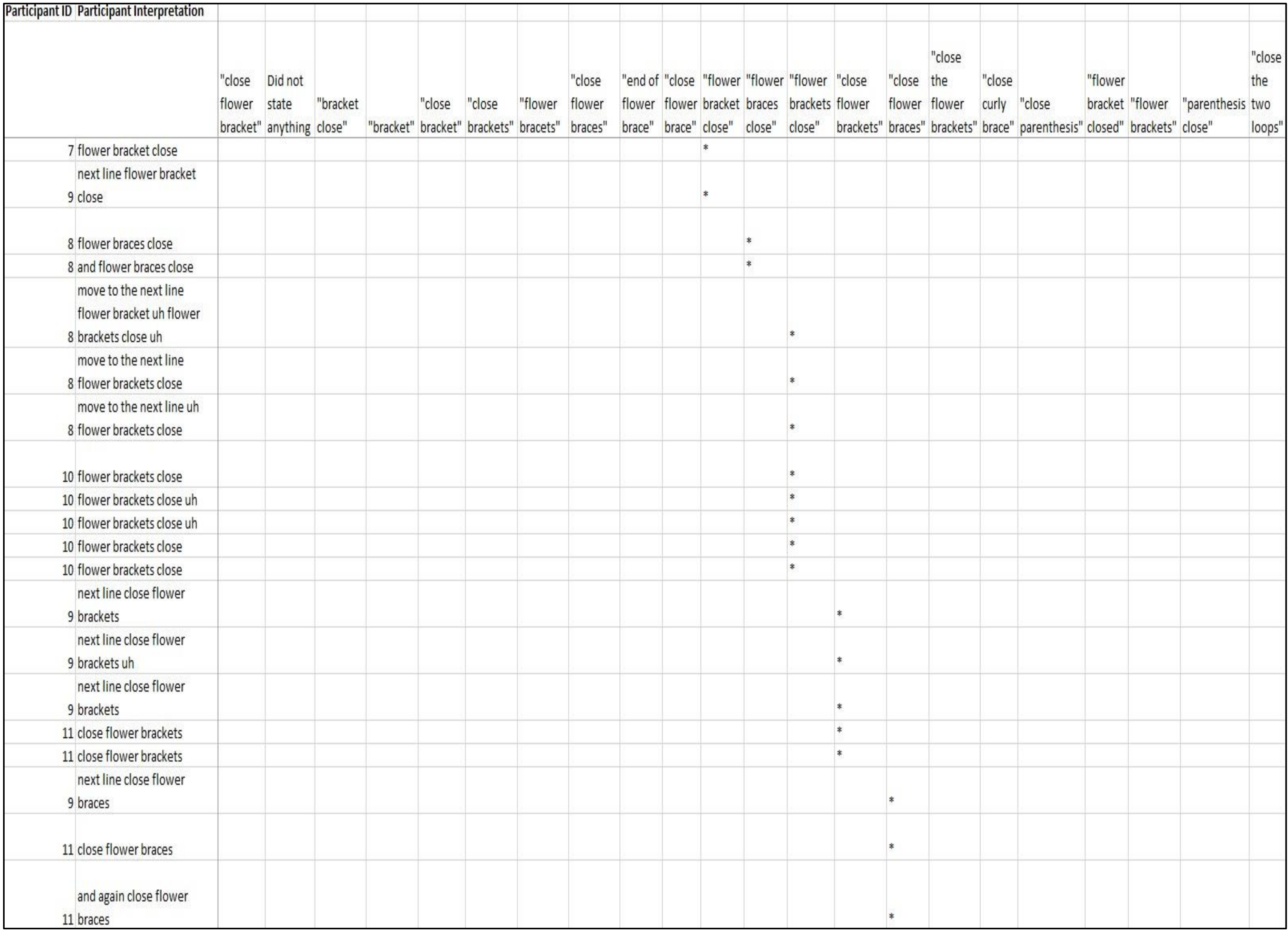


Participant ID	Participant Interpretation																						
		"close flower bracket"	Did not state anything	"bracket close"	"bracket"	"close bracket"	"close brackets"	"flower bracets"	"close flower braces"	"end of flower brace"	"close flower brace"	"flower bracket close"	"flower braces close"	"flower brackets close"	"close flower brackets"	"close flower braces"	"close the flower brackets"	"close curly brace"	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"
1	close flower bracket	*																					
1	close flower bracket	*																					
1	close flower bracket	*																					
1	close flower bracket	*																					
1	close flower bracket	*																					
2			*																				
2			*																				
2			*																				
2			*																				
2			*																				
16			*																				
16			*																				
19			*																				
19			*																				
19			*																				
19			*																				
19			*																				
20			*																				
20			*																				
20			*																				
20			*																				
20			*																				
21			*																				
21			*																				
21			*																				
21			*																				
21			*																				
23			*																				
23			*																				
23			*																				
24			*																				
24			*																				

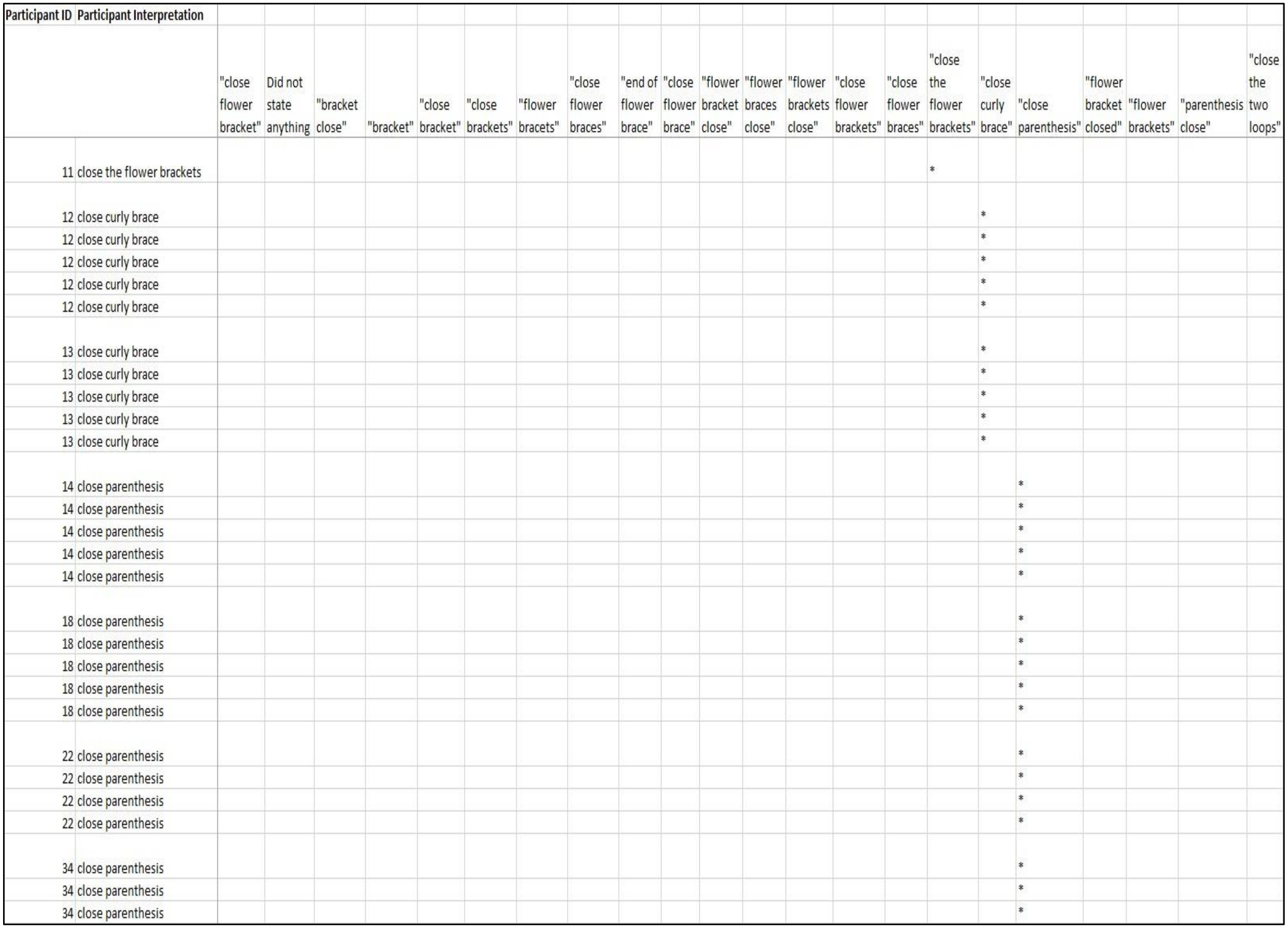
Participant ID	Participant Interpretation																						
		"close flower bracket"	Did not state anything	"bracket close"	"bracket"	"close bracket"	"close brackets"	"flower bracets"	"close flower braces"	"end of flower brace"	"close flower brace"	"flower bracket close"	"flower braces close"	"flower brackets close"	"close flower brackets"	"close flower braces"	"close the flower brackets"	"close curly brace"	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"
24			*																				
24			*																				
26			*																				
26			*																				
26			*																				
26			*																				
28			*																				
29			*																				
29			*																				
29			*																				
29			*																				
29			*																				
31			*																				
31			*																				
31			*																				
31			*																				
31			*																				
32			*																				
32			*																				
32			*																				
32			*																				
32			*																				
33			*																				
33			*																				
33			*																				
33			*																				
33			*																				
35			*																				
35			*																				



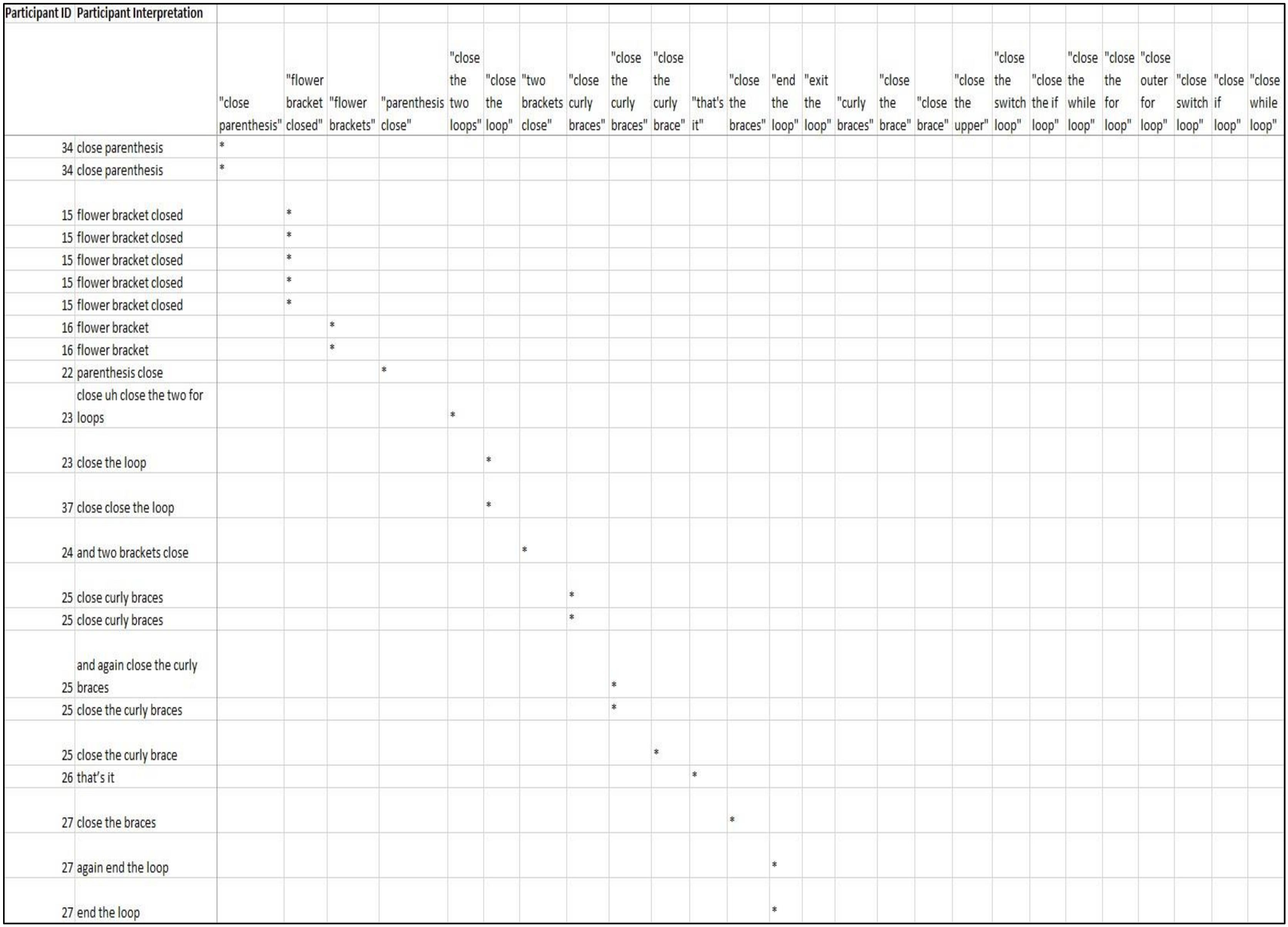
Participant ID	Participant Interpretation	"close flower bracket"	Did not state anything	"bracket close"	"bracket"	"close bracket"	"close brackets"	"flower bracets"	"close flower braces"	"end of flower brace"	"close flower brace"	"flower bracket close"	"flower braces close"	"flower brackets close"	"close flower brackets"	"close flower braces"	"close the flower brackets"	"close curly brace"	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"
35			*																				
35			*																				
36			*																				
36			*																				
36			*																				
36			*																				
36			*																				
3	and then bracket close			*																			
3	well both bracket close			*																			
3	bracket close			*																			
3	bracket				*																		
3	bracket				*																		
5	close bracket					*																	
5	close bracket					*																	
5	close bracket					*																	
5	close bracket					*																	
5	close brackets						*																
6	flower brackets							*															
6	flower brackets							*															
16	flower brackets							*															
6	close flower braces								*														
6	end of flower brace									*													
6	and close flower brace										*												
7	flower bracket close											*											
7	flower bracket close											*											
7	flower bracket close											*											
7	flower bracket close											*											



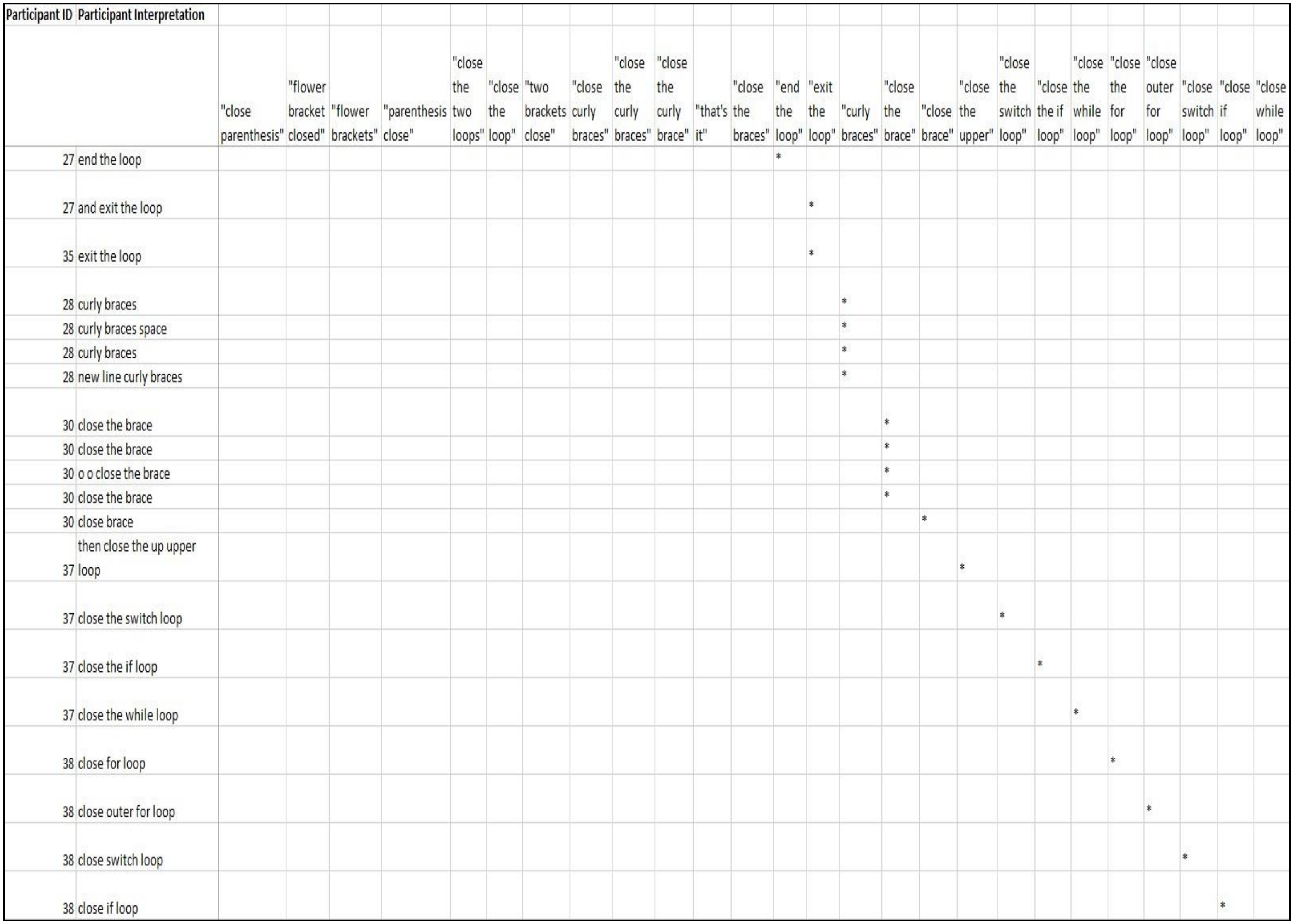
Participant ID	Participant Interpretation	"close flower bracket"	Did not state anything	"bracket close"	"bracket"	"close bracket"	"close brackets"	"flower bracets"	"close flower braces"	"end of flower brace"	"close flower brace"	"flower bracket close"	"flower braces close"	"flower brackets close"	"close flower brackets"	"close flower braces"	"close the flower brackets"	"close curly brace"	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"
7	flower bracket close											*											
9	next line flower bracket close											*											
8	flower braces close												*										
8	and flower braces close												*										
8	move to the next line flower bracket uh flower brackets close uh													*									
8	move to the next line flower brackets close													*									
8	move to the next line uh flower brackets close													*									
10	flower brackets close													*									
10	flower brackets close uh													*									
10	flower brackets close uh													*									
10	flower brackets close													*									
10	flower brackets close													*									
9	next line close flower brackets														*								
9	next line close flower brackets uh														*								
9	next line close flower brackets														*								
11	close flower brackets														*								
11	close flower brackets														*								
9	next line close flower braces															*							
11	close flower braces															*							
11	and again close flower braces															*							



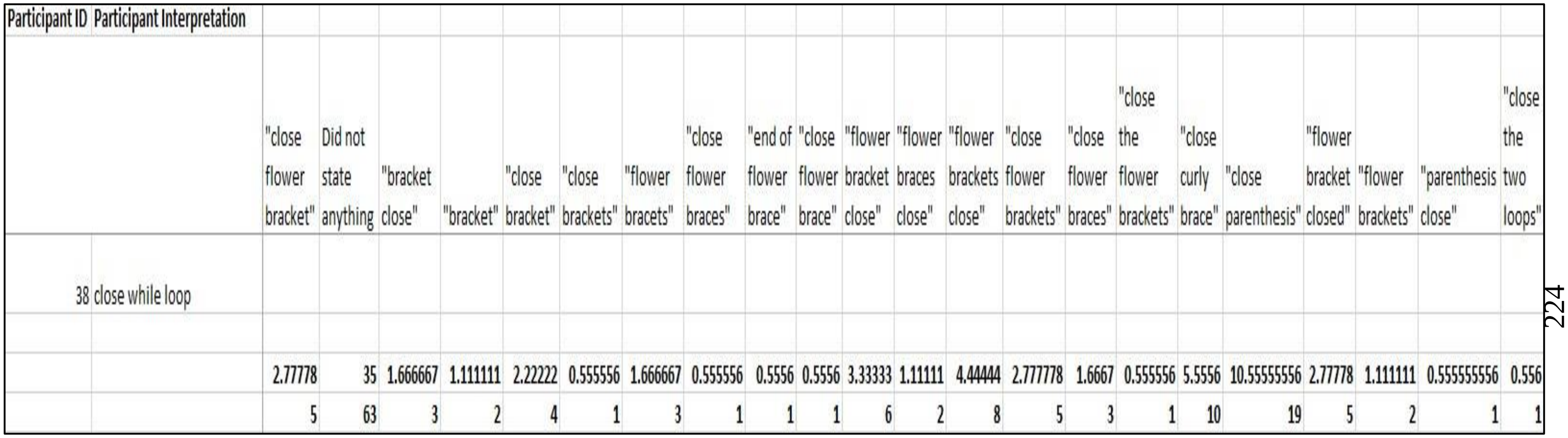
Participant ID	Participant Interpretation	"close flower bracket"	Did not state anything	"bracket close"	"bracket"	"close bracket"	"close brackets"	"flower bracets"	"close flower braces"	"end of flower brace"	"close flower brace"	"flower bracket close"	"flower braces close"	"flower brackets close"	"close flower brackets"	"close flower braces"	"close the flower brackets"	"close curly brace"	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"
11	close the flower brackets																*						
12	close curly brace																	*					
12	close curly brace																	*					
12	close curly brace																	*					
12	close curly brace																	*					
12	close curly brace																	*					
13	close curly brace																	*					
13	close curly brace																	*					
13	close curly brace																	*					
13	close curly brace																	*					
13	close curly brace																	*					
14	close parenthesis																		*				
14	close parenthesis																		*				
14	close parenthesis																		*				
14	close parenthesis																		*				
14	close parenthesis																		*				
18	close parenthesis																		*				
18	close parenthesis																		*				
18	close parenthesis																		*				
18	close parenthesis																		*				
18	close parenthesis																		*				
22	close parenthesis																		*				
22	close parenthesis																		*				
22	close parenthesis																		*				
22	close parenthesis																		*				
34	close parenthesis																		*				
34	close parenthesis																		*				
34	close parenthesis																		*				



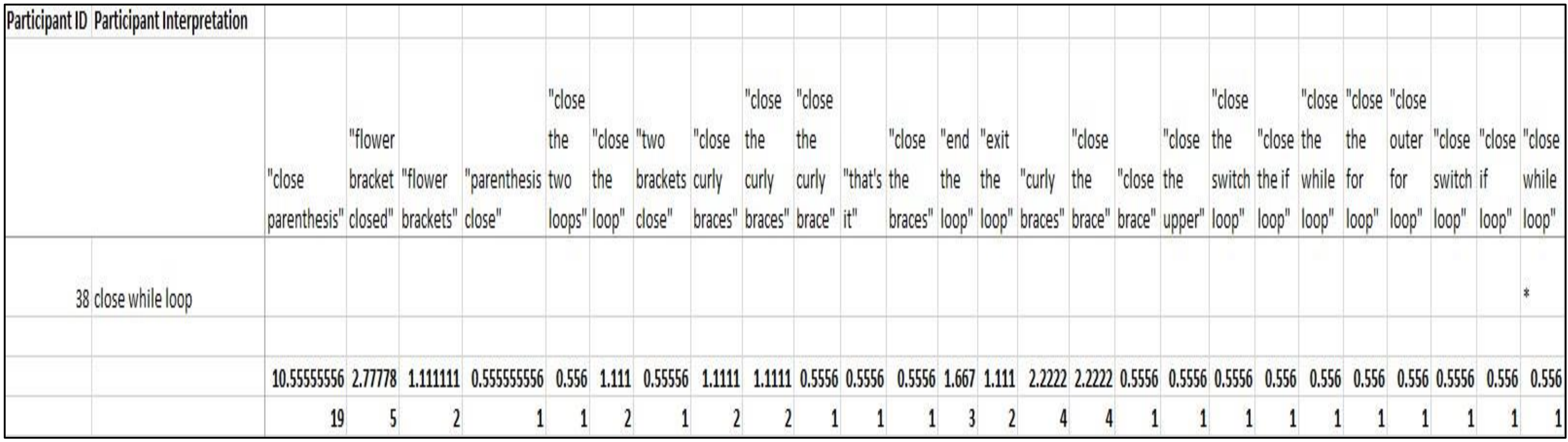
Participant ID	Participant Interpretation	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"	"close the loop"	"two brackets close"	"close curly braces"	"close the curly braces"	"close the curly brace"	"that's it"	"close the braces"	"end the loop"	"exit the loop"	"curly braces"	"close the brace"	"close brace"	"close the upper"	"close the switch loop"	"close the if loop"	"close the while loop"	"close the for loop"	"close outer for loop"	"close switch loop"	"close if loop"	"close while loop"
34	close parenthesis	*																									
34	close parenthesis	*																									
15	flower bracket closed		*																								
15	flower bracket closed		*																								
15	flower bracket closed		*																								
15	flower bracket closed		*																								
15	flower bracket closed		*																								
16	flower bracket			*																							
16	flower bracket			*																							
22	parenthesis close				*																						
23	close uh close the two for loops					*																					
23	close the loop						*																				
37	close close the loop						*																				
24	and two brackets close							*																			
25	close curly braces								*																		
25	close curly braces								*																		
25	and again close the curly braces									*																	
25	close the curly braces									*																	
25	close the curly brace										*																
26	that's it											*															
27	close the braces												*														
27	again end the loop													*													
27	end the loop													*													



Participant ID	Participant Interpretation	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"	"close the loop"	"two brackets close"	"close curly braces"	"close the curly braces"	"close the curly brace"	"that's it"	"close the braces"	"end the loop"	"exit the loop"	"curly braces"	"close the brace"	"close brace"	"close the upper"	"close the switch loop"	"close the if loop"	"close the while loop"	"close the for loop"	"close outer for loop"	"close switch loop"	"close if loop"	"close while loop"
27	end the loop													*													
27	and exit the loop														*												
35	exit the loop														*												
28	curly braces															*											
28	curly braces space															*											
28	curly braces															*											
28	new line curly braces															*											
30	close the brace																*										
30	close the brace																*										
30	o o close the brace																*										
30	close the brace																*										
30	close brace																	*									
37	then close the up upper loop																		*								
37	close the switch loop																			*							
37	close the if loop																				*						
37	close the while loop																					*					
38	close for loop																						*				
38	close outer for loop																							*			
38	close switch loop																								*		
38	close if loop																									*	

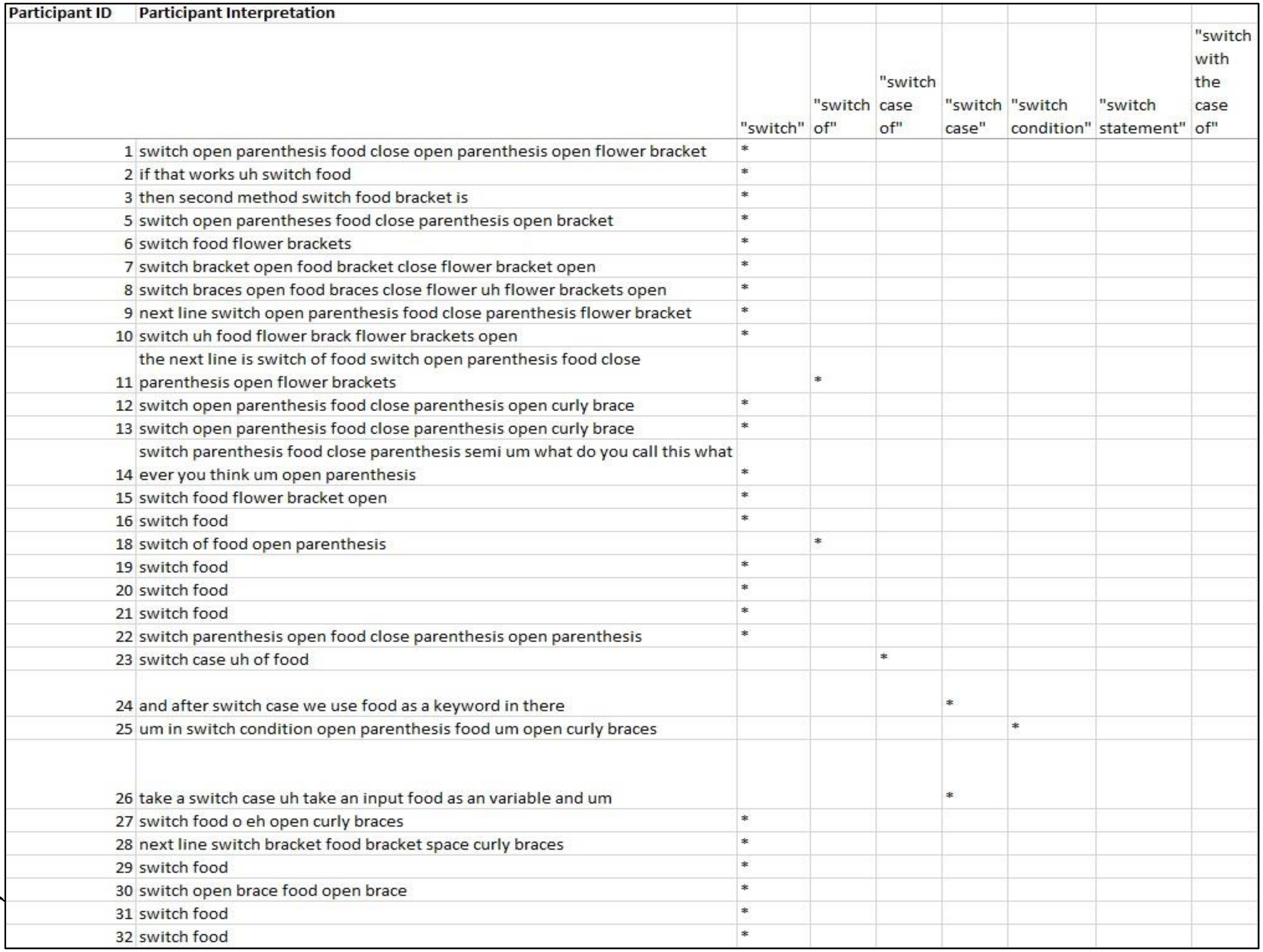


Participant ID	Participant Interpretation																						
		"close flower bracket"	Did not state anything	"bracket close"	"bracket"	"close bracket"	"close brackets"	"flower bracets"	"close flower braces"	"end of flower brace"	"close flower brace"	"flower bracket close"	"flower braces close"	"flower brackets close"	"close flower brackets"	"close flower braces"	"close the flower brackets"	"close curly brace"	"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"
38	close while loop																						
		2.77778	35	1.666667	1.111111	2.22222	0.555556	1.666667	0.555556	0.5556	0.5556	3.33333	1.11111	4.44444	2.777778	1.6667	0.555556	5.5556	10.55555556	2.77778	1.111111	0.555555556	0.556
		5	63	3	2	4	1	3	1	1	1	6	2	8	5	3	1	10	19	5	2	1	1

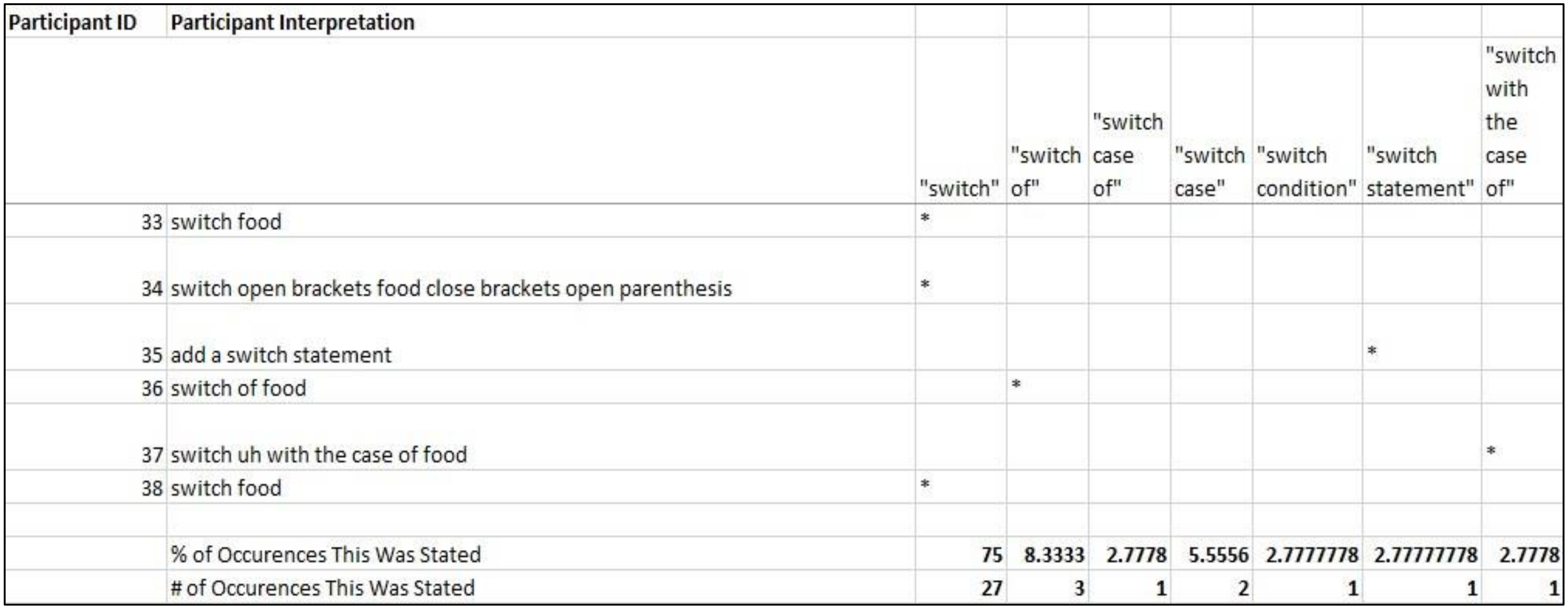


Participant ID	Participant Interpretation																										
		"close parenthesis"	"flower bracket closed"	"flower brackets"	"parenthesis close"	"close the two loops"	"close the loop"	"two brackets close"	"close curly braces"	"close the curly braces"	"close the curly brace"	"that's it"	"close the braces"	"end the loop"	"exit the loop"	"curly braces"	"close the brace"	"close brace"	"close the upper"	"close the switch loop"	"close the if loop"	"close the while loop"	"close the for loop"	"close outer for loop"	"close switch loop"	"close if loop"	"close while loop"
38	close while loop																										*
		10.55555556	2.77778	1.111111	0.555555556	0.556	1.111	0.55556	1.1111	1.1111	0.5556	0.5556	0.5556	1.667	1.111	2.2222	2.2222	0.5556	0.5556	0.5556	0.556	0.556	0.556	0.556	0.5556	0.556	0.556
		19	5	2	1	1	2	1	2	2	1	1	1	3	2	4	4	1	1	1	1	1	1	1	1	1	1

E.16) switch

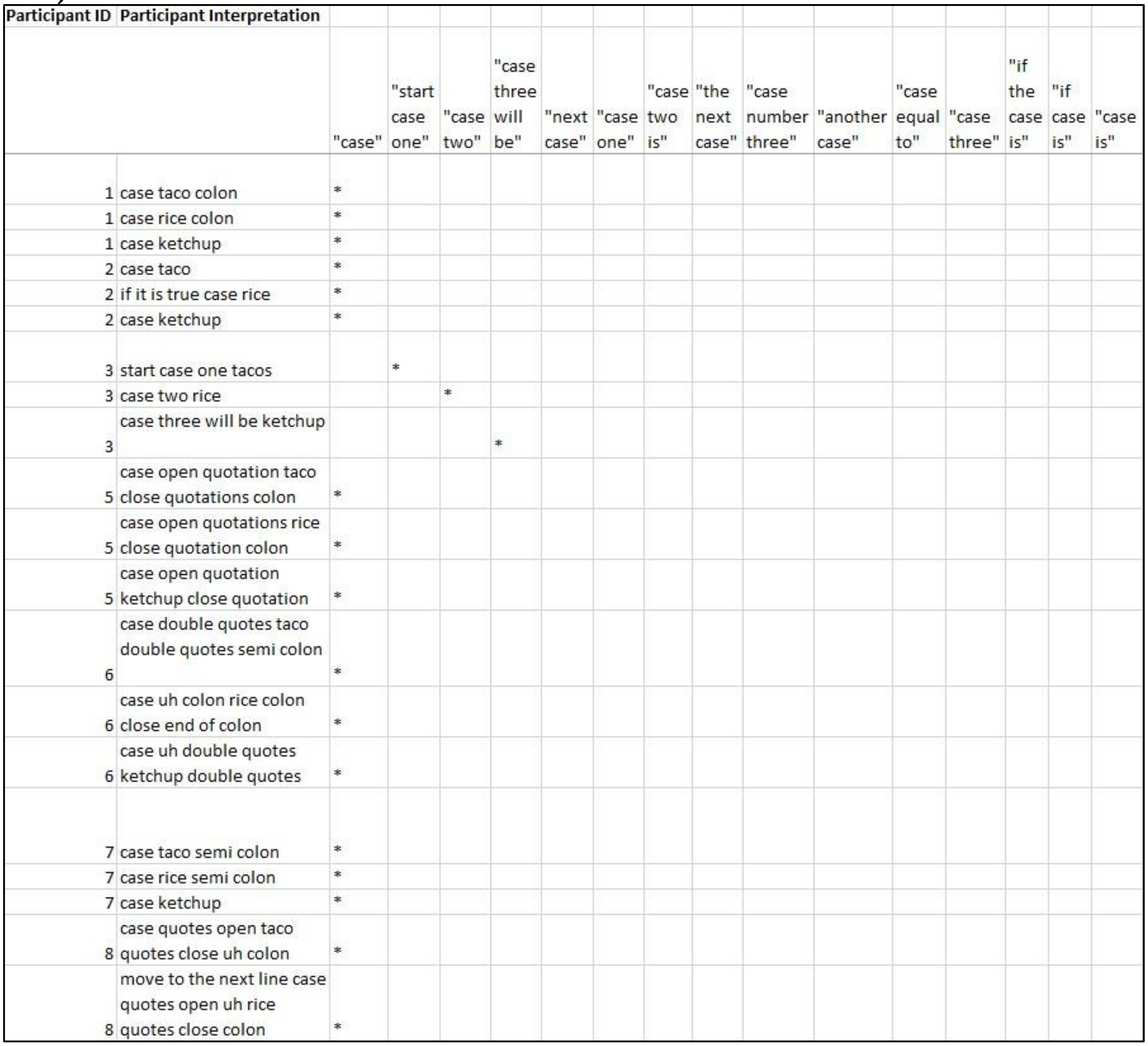


Participant ID	Participant Interpretation							
		"switch"	"switch of"	"switch case of"	"switch case"	"switch condition"	"switch statement"	"switch with the case of"
1	switch open parenthesis food close open parenthesis open flower bracket	*						
2	if that works uh switch food	*						
3	then second method switch food bracket is	*						
5	switch open parentheses food close parenthesis open bracket	*						
6	switch food flower brackets	*						
7	switch bracket open food bracket close flower bracket open	*						
8	switch braces open food braces close flower uh flower brackets open	*						
9	next line switch open parenthesis food close parenthesis flower bracket	*						
10	switch uh food flower brack flower brackets open	*						
11	the next line is switch of food switch open parenthesis food close parenthesis open flower brackets		*					
12	switch open parenthesis food close parenthesis open curly brace	*						
13	switch open parenthesis food close parenthesis open curly brace	*						
14	switch parenthesis food close parenthesis semi um what do you call this what ever you think um open parenthesis	*						
15	switch food flower bracket open	*						
16	switch food	*						
18	switch of food open parenthesis		*					
19	switch food	*						
20	switch food	*						
21	switch food	*						
22	switch parenthesis open food close parenthesis open parenthesis	*						
23	switch case uh of food			*				
24	and after switch case we use food as a keyword in there				*			
25	um in switch condition open parenthesis food um open curly braces					*		
26	take a switch case uh take an input food as an variable and um				*			
27	switch food o eh open curly braces	*						
28	next line switch bracket food bracket space curly braces	*						
29	switch food	*						
30	switch open brace food open brace	*						
31	switch food	*						
32	switch food	*						

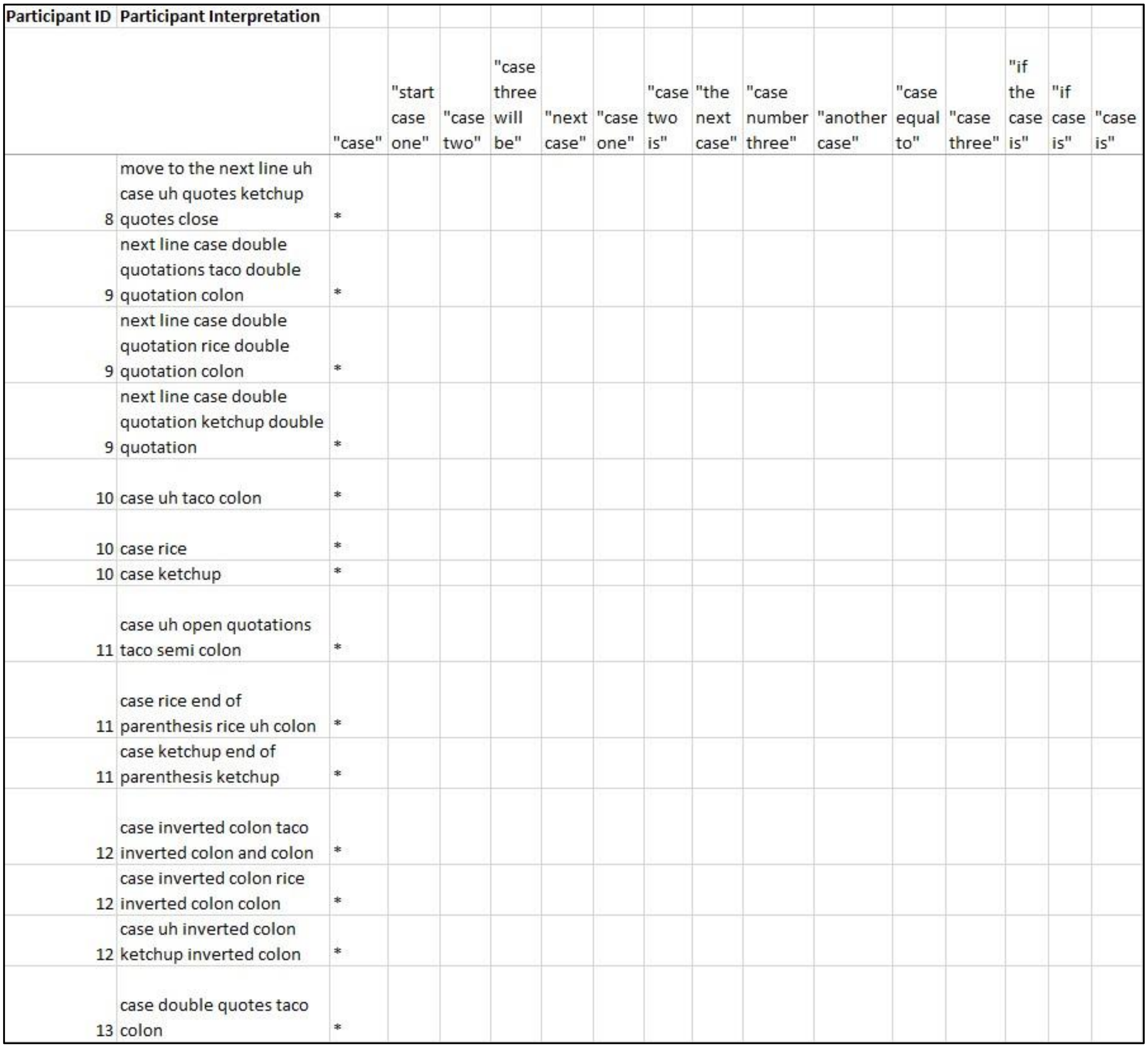


Participant ID	Participant Interpretation							
		"switch"	"switch of"	"switch case of"	"switch case"	"switch condition"	"switch statement"	"switch with the case of"
33	switch food	*						
34	switch open brackets food close brackets open parenthesis	*						
35	add a switch statement						*	
36	switch of food		*					
37	switch uh with the case of food							*
38	switch food	*						
	% of Occurences This Was Stated	**75**	**8.3333**	**2.7778**	**5.5556**	**2.7777778**	**2.77777778**	**2.7778**
	# of Occurences This Was Stated	**27**	**3**	**1**	**2**	**1**	**1**	**1**

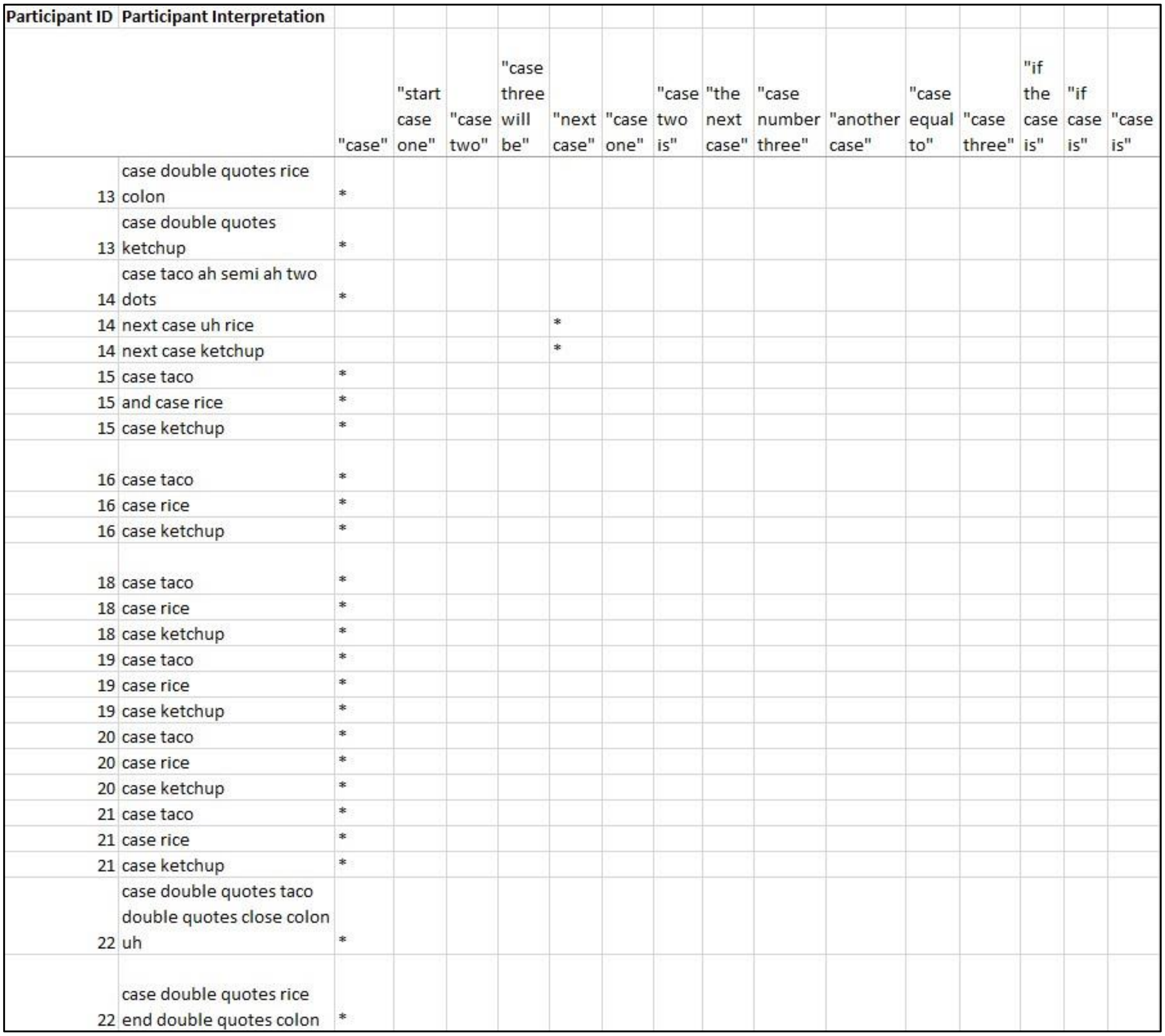
E.17) case



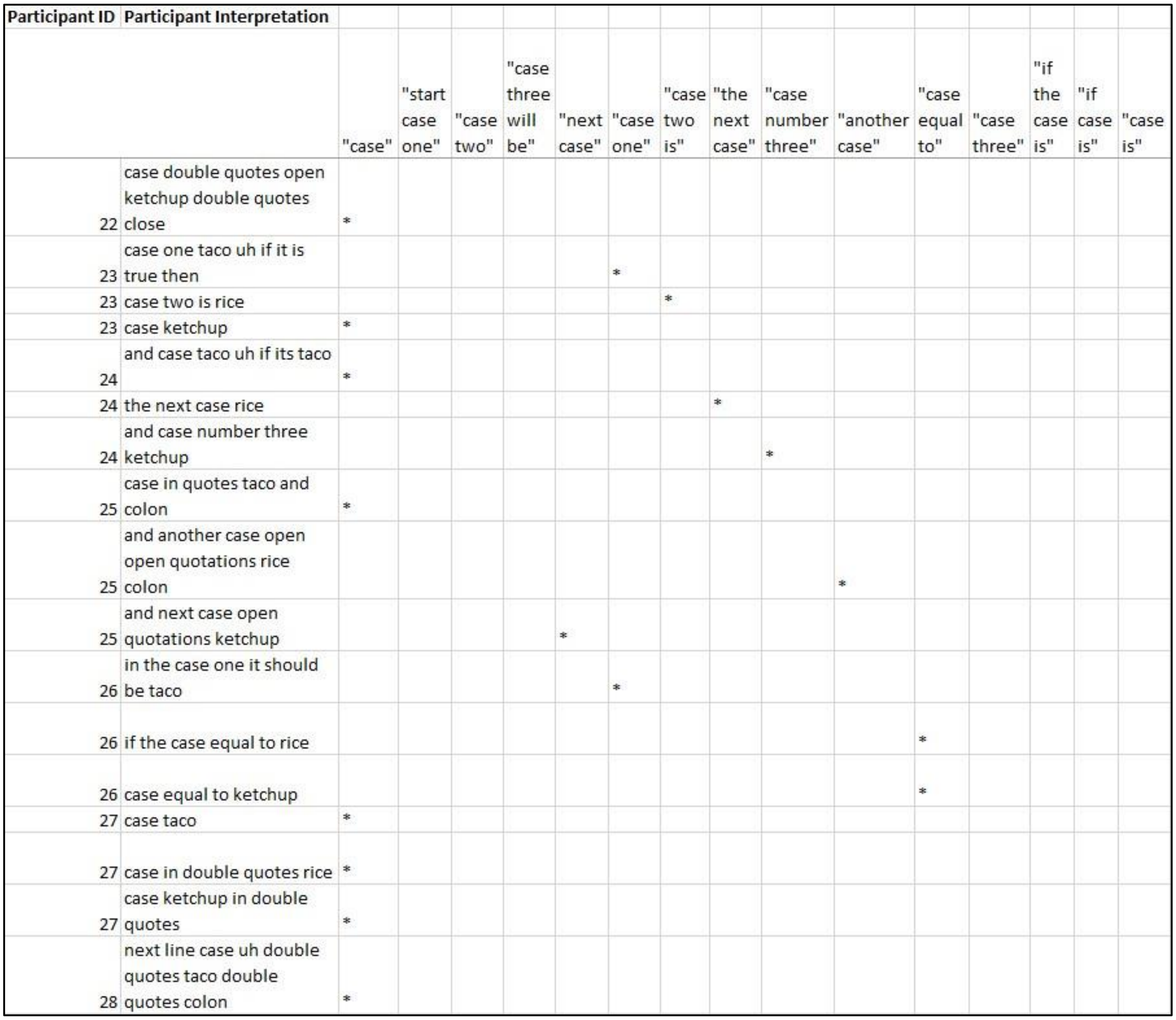
Participant ID	Participant Interpretation															
		"case"	"start case one"	"case two"	"case three will be"	"next case"	"case one"	"case two is"	"the next case"	"case number three"	"another case"	"case equal to"	"case three"	"if the case is"	"if case is"	"case is"
1	case taco colon	*														
1	case rice colon	*														
1	case ketchup	*														
2	case taco	*														
2	if it is true case rice	*														
2	case ketchup	*														
3	start case one tacos		*													
3	case two rice			*												
3	case three will be ketchup				*											
5	case open quotation taco close quotations colon	*														
5	case open quotations rice close quotation colon	*														
5	case open quotation ketchup close quotation	*														
6	case double quotes taco double quotes semi colon	*														
6	case uh colon rice colon close end of colon	*														
6	case uh double quotes ketchup double quotes	*														
7	case taco semi colon	*														
7	case rice semi colon	*														
7	case ketchup	*														
8	case quotes open taco quotes close uh colon	*														
8	move to the next line case quotes open uh rice quotes close colon	*														



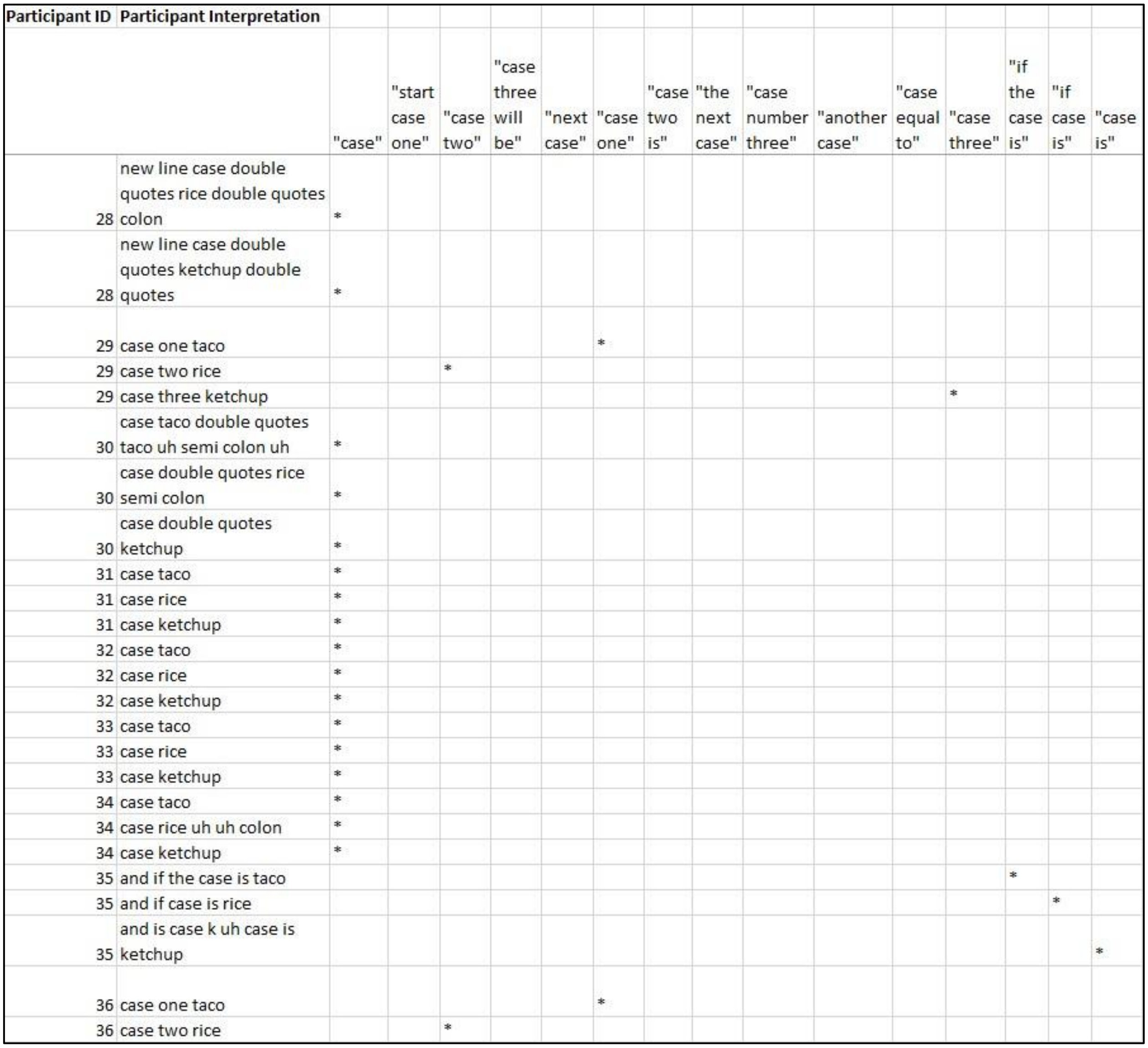
Participant ID	Participant Interpretation															
		"case"	"start case one"	"case two"	"case three will be"	"next case"	"case one"	"case two is"	"the next case"	"case number three"	"another case"	"case equal to"	"case three"	"if the case is"	"if case is"	"case is"
8	move to the next line uh case uh quotes ketchup quotes close	*														
9	next line case double quotations taco double quotation colon	*														
9	next line case double quotation rice double quotation colon	*														
9	next line case double quotation ketchup double quotation	*														
10	case uh taco colon	*														
10	case rice	*														
10	case ketchup	*														
11	case uh open quotations taco semi colon	*														
11	case rice end of parenthesis rice uh colon	*														
11	case ketchup end of parenthesis ketchup	*														
12	case inverted colon taco inverted colon and colon	*														
12	case inverted colon rice inverted colon colon	*														
12	case uh inverted colon ketchup inverted colon	*														
13	case double quotes taco colon	*														



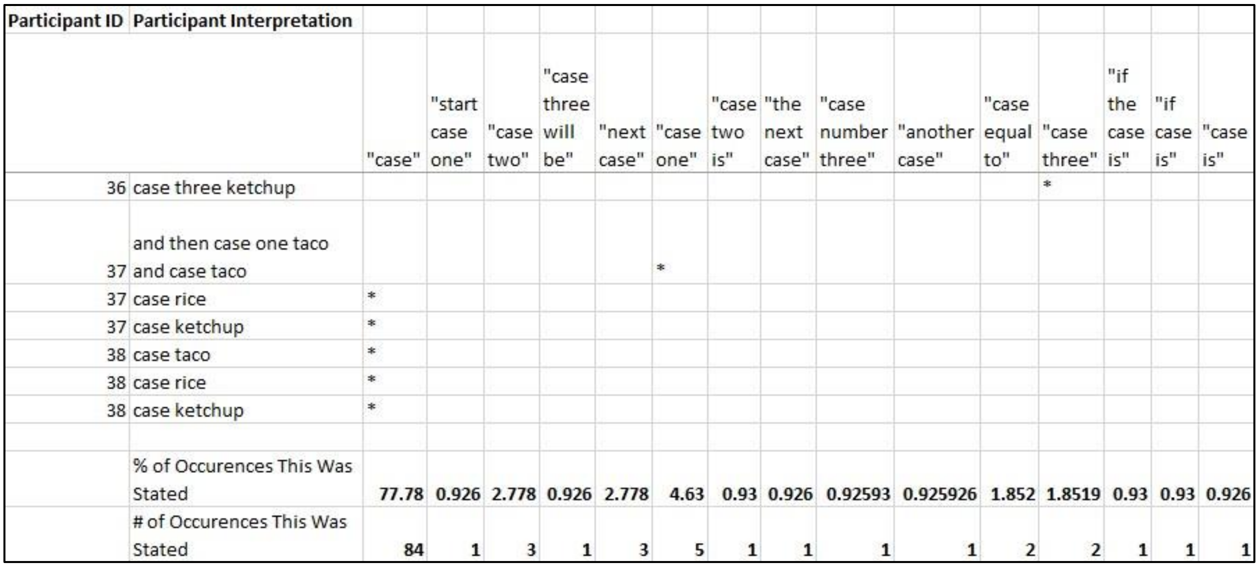
Participant ID	Participant Interpretation															
		"case"	"start case one"	"case two"	"case three will be"	"next case"	"case one"	"case two is"	"the next case"	"case number three"	"another case"	"case equal to"	"case three"	"if the case is"	"if case is"	"case is"
13	case double quotes rice colon	*														
13	case double quotes ketchup	*														
14	case taco ah semi ah two dots	*														
14	next case uh rice					*										
14	next case ketchup					*										
15	case taco	*														
15	and case rice	*														
15	case ketchup	*														
16	case taco	*														
16	case rice	*														
16	case ketchup	*														
18	case taco	*														
18	case rice	*														
18	case ketchup	*														
19	case taco	*														
19	case rice	*														
19	case ketchup	*														
20	case taco	*														
20	case rice	*														
20	case ketchup	*														
21	case taco	*														
21	case rice	*														
21	case ketchup	*														
22	case double quotes taco double quotes close colon uh	*														
22	case double quotes rice end double quotes colon	*														



Participant ID	Participant Interpretation	"case"	"start case one"	"case two"	"case three will be"	"next case"	"case one"	"case two is"	"the next case"	"case number three"	"another case"	"case equal to"	"case three"	"if the case is"	"if case is"	"case is"
22	case double quotes open ketchup double quotes close	*														
23	case one taco uh if it is true then						*									
23	case two is rice							*								
23	case ketchup	*														
24	and case taco uh if its taco	*														
24	the next case rice								*							
24	and case number three ketchup									*						
25	case in quotes taco and colon	*														
25	and another case open open quotations rice colon										*					
25	and next case open quotations ketchup					*										
26	in the case one it should be taco						*									
26	if the case equal to rice											*				
26	case equal to ketchup											*				
27	case taco	*														
27	case in double quotes rice	*														
27	case ketchup in double quotes	*														
28	next line case uh double quotes taco double quotes colon	*														

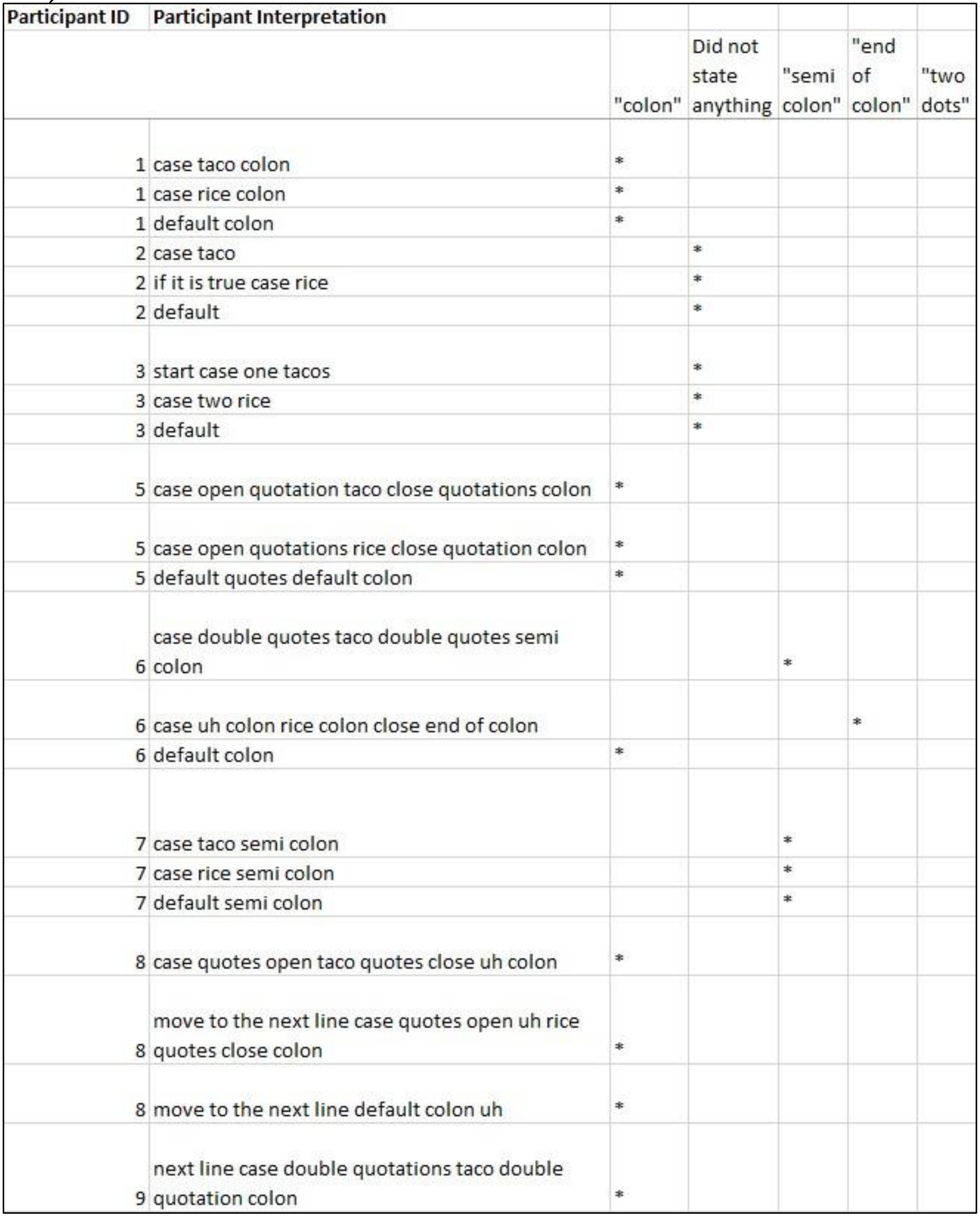


Participant ID	Participant Interpretation															
		"case"	"start case one"	"case two"	"case three will be"	"next case"	"case one"	"case two is"	"the next case"	"case number three"	"another case"	"case equal to"	"case three"	"if the case is"	"if case is"	"case is"
28	new line case double quotes rice double quotes colon	*														
28	new line case double quotes ketchup double quotes	*														
29	case one taco						*									
29	case two rice			*												
29	case three ketchup												*			
30	case taco double quotes taco uh semi colon uh	*														
30	case double quotes rice semi colon	*														
30	case double quotes ketchup	*														
31	case taco	*														
31	case rice	*														
31	case ketchup	*														
32	case taco	*														
32	case rice	*														
32	case ketchup	*														
33	case taco	*														
33	case rice	*														
33	case ketchup	*														
34	case taco	*														
34	case rice uh uh colon	*														
34	case ketchup	*														
35	and if the case is taco													*		
35	and if case is rice														*	
35	and is case k uh case is ketchup															*
36	case one taco						*									
36	case two rice			*												

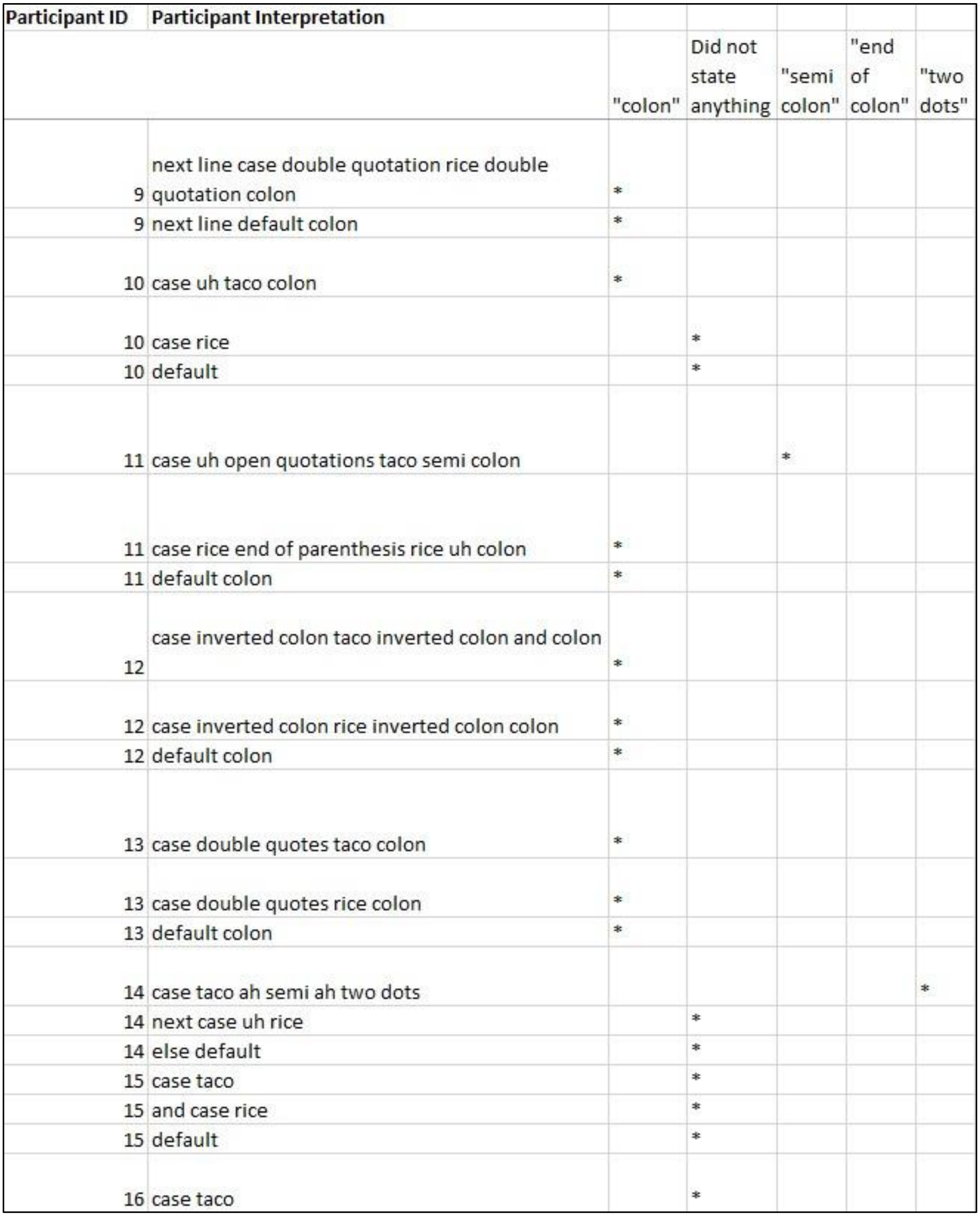


Participant ID	Participant Interpretation															
		"case"	"start case one"	"case two"	"case three will be"	"next case"	"case one"	"case two is"	"the next case"	"case number three"	"another case"	"case equal to"	"case three"	"if the case is"	"if case is"	"case is"
36	case three ketchup												*			
37	and then case one taco and case taco						*									
37	case rice	*														
37	case ketchup	*														
38	case taco	*														
38	case rice	*														
38	case ketchup	*														
	% of Occurences This Was Stated	**77.78**	**0.926**	**2.778**	**0.926**	**2.778**	**4.63**	**0.93**	**0.926**	**0.92593**	**0.925926**	**1.852**	**1.8519**	**0.93**	**0.93**	**0.926**
	# of Occurences This Was Stated	**84**	**1**	**3**	**1**	**3**	**5**	**1**	**1**	**1**	**1**	**2**	**2**	**1**	**1**	**1**

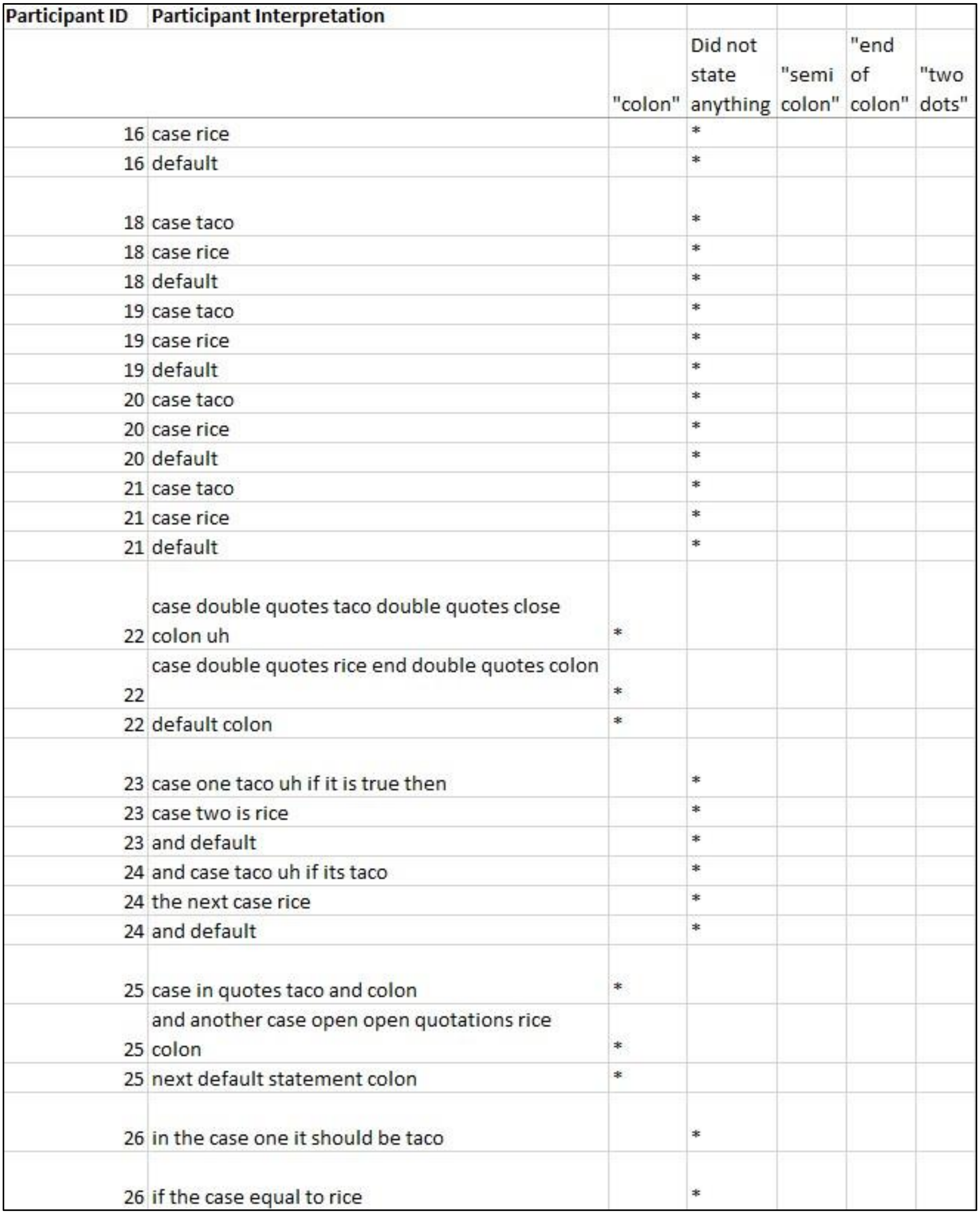
E.18) colon



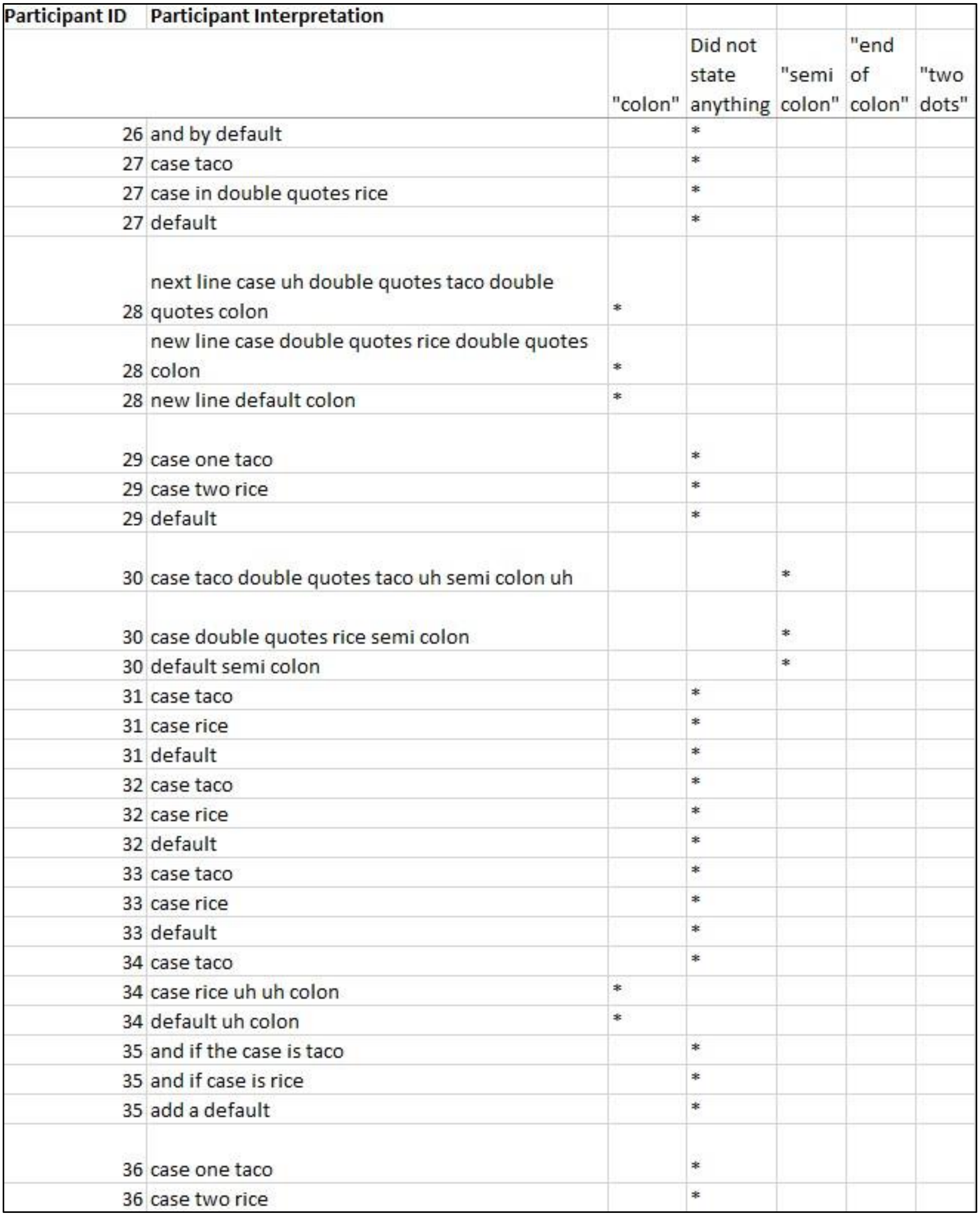
Participant ID	Participant Interpretation					
		"colon"	Did not state anything	"semi colon"	"end of colon"	"two dots"
1	case taco colon	*				
1	case rice colon	*				
1	default colon	*				
2	case taco		*			
2	if it is true case rice		*			
2	default		*			
3	start case one tacos		*			
3	case two rice		*			
3	default		*			
5	case open quotation taco close quotations colon	*				
5	case open quotations rice close quotation colon	*				
5	default quotes default colon	*				
6	case double quotes taco double quotes semi colon			*		
6	case uh colon rice colon close end of colon				*	
6	default colon	*				
7	case taco semi colon			*		
7	case rice semi colon			*		
7	default semi colon			*		
8	case quotes open taco quotes close uh colon	*				
8	move to the next line case quotes open uh rice quotes close colon	*				
8	move to the next line default colon uh	*				
9	next line case double quotation taco double quotation colon	*				



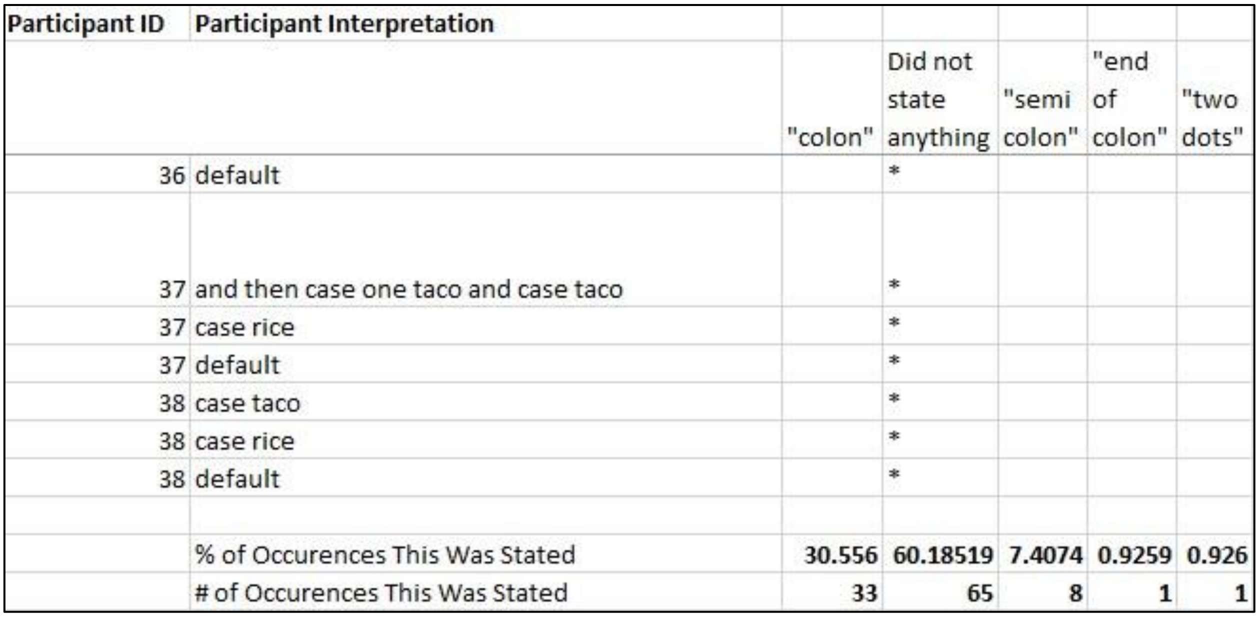
Participant ID	Participant Interpretation					
		"colon"	Did not state anything	"semi colon"	"end of colon"	"two dots"
9	next line case double quotation rice double quotation colon	*				
9	next line default colon	*				
10	case uh taco colon	*				
10	case rice		*			
10	default		*			
11	case uh open quotations taco semi colon			*		
11	case rice end of parenthesis rice uh colon	*				
11	default colon	*				
12	case inverted colon taco inverted colon and colon	*				
12	case inverted colon rice inverted colon colon	*				
12	default colon	*				
13	case double quotes taco colon	*				
13	case double quotes rice colon	*				
13	default colon	*				
14	case taco ah semi ah two dots					*
14	next case uh rice		*			
14	else default		*			
15	case taco		*			
15	and case rice		*			
15	default		*			
16	case taco		*			



Participant ID	Participant Interpretation					
		"colon"	Did not state anything	"semi colon"	"end of colon"	"two dots"
16	case rice		*			
16	default		*			
18	case taco		*			
18	case rice		*			
18	default		*			
19	case taco		*			
19	case rice		*			
19	default		*			
20	case taco		*			
20	case rice		*			
20	default		*			
21	case taco		*			
21	case rice		*			
21	default		*			
22	case double quotes taco double quotes close colon uh	*				
22	case double quotes rice end double quotes colon	*				
22	default colon	*				
23	case one taco uh if it is true then		*			
23	case two is rice		*			
23	and default		*			
24	and case taco uh if its taco		*			
24	the next case rice		*			
24	and default		*			
25	case in quotes taco and colon	*				
25	and another case open open quotations rice colon	*				
25	next default statement colon	*				
26	in the case one it should be taco		*			
26	if the case equal to rice		*			

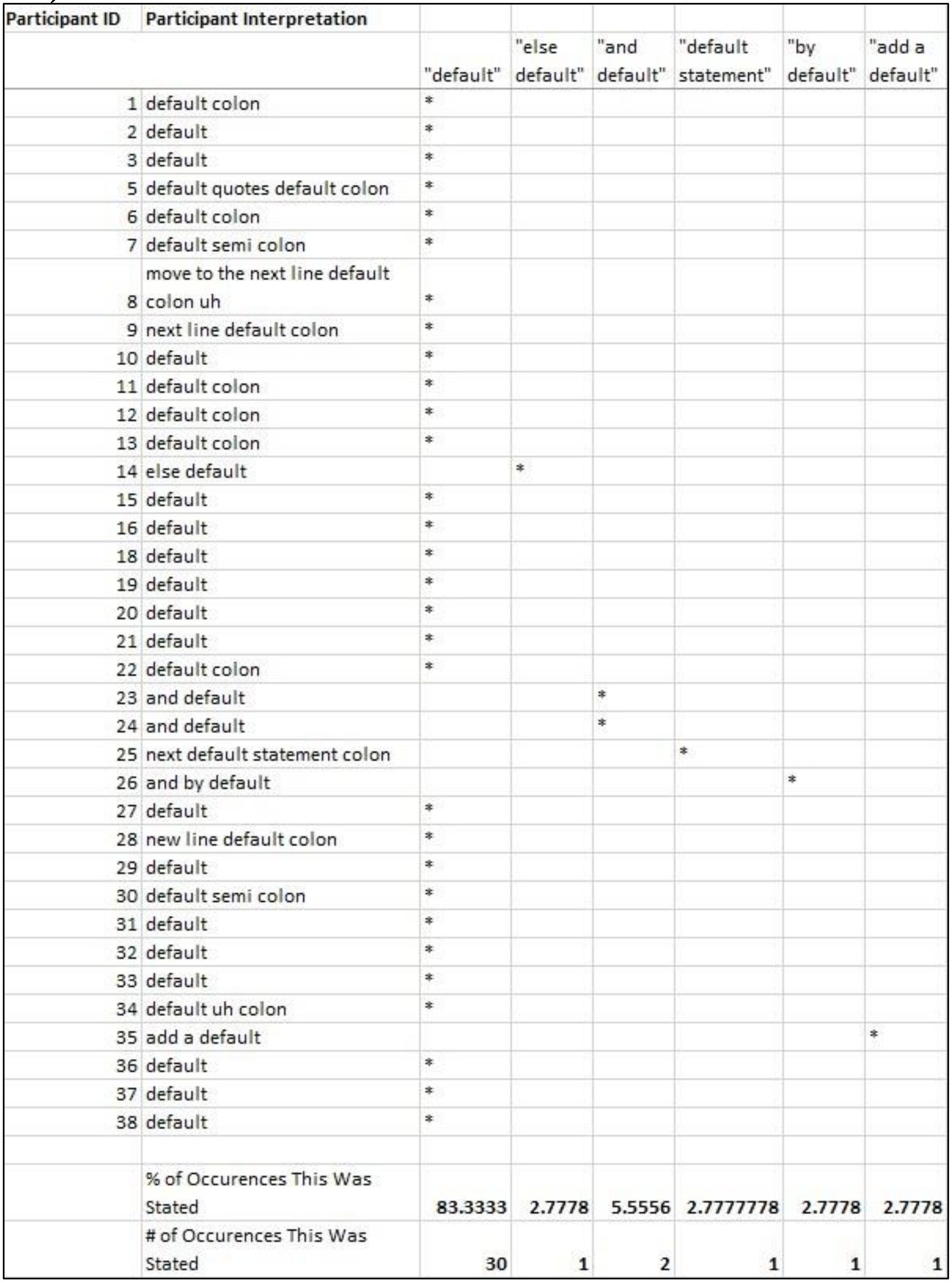


Participant ID	Participant Interpretation					
		"colon"	Did not state anything	"semi colon"	"end of colon"	"two dots"
26	and by default		*			
27	case taco		*			
27	case in double quotes rice		*			
27	default		*			
28	next line case uh double quotes taco double quotes colon	*				
28	new line case double quotes rice double quotes colon	*				
28	new line default colon	*				
29	case one taco		*			
29	case two rice		*			
29	default		*			
30	case taco double quotes taco uh semi colon uh			*		
30	case double quotes rice semi colon			*		
30	default semi colon			*		
31	case taco		*			
31	case rice		*			
31	default		*			
32	case taco		*			
32	case rice		*			
32	default		*			
33	case taco		*			
33	case rice		*			
33	default		*			
34	case taco		*			
34	case rice uh uh colon	*				
34	default uh colon	*				
35	and if the case is taco		*			
35	and if case is rice		*			
35	add a default		*			
36	case one taco		*			
36	case two rice		*			



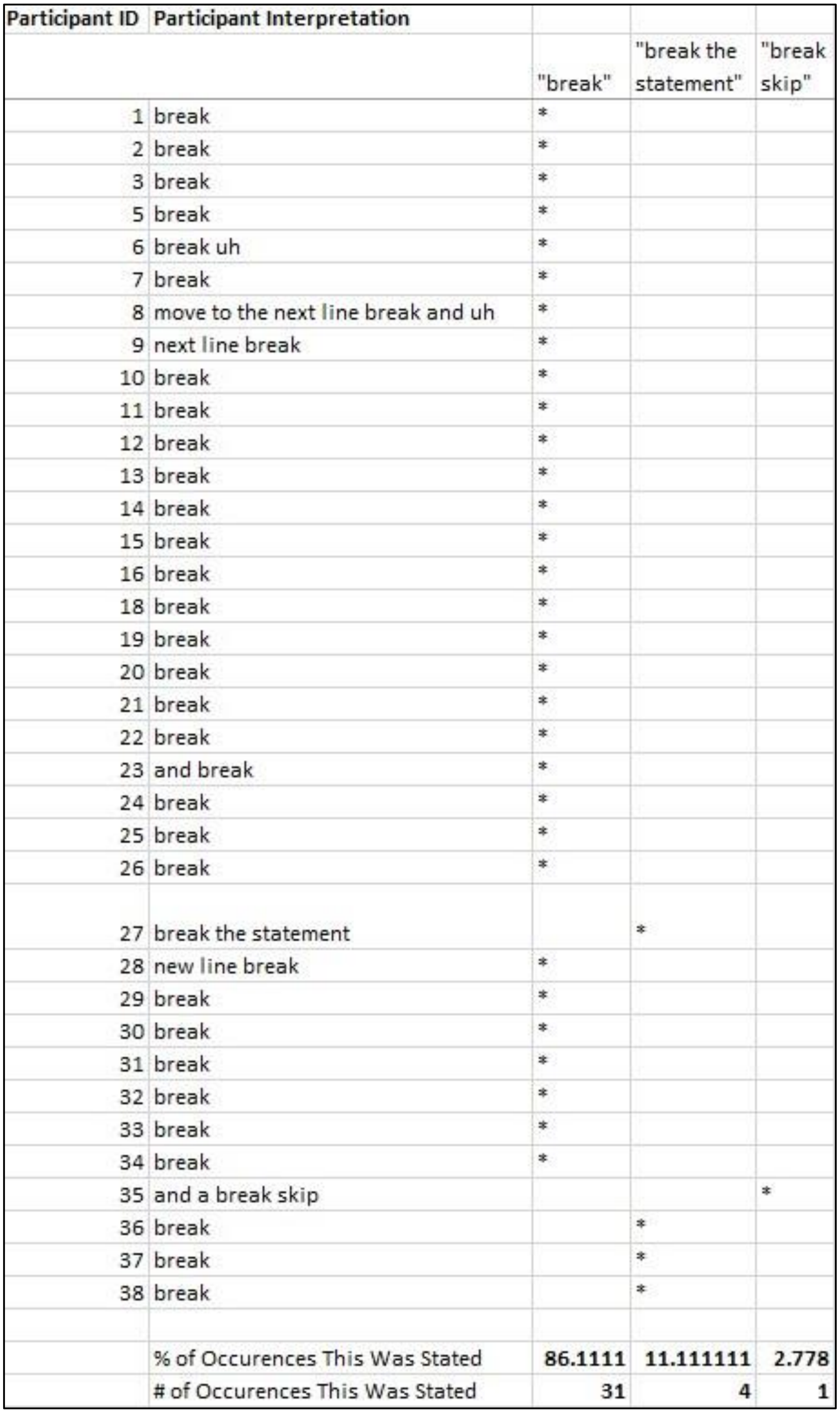
Participant ID	Participant Interpretation					
		"colon"	Did not state anything	"semi colon"	"end of colon"	"two dots"
36	default		*			
37	and then case one taco and case taco		*			
37	case rice		*			
37	default		*			
38	case taco		*			
38	case rice		*			
38	default		*			
	% of Occurences This Was Stated	**30.556**	**60.18519**	**7.4074**	**0.9259**	**0.926**
	# of Occurences This Was Stated	**33**	**65**	**8**	**1**	**1**

E.19) default



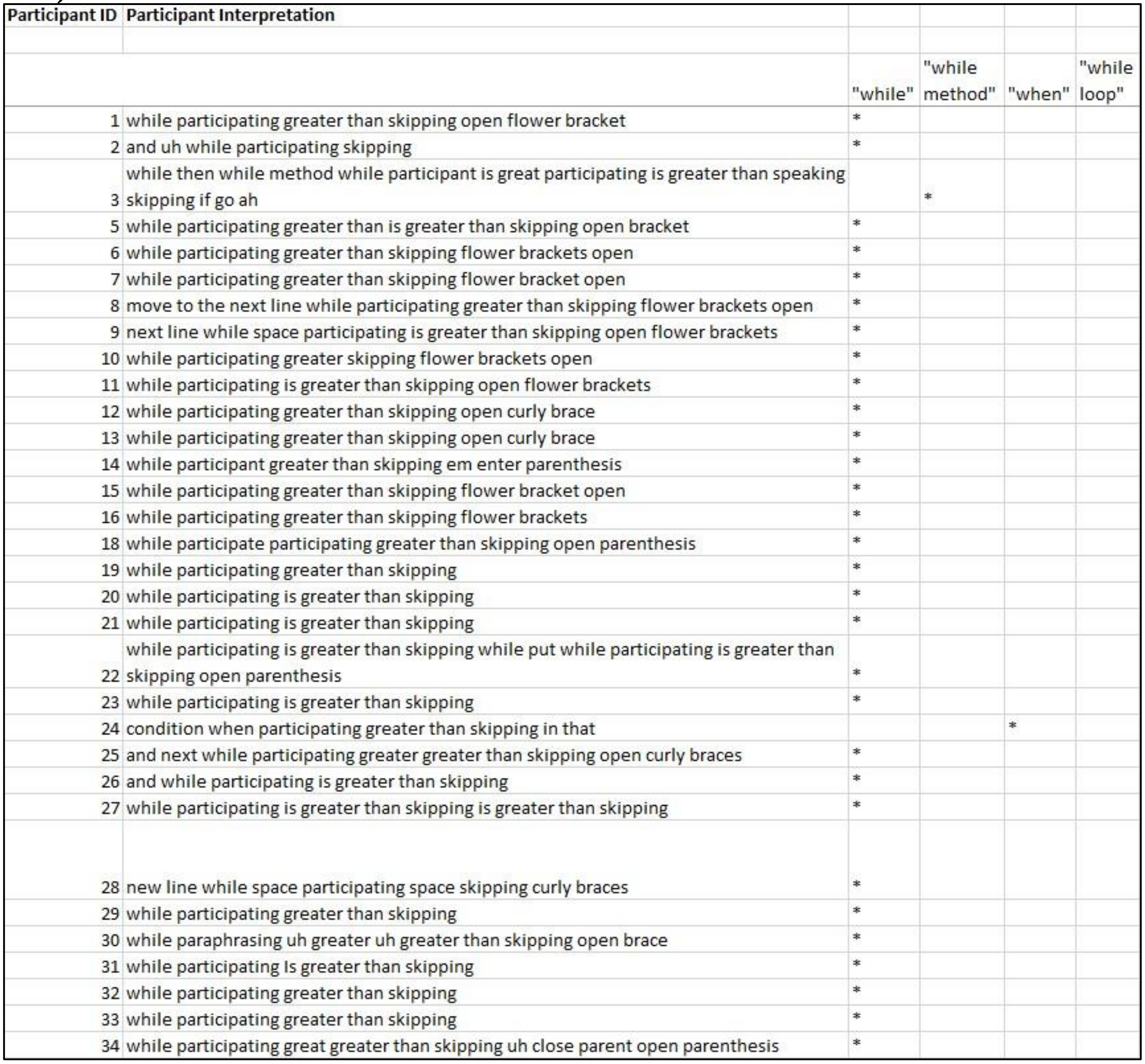
Participant ID	Participant Interpretation						
		"default"	"else default"	"and default"	"default statement"	"by default"	"add a default"
1	default colon	*					
2	default	*					
3	default	*					
5	default quotes default colon	*					
6	default colon	*					
7	default semi colon	*					
8	move to the next line default colon uh	*					
9	next line default colon	*					
10	default	*					
11	default colon	*					
12	default colon	*					
13	default colon	*					
14	else default		*				
15	default	*					
16	default	*					
18	default	*					
19	default	*					
20	default	*					
21	default	*					
22	default colon	*					
23	and default			*			
24	and default			*			
25	next default statement colon				*		
26	and by default					*	
27	default	*					
28	new line default colon	*					
29	default	*					
30	default semi colon	*					
31	default	*					
32	default	*					
33	default	*					
34	default uh colon	*					
35	add a default						*
36	default	*					
37	default	*					
38	default	*					
	% of Occurences This Was Stated	**83.3333**	**2.7778**	**5.5556**	**2.7777778**	**2.7778**	**2.7778**
	# of Occurences This Was Stated	**30**	**1**	**2**	**1**	**1**	**1**

E.20) break

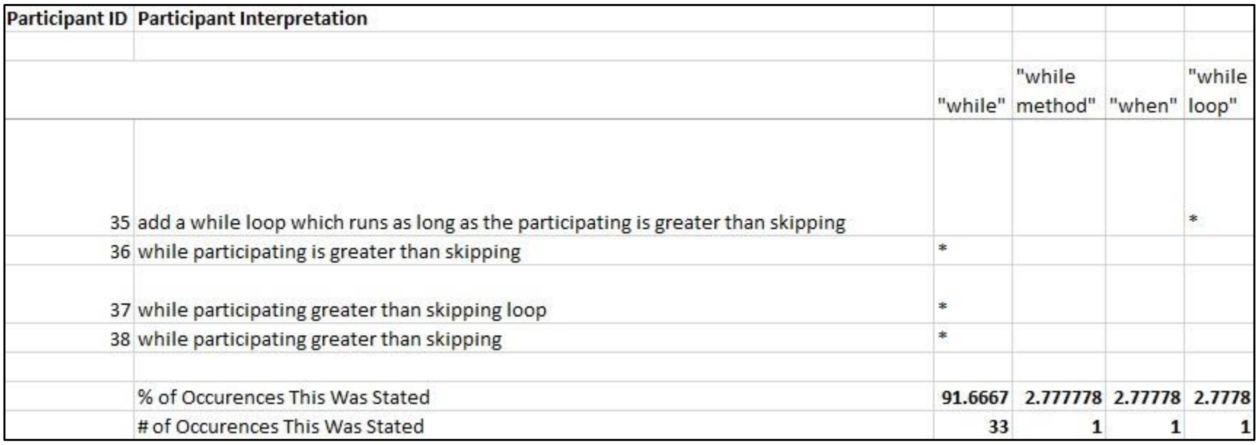


Participant ID	Participant Interpretation			
		"break"	"break the statement"	"break skip"
1	break	*		
2	break	*		
3	break	*		
5	break	*		
6	break uh	*		
7	break	*		
8	move to the next line break and uh	*		
9	next line break	*		
10	break	*		
11	break	*		
12	break	*		
13	break	*		
14	break	*		
15	break	*		
16	break	*		
18	break	*		
19	break	*		
20	break	*		
21	break	*		
22	break	*		
23	and break	*		
24	break	*		
25	break	*		
26	break	*		
27	break the statement		*	
28	new line break	*		
29	break	*		
30	break	*		
31	break	*		
32	break	*		
33	break	*		
34	break	*		
35	and a break skip			*
36	break		*	
37	break		*	
38	break		*	
	% of Occurences This Was Stated	**86.1111**	**11.111111**	**2.778**
	# of Occurences This Was Stated	**31**	**4**	**1**

E.21) while

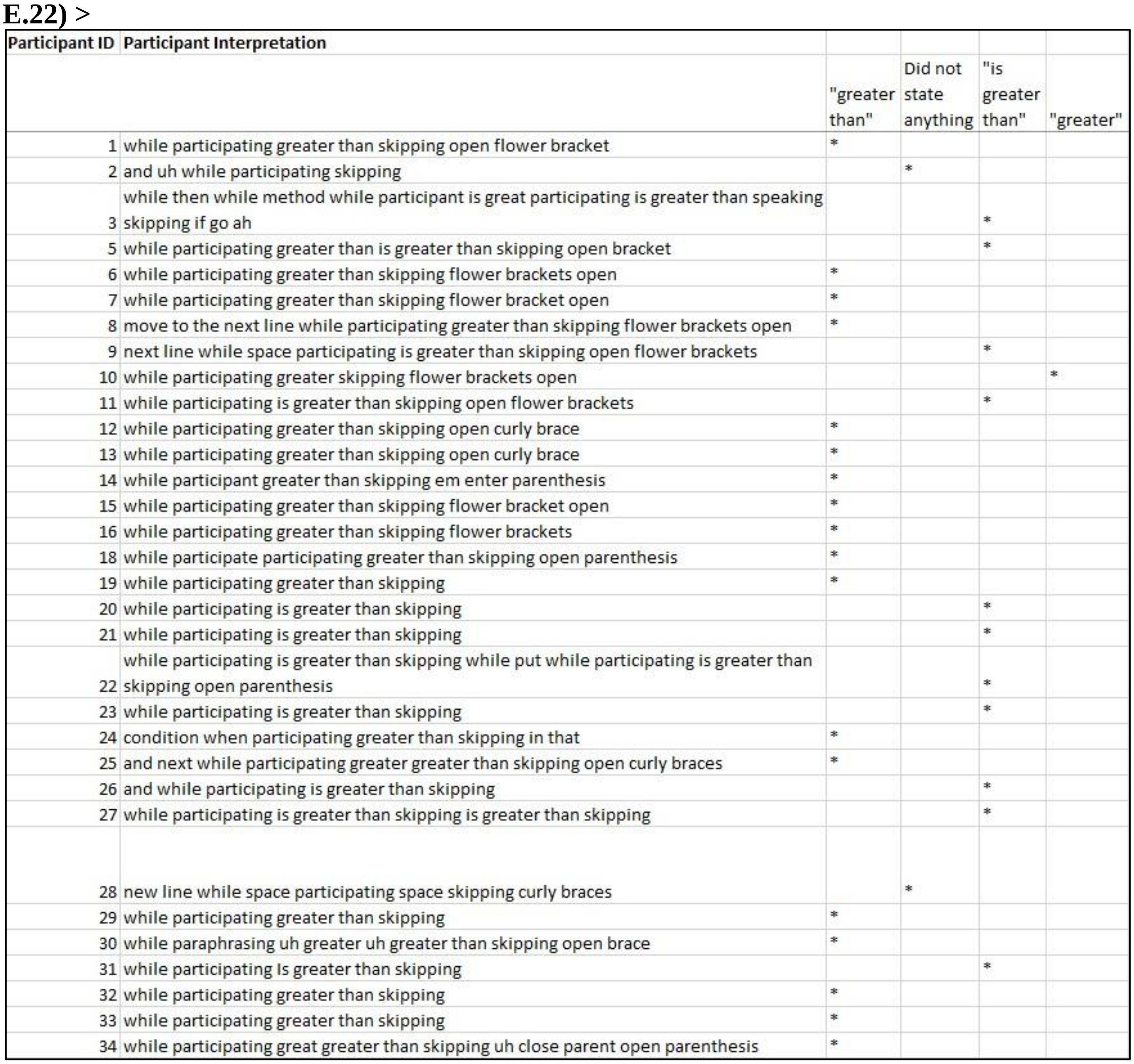


Participant ID	Participant Interpretation	"while"	"while method"	"when"	"while loop"
1	while participating greater than skipping open flower bracket	*			
2	and uh while participating skipping	*			
3	while then while method while participant is great participating is greater than speaking skipping if go ah		*		
5	while participating greater than is greater than skipping open bracket	*			
6	while participating greater than skipping flower brackets open	*			
7	while participating greater than skipping flower bracket open	*			
8	move to the next line while participating greater than skipping flower brackets open	*			
9	next line while space participating is greater than skipping open flower brackets	*			
10	while participating greater skipping flower brackets open	*			
11	while participating is greater than skipping open flower brackets	*			
12	while participating greater than skipping open curly brace	*			
13	while participating greater than skipping open curly brace	*			
14	while participant greater than skipping em enter parenthesis	*			
15	while participating greater than skipping flower bracket open	*			
16	while participating greater than skipping flower brackets	*			
18	while participate participating greater than skipping open parenthesis	*			
19	while participating greater than skipping	*			
20	while participating is greater than skipping	*			
21	while participating is greater than skipping	*			
22	while participating is greater than skipping while put while participating is greater than skipping open parenthesis	*			
23	while participating is greater than skipping	*			
24	condition when participating greater than skipping in that			*	
25	and next while participating greater greater than skipping open curly braces	*			
26	and while participating is greater than skipping	*			
27	while participating is greater than skipping is greater than skipping	*			
28	new line while space participating space skipping curly braces	*			
29	while participating greater than skipping	*			
30	while paraphrasing uh greater uh greater than skipping open brace	*			
31	while participating Is greater than skipping	*			
32	while participating greater than skipping	*			
33	while participating greater than skipping	*			
34	while participating great greater than skipping uh close parent open parenthesis	*			

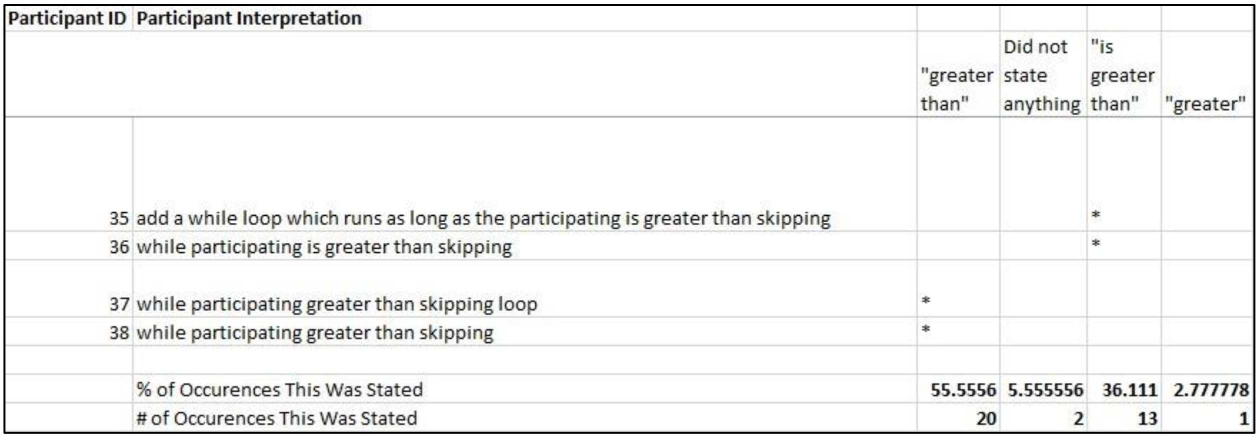


Participant ID	Participant Interpretation				
		"while"	"while method"	"when"	"while loop"
35	add a while loop which runs as long as the participating is greater than skipping				*
36	while participating is greater than skipping	*			
37	while participating greater than skipping loop	*			
38	while participating greater than skipping	*			
	% of Occurences This Was Stated	**91.6667**	**2.777778**	**2.77778**	**2.7778**
	# of Occurences This Was Stated	**33**	**1**	**1**	**1**

E.22) >

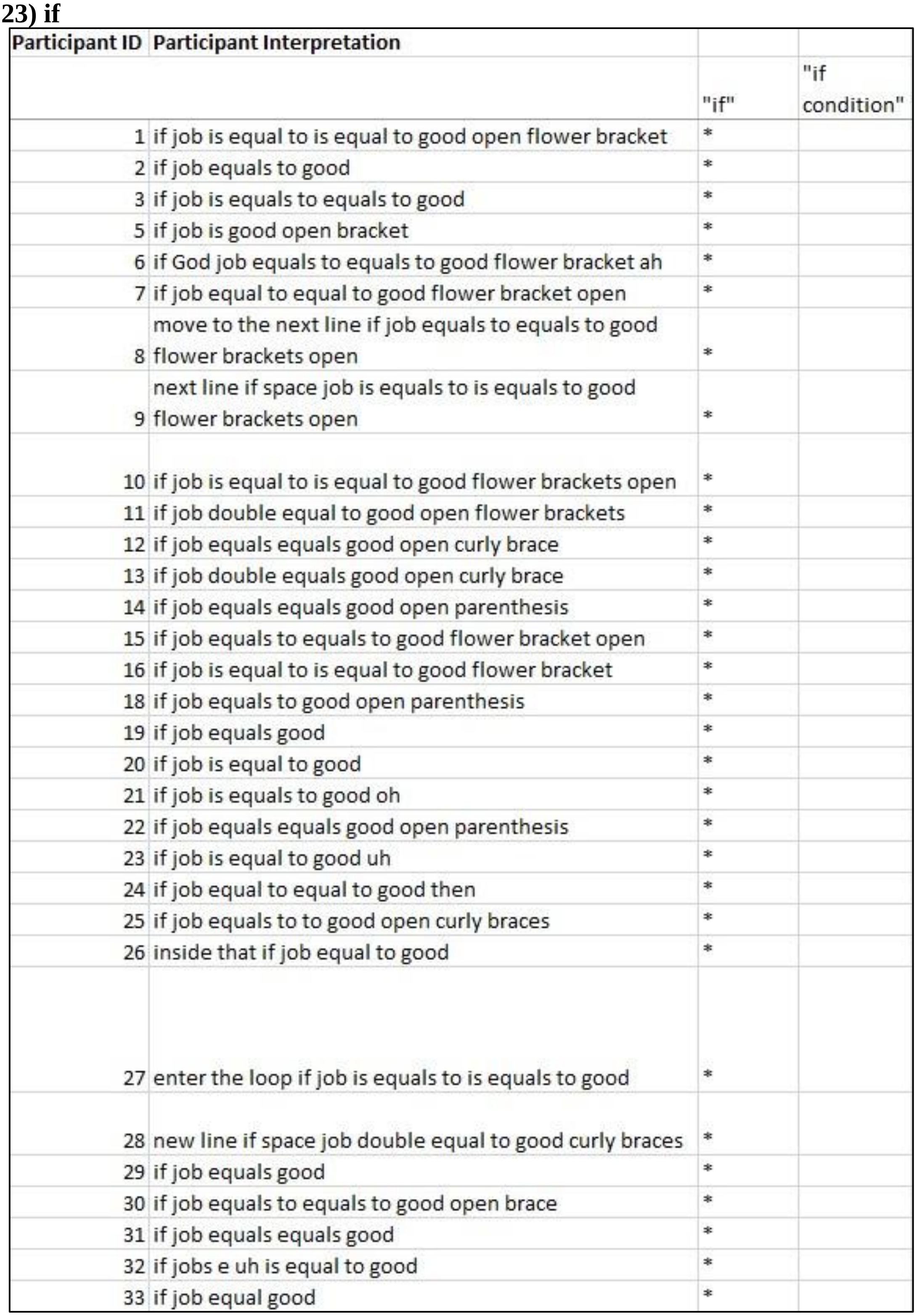


Participant ID	Participant Interpretation				
		"greater than"	Did not state anything	"is greater than"	"greater"
1	while participating greater than skipping open flower bracket	*			
2	and uh while participating skipping		*		
3	while then while method while participant is great participating is greater than speaking skipping if go ah			*	
5	while participating greater than is greater than skipping open bracket			*	
6	while participating greater than skipping flower brackets open	*			
7	while participating greater than skipping flower bracket open	*			
8	move to the next line while participating greater than skipping flower brackets open	*			
9	next line while space participating is greater than skipping open flower brackets			*	
10	while participating greater skipping flower brackets open				*
11	while participating is greater than skipping open flower brackets			*	
12	while participating greater than skipping open curly brace	*			
13	while participating greater than skipping open curly brace	*			
14	while participant greater than skipping em enter parenthesis	*			
15	while participating greater than skipping flower bracket open	*			
16	while participating greater than skipping flower brackets	*			
18	while participate participating greater than skipping open parenthesis	*			
19	while participating greater than skipping	*			
20	while participating is greater than skipping			*	
21	while participating is greater than skipping			*	
22	while participating is greater than skipping while put while participating is greater than skipping open parenthesis			*	
23	while participating is greater than skipping			*	
24	condition when participating greater than skipping in that	*			
25	and next while participating greater greater than skipping open curly braces	*			
26	and while participating is greater than skipping			*	
27	while participating is greater than skipping is greater than skipping			*	
28	new line while space participating space skipping curly braces		*		
29	while participating greater than skipping	*			
30	while paraphrasing uh greater uh greater than skipping open brace	*			
31	while participating Is greater than skipping			*	
32	while participating greater than skipping	*			
33	while participating greater than skipping	*			
34	while participating great greater than skipping uh close parent open parenthesis	*			

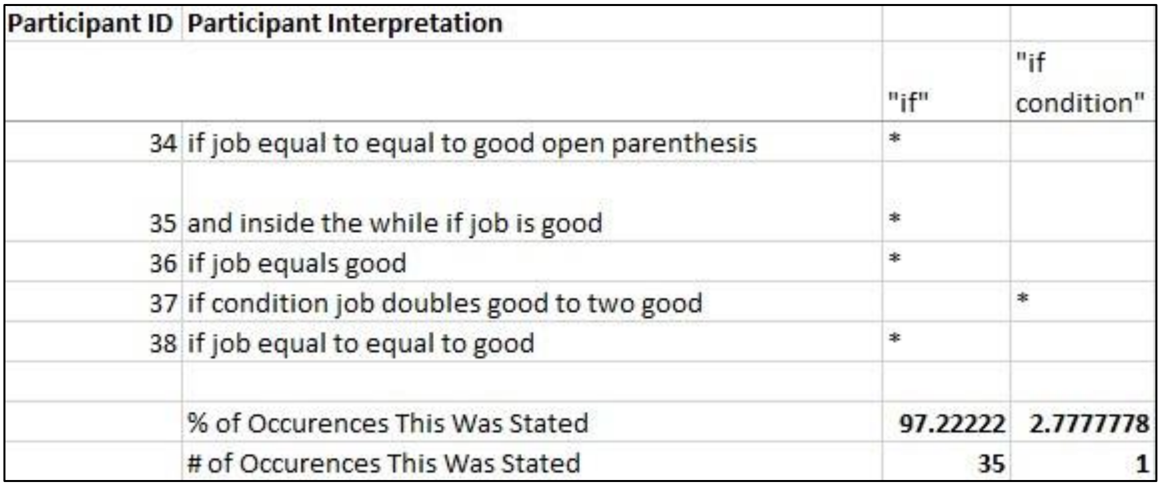


Participant ID	Participant Interpretation				
		"greater than"	Did not state anything	"is greater than"	"greater"
35	add a while loop which runs as long as the participating is greater than skipping			*	
36	while participating is greater than skipping			*	
37	while participating greater than skipping loop	*			
38	while participating greater than skipping	*			
	% of Occurences This Was Stated	**55.5556**	**5.555556**	**36.111**	**2.777778**
	# of Occurences This Was Stated	**20**	**2**	**13**	**1**

E.23) if

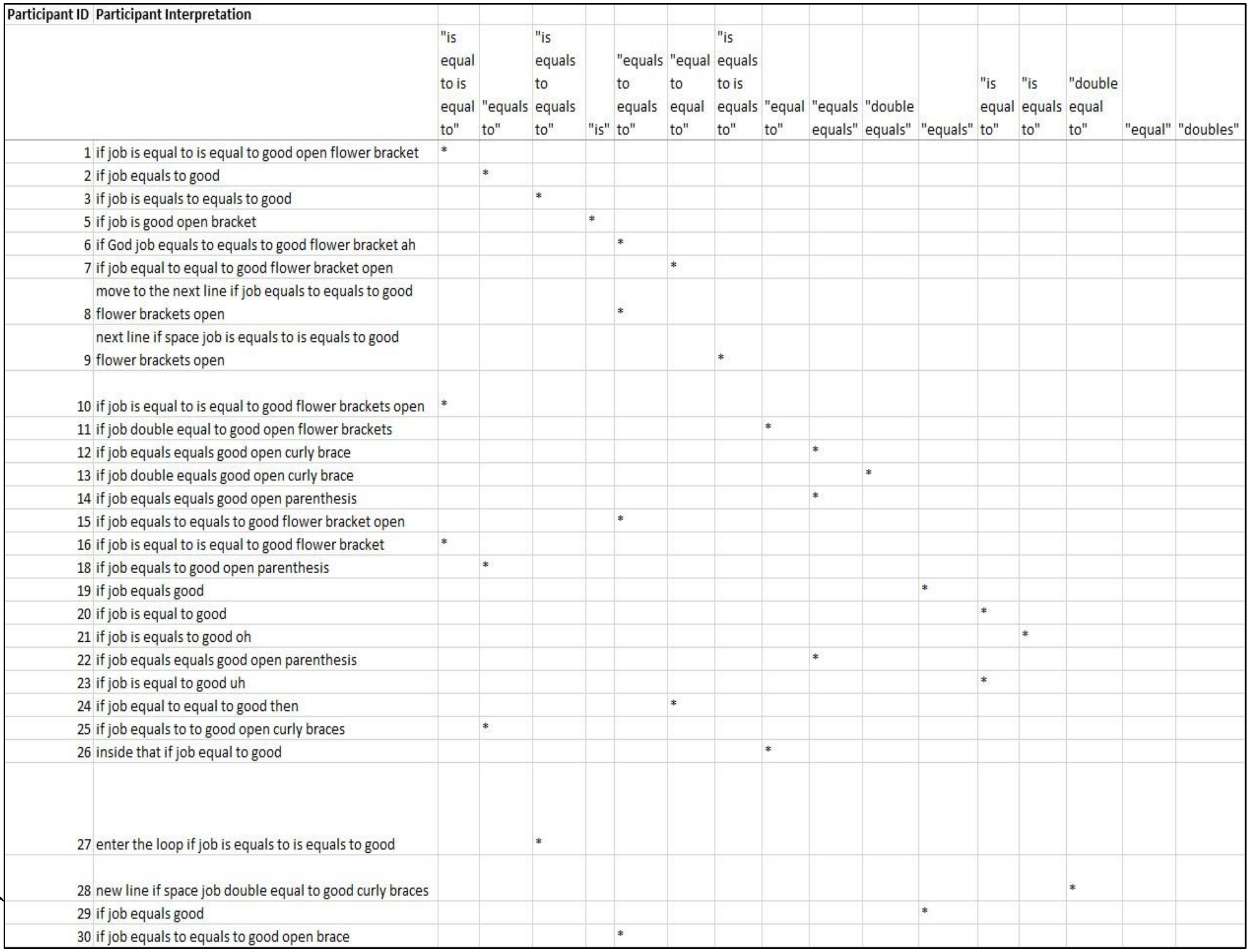


Participant ID	Participant Interpretation		
		"if"	"if condition"
1	if job is equal to is equal to good open flower bracket	*	
2	if job equals to good	*	
3	if job is equals to equals to good	*	
5	if job is good open bracket	*	
6	if God job equals to equals to good flower bracket ah	*	
7	if job equal to equal to good flower bracket open	*	
8	move to the next line if job equals to equals to good flower brackets open	*	
9	next line if space job is equals to is equals to good flower brackets open	*	
10	if job is equal to is equal to good flower brackets open	*	
11	if job double equal to good open flower brackets	*	
12	if job equals equals good open curly brace	*	
13	if job double equals good open curly brace	*	
14	if job equals equals good open parenthesis	*	
15	if job equals to equals to good flower bracket open	*	
16	if job is equal to is equal to good flower bracket	*	
18	if job equals to good open parenthesis	*	
19	if job equals good	*	
20	if job is equal to good	*	
21	if job is equals to good oh	*	
22	if job equals equals good open parenthesis	*	
23	if job is equal to good uh	*	
24	if job equal to equal to good then	*	
25	if job equals to to good open curly braces	*	
26	inside that if job equal to good	*	
27	enter the loop if job is equals to is equals to good	*	
28	new line if space job double equal to good curly braces	*	
29	if job equals good	*	
30	if job equals to equals to good open brace	*	
31	if job equals equals good	*	
32	if jobs e uh is equal to good	*	
33	if job equal good	*	

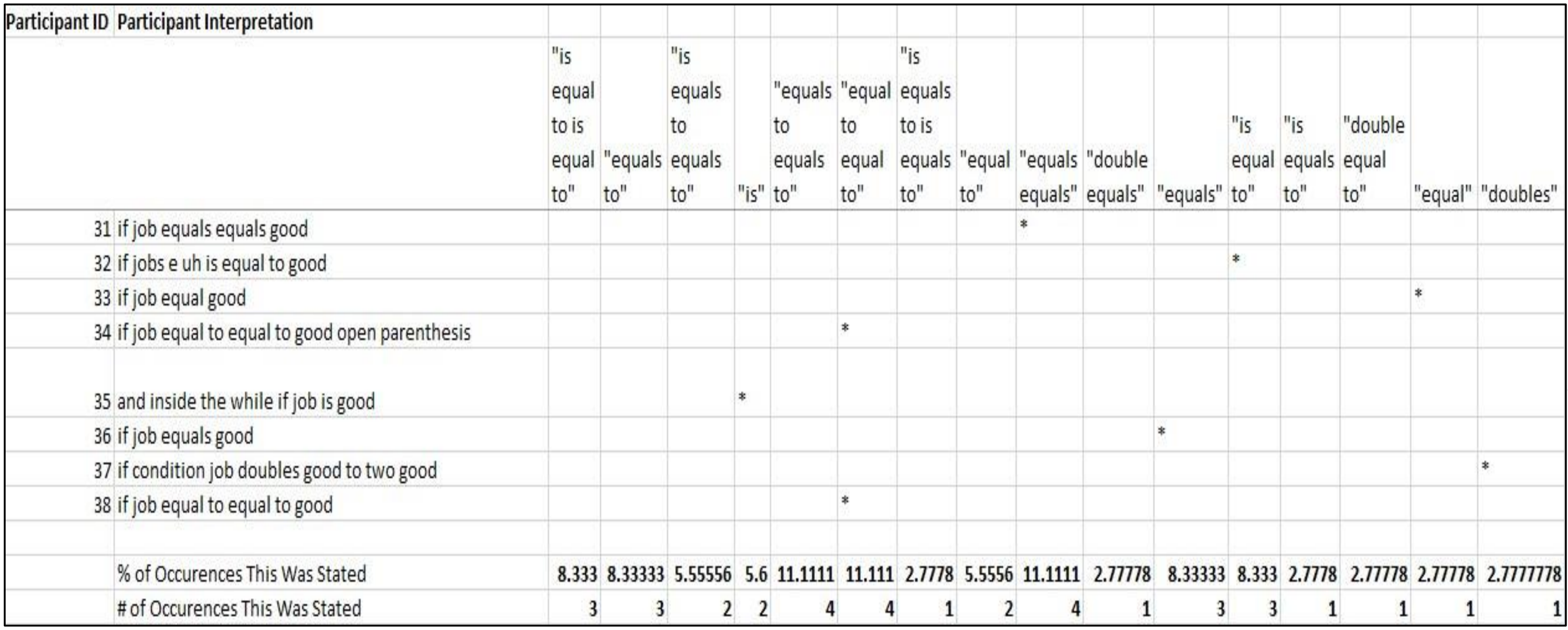


Participant ID	Participant Interpretation		
		"if"	"if condition"
34	if job equal to equal to good open parenthesis	*	
35	and inside the while if job is good	*	
36	if job equals good	*	
37	if condition job doubles good to two good		*
38	if job equal to equal to good	*	
	% of Occurences This Was Stated	**97.22222**	**2.7777778**
	# of Occurences This Was Stated	**35**	**1**

E.24) ==

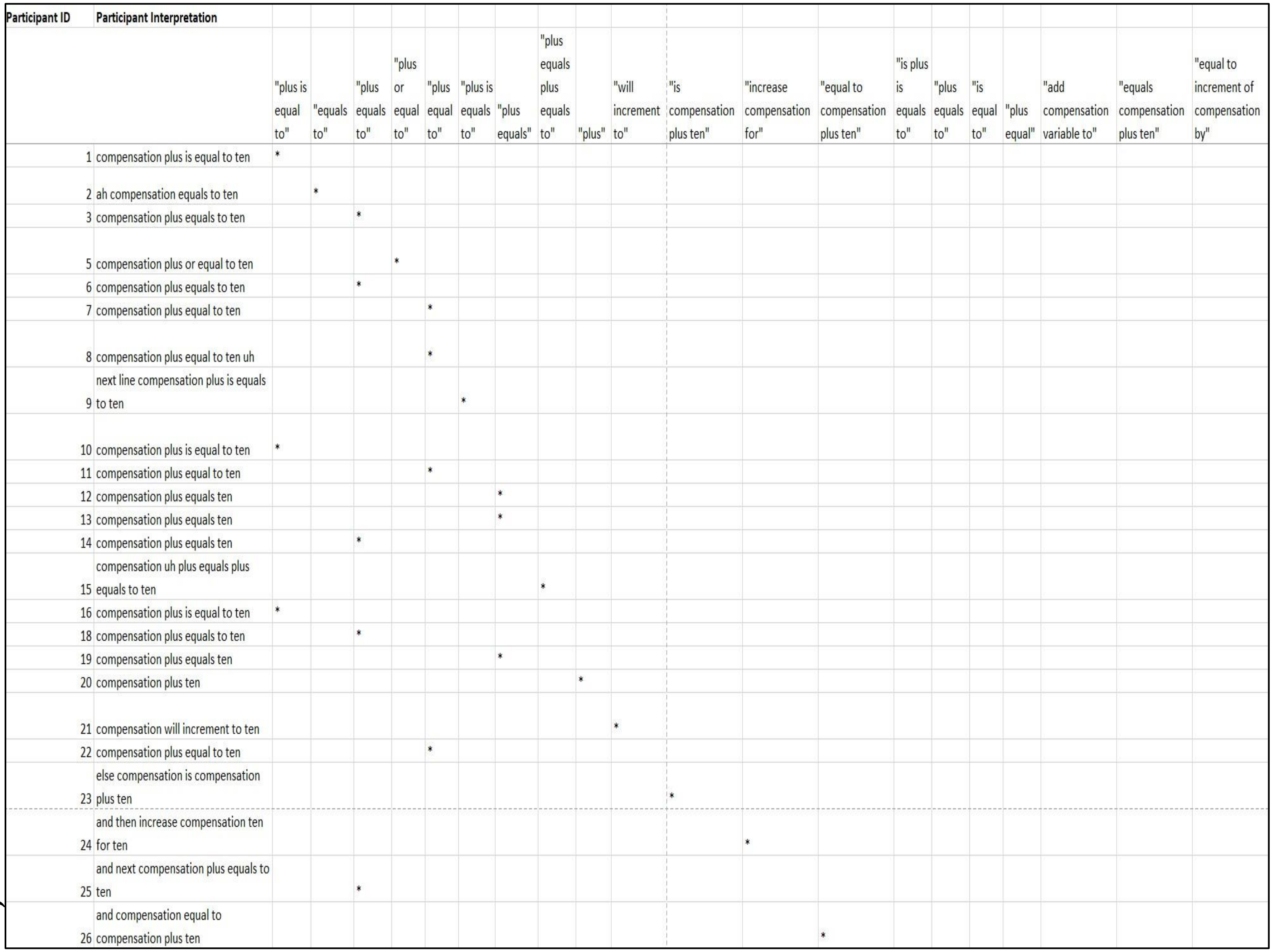


Participant ID	Participant Interpretation																
		"is equal to is equal to"	"equals to"	"is equals to equals to"	"is"	"equals to equals to"	"equal to equal to"	"is equals to is equals to"	"equal to"	"equals equals"	"double equals"	"equals"	"is equal to"	"is equals to"	"double equal to"	"equal"	"doubles"
1	if job is equal to is equal to good open flower bracket	*															
2	if job equals to good		*														
3	if job is equals to equals to good			*													
5	if job is good open bracket				*												
6	if God job equals to equals to good flower bracket ah					*											
7	if job equal to equal to good flower bracket open						*										
8	move to the next line if job equals to equals to good flower brackets open					*											
9	next line if space job is equals to is equals to good flower brackets open							*									
10	if job is equal to is equal to good flower brackets open	*															
11	if job double equal to good open flower brackets								*								
12	if job equals equals good open curly brace									*							
13	if job double equals good open curly brace										*						
14	if job equals equals good open parenthesis									*							
15	if job equals to equals to good flower bracket open					*											
16	if job is equal to is equal to good flower bracket	*															
18	if job equals to good open parenthesis		*														
19	if job equals good											*					
20	if job is equal to good												*				
21	if job is equals to good oh													*			
22	if job equals equals good open parenthesis									*							
23	if job is equal to good uh												*				
24	if job equal to equal to good then						*										
25	if job equals to to good open curly braces		*														
26	inside that if job equal to good								*								
27	enter the loop if job is equals to is equals to good			*													
28	new line if space job double equal to good curly braces														*		
29	if job equals good											*					
30	if job equals to equals to good open brace					*											



Participant ID	Participant Interpretation																
		"is equal to is equal to"	"equals to"	"is equals to equals to"	"is"	"equals to equals to"	"equal to equal to"	"is equals to is equals to"	"equal to"	"equals equals"	"double equals"	"equals"	"is equal to"	"is equals to"	"double equal to"	"equal"	"doubles"
31	if job equals equals good									*							
32	if jobs e uh is equal to good												*				
33	if job equal good															*	
34	if job equal to equal to good open parenthesis						*										
35	and inside the while if job is good				*												
36	if job equals good											*					
37	if condition job doubles good to two good																*
38	if job equal to equal to good						*										
	% of Occurences This Was Stated	8.333	8.33333	5.55556	5.6	11.1111	11.111	2.7778	5.5556	11.1111	2.77778	8.33333	8.333	2.7778	2.77778	2.77778	2.7777778
	# of Occurences This Was Stated	3	3	2	2	4	4	1	2	4	1	3	3	1	1	1	1

E.25) +=



Participant ID	Participant Interpretation	"plus is equal to"	"equals to"	"plus equals to"	"plus or equal to"	"plus equal to"	"plus is equals to"	"plus equals"	"plus equals plus equals to"	"plus"	"will increment to"	"is compensation plus ten"	"increase compensation for"	"equal to compensation plus ten"	"is plus is equals to"	"plus equals to"	"is equal to"	"plus equal"	"add compensation variable to"	"equals compensation plus ten"	"equal to increment of compensation by"
1	compensation plus is equal to ten	*																			
2	ah compensation equals to ten		*																		
3	compensation plus equals to ten			*																	
5	compensation plus or equal to ten				*																
6	compensation plus equals to ten			*																	
7	compensation plus equal to ten					*															
8	compensation plus equal to ten uh					*															
9	next line compensation plus is equals to ten						*														
10	compensation plus is equal to ten	*																			
11	compensation plus equal to ten					*															
12	compensation plus equals ten							*													
13	compensation plus equals ten							*													
14	compensation plus equals ten			*																	
15	compensation uh plus equals plus equals to ten								*												
16	compensation plus is equal to ten	*																			
18	compensation plus equals to ten			*																	
19	compensation plus equals ten							*													
20	compensation plus ten									*											
21	compensation will increment to ten										*										
22	compensation plus equal to ten					*															
23	else compensation is compensation plus ten											*									
24	and then increase compensation ten for ten												*								
25	and next compensation plus equals to ten			*																	
26	and compensation equal to compensation plus ten													*							

Participant ID	Participant Interpretation																				
		"plus is equal to"	"equals to"	"plus equals to"	"plus or equal to"	"plus equal to"	"plus is equals to"	"plus equals"	"plus equals plus equals to"	"plus"	"will increment to"	"is compensation plus ten"	"increase compensation for"	"equal to compensation plus ten"	"is plus is equals to"	"plus equals to"	"is equal to"	"plus equal"	"add compensation variable to"	"equals compensation plus ten"	"equal to increment of compensation by"
27	compensation is plus is equals to ten														*						
28	new line compensation space plus is equal to ten curly braces						*														
29	compensation plus equals ten							*													
30	competition plus equals to ten															*					
31	compensation plus equals ten							*													
32	compensation rare is equal to ten																*				
33	compensation plus equal ten																	*			
34	compensation plus uh plus or equal to ten				*																
35	add oh compensation variable to ten																		*		
36	and compensation equals compensation plus ten																			*	
37	compensation equal to increment of compensation by ten																				*
38	compensation equal to compensation plus ten													*							
	% of Participant That Stated This	**8.3333**	**2.7778**	**13.89**	**5.556**	**11.11**	**5.556**	**13.889**	**2.778**	**2.778**	**2.7777778**	**2.777777778**	**2.777777778**	**5.555555556**	**2.778**	**2.778**	**2.778**	**2.778**	**2.777777778**	**2.777777778**	**2.777777778**
	# of Particpants That Stated This	**3**	**1**	**5**	**2**	**4**	**2**	**5**	**1**	**1**	**1**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

APPENDIX F:

APPLE'S SPEECH TO TEXT TRANSCRIPTIONS

Participant ID	**Manual Transcription**
1	Zero one open flower bracket print (double: hello space word double quotes) 4X021 open flat bracket where space test is equal to 10+ lower bracket close la bracket switch (food clothes (open fly back in case taco: good is equal to two case price: good as equal to kiss catch up good as equal to default: break close flop racket while participating greater than skipping open flower packet if job is equal to is equal to God open flower bracket compensation plus is equal to 10 clothes flat bracket
2	Fun 021921120 available test done yes rice balls deep on break and while participating skipping equals to a girl like you very much compensation
3	4021 second I will be print hello 140211S equals to 10 second method switch for the tacos good equals to two rice good equals two testing will be catch-up default break jacket by then why participant participating is greater than speaking skipping if job is equal to equal to Goode Bird response thank you very much

Participant ID	Manual Transcription
5	4_0... One [print parentheses (quotations hello world“ Tatian's) 4XO... One [variable test equals 10]] switch (food) [Case "Tatian taco plus quotations: good equals true case open quotations rice close quotation: good equals true case open quotation catch up close quotation good eagles Falls default codes while participating greater than greater than skipping open bracketif job is good [variable response equals
	""Tatian thank you very much "" Tatian] compensation plus or equal to 10]
6	For_O to a club bracket open printf double coats hello world the double clothes clothes in the fireplace for ex 021 progress where is equals to 10 fabricate fabricate switch Ford case double coats taco double boards;: good equals to rise case: rice calling clothes in the colon good equals to two case double coat ketchup double coats good equals two falls default: break up close of racist while participating greater than skipping job equals request to get the bracket where response equals two double coats thank you very much double coats clothes in the flowerbeds compensation plus equals to 10 and the clothes from Brix

7	For_0... One flower bracket open print bracket open hello bracket close for XO... One floor bracket open we are best equal to 10 flow better clothes lol bracket close switch bracket open for bracket close flower bracket open case taco; good equal to two KS right; good equal true yes catch-up good equal to Falls default; break from bracket close while participating greater than skipping floor bracket open jobs equal to equal to pour floor bracket open we are responsible to thank you very much my back and close compensation plus equal to 10 floor bracket close
8	4_021 La Brisa open motor the next line print bases open courts open hello space world coats clothes clothes go to the next line 4X021 bases open where is equal to 10 on Florida schools in Florida clothes switch bases open food banks close floor floor bracket open guess courts

Participant ID	**Manual Transcription**
	open taco coats clothes: what is an excellent good equal to true excellent case courts open race course close: equal to throw guess coats coats clothes what a 1 excellent good equal to Falls before: it's close to the next little while what is putting brighter than skipping floor bracket open job equals two equals to good floor bracket open motor the next one where response equal 2 quarts open thank you very much coach close with a excellent products close compensation plus equal to 10

9	4_03. One open flat brackets Print (double condition hello space world double quotation) Four bass X space 03. One lower brackets What space test is equal to 10 Clothesline rackets rackets switch (third) flower bracket Case double quotation tackle double quotation: Good is equal to Case double condition rice double quotation: God is equal to two God is equal to false next line default: Brake Flower bracketing While space participating is greater than skipping open flower brackets F space job is equal to is equal still good so brackets open

Participant ID	**Manual Transcription**

	Where response IS equals to double quotation thank you very much double creation Clothes for our records Compensation plus is equals to 10 Close flowerbeds
10	45021 open print hello world for X021 flow back it's open bar test is equal to 10 it's close switch of food is open case tackle call good is equal is is true case rice good is true case ketchup goodies falls default break flower bracket screws while participating greater skipping flower brackets open if job is equal to is equal to God flow rack is open why response is thank you very much lol bracket screws compensation plus is equal to
11	For_021 open services print (in the quotations for X021 open floor brackets that space test equal to 10) and switch off phone switch (four) open floor bracket case; good case rise: good equal to cook default: break while participating is greater than Open floor bracket where response is assigned to thank you very much compensation plus equal to 10 close the products

12	For_0 until one {friend (hello world in the inverted quotes) for X in zero until one { where test equals 10}} switch (food) case inverted golden tackle inverted calling and calling good equals true case and what did call Rice inverted call me call me good equals true case of an audit call default column break} while participating greater than skipping { if

Participant ID	**Manual Transcription**
	job equals equals good { where response equals inverted codes thank you very much inverted codes} compensation less equals 10}
13	_Zero... One open print (double coats hello close double course close balances for XO... One { that equals 10} clothes switch (four) { case l will go to tackle: good equals trueGood equals true case double coats good equals falls before: break} while participating greater than skipping (job double equals good { where response equals double force thank you very much} compensation plus equals 10
14	For weight 021 (print; hello world close; 4X021 (where rebel test equal to 10)) six parentheses food); (case tackle; add to that good equals the true next case next case catch up good equals the falls as default break) well participant greater than skipping into Brandon if job equals equals good (variable response equals thank you very much) compensation plus equals 10

15	For 021 fly bracket open print hello for X021 club bracket open wide test equals to 10 club like a closed flat black at the clothes switch foot flat bucket open case taco good is equals to two in case rice good is able to draw a sketch of good girls to Falls default break flat bracket broke clothes while party speeding greater than skipping play bracket open if job equals two equals two good plow bracket open wide response is equal to thank you so much thank you very much club bracket close compensation plus Club bracket closed

Participant ID	**Manual Transcription**
16	_021 flava rackets print hello world for X021 La brackets were test is equal to 10 switch food case taco good is equal true case rice good is equal true case ketchup what is equal to Falls default break lol brackets while participative greater than sleeping blah blah kids if job is equal to is equal a good one response is equal to thank you very much fabric compensation plus is equal to 10 club racket
18	_Zero... One (print hello world follow up x-ray... One (that testicles to 10)) switch off food (case tackle go request a true case rice request a true case ketchup falls default break) why participate participating greater than skipping open finances if job equals two good (where response equals to thank you very much) compensation plus equals to 10)

19	For_ 021 print hello world for X02 on that test equals 10 Switchfoot case taco go to Scholes true case right good equals true case ketchup good equals fault default break while parties participating grade and keeping his job is called Goode where respond to calls thank you very much compensation plus equals 10
20	40214X0 toone variable test is equal te zero switch for the stucco guy is rice good is going through a sketch of Cruz going to force the fall break while participating is greater than skipping this job is equal to good variable response is equal to thank you very much compensation +10
21	OK 4021 print Hallawood 4X0210 switch for testicle go to Picosito case Rice go request to schedule a false default break while

Participant ID	**Manual Transcription**
	participating in skipping if job is equal to the girls response equals to thank you very much compensation

22	For a four I go to zero to one parentheses open print parentheses "hello world" parentheses 4X goes from 0 to 1 parenthesis open fire test equal to 10)) switch parentheses open food) (case double coats taco double coats clothes: good equals a true case double codes race": good equals a true case double coats open ketchup double coats clothes good equal to Falls default: break parentheses close while part while part while participating is greater than skipping (if job equals equals good (why response equal to open "thank you very much") compensation plus equal to 10)
23	400 to one or open the loop print hello world for zero to one variable test is equal to 10 close or closer to four loads switch case food case one taco case to is rice good to case ketchup good is false and default and break while participating is greater than skipping if job is equal if to good response iS thank you very much else compensation based compensation +10
24	45214821 wearable testicle to 10 and two brackets close enough that switch case we use food as a keyboard in that case taco taco good control the next case rice and case number three touchup would equal to Paul's condition by participating in that job than responsive I mean thank you very much and then increase compensation then

Participant ID	**Manual Transcription**

25	Oh fart "addresses print (close the prices for X021 open collie braces wire test equals to 10} and I can close the curly braces and switch condition (food {his keys and codes that go in the next time good equals to draw in another case open open quotations rice: next line good equals to draw an excuse "Tatian with ketchup Break} us while participating greater than skipping { this job equals 2 { his response equals door and open quotation quotations thank you very much and makes compensation plus equals to 10
26	Fire follow from zero to one print hello one inside that folder for X equal to zero to one inside that bad taste equal to 10 take a switch case take input food as an variable and in the case one it should be taco and good equal to two sequel to rice Goode equals a true case equal to catch up good equal to Falls and by default break that's it I am wild parties painting is greater than skipping inside that if job equal to go to where variable response equal to thank you so very much and compensation is equal to
27	For_0 one open races hello world" for XO one open but just as equals to 10 again and lube switch for { this case tackle good is equals to two case" rice look good is equals to two case ketchup in double coat good is equal to Falls default break the statement white participating is greater than skipping job is equal to is equal to good open the loop that response is equals to thank you very much" in the loop compensation

Participant ID	Manual Transcription
28	4_000... One of the places Bryant brackets hello world double coats racket that's fine for XO... One called braces where space test is equal to one zero qualifications converses space next line switch bracket food bracket space Cody braces next line case gods tackle double quotes: newline good is it cold or two newline case double coats rice double codes: newline good is equal to two double codes Good is equal to falls Default: Brake caliber size While space participating space skipping college prices F job double equal to good quality races Where space response is it going to double codes thank you very much W goats Calibrations Compensation space plus is equal to 10: braces
29	For_0 to one friend hello world for Gia to one variable test equals 10 switch phone cases one tackle the request through a store rice correct. Case we catch up coequal Sioux Falls default rate while participating better than skipping F job equals good very good response thank you very much compensation plus equals 10

30	4_021 open races print [hello world enter 4XX021 open dress code test is equal to 10})(food or { case taco double coats topcoat good equals to true case double coats rice; good equals to true case double coats

Participant ID	Manual Transcription
	ketchup good equals two falls default; break close the brace while paraphrasing greater than skipping { its job equals to thank you very much } competition plus equals to 10}
31	For some integer between zero and one print hello world for some manager acts between zero and one variable test equals zero switch food case taco good equals true case race good equals true case catch-up good equals true default break while participating is greater than skipping his job equals equals good variable response equals thank you very much compensation plus equals 10
32	4021 print hello world 4X021 by test equals 10 switch food case taco good equals true case rice good equals two case ketchup good equals force default break while participating greater than skipping if job is equal to good bad response equals thank you very much compensation is equal to 10

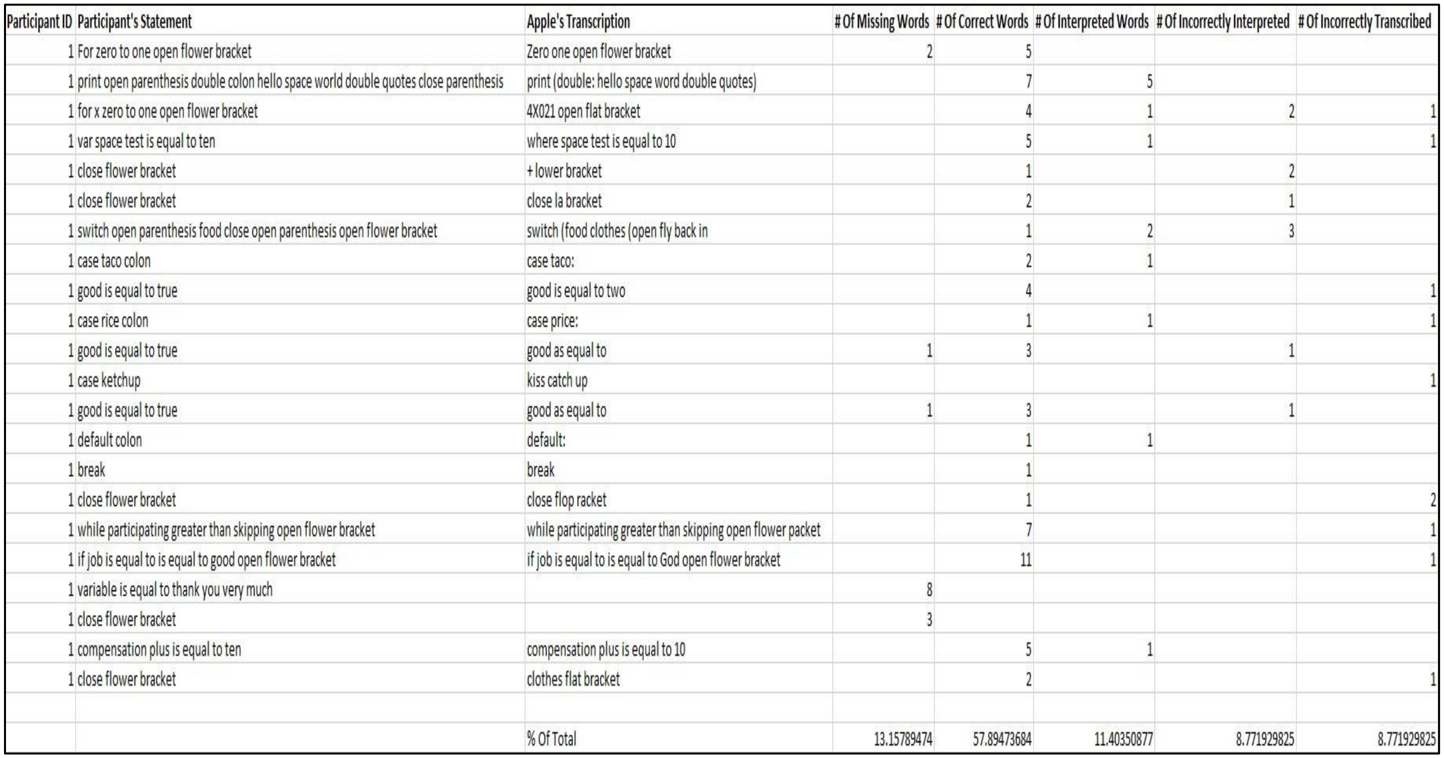
Participant ID	**Manual Transcription**
33	0214X0211 is equal to 10 switch food case taco good equal two cases rise good equal true good equal fall break while participating greater than equal good response equals thank you so much compensation plus equal time
34	4021 (print hello world for X equal to zeroone (one test equal to 10)) switch [food] (case taco good equal true case price: good equal to two case ketchup good equal to falls default: break) while participating get a greater than escaping) (its job equal to equal to good (where response equal to thank you very much) compensation plus plus or equal to 10)
35	4021 print hello world and 4X4021+10 to test variable exit the Loop add a switch statement and if the case is taco then good crew is assigned to Goode and if case is rice then true is a saint good and if cases ketchup then falls is assigned a girl add a default under break skip ad a while loop which runs as long as the participating is greater than skipping and inside the wall then add a response thank you very much add a compensation variable to10

36	Ride 4011 switch of food case one taco good equals to two cases to rise good equals true case we catch up good equals falls default break while participating if job equals good variable response equals thank you very much and compensation equals compensation +10
37	Follow 021 print hello world then follow OXO toone variable test equal to 10 close the loop then close the switch with a case of food and then case one taco in case to case rice would equal to two case catch-up would equal two falls close the switch while participating better than skipping look if condition job double sequel to do good variable response equal to thank you very much close the F Loop compensation equal to incremental compensation by 10 close the window
38	4_021 print hello world for X021 Welltest equal to 10 clothes for the clothes out of follow which phone case taco good equal to throw this rice ketchup good equal to force the fall break clothes switch though
Participant ID	**Manual Transcription**
	while participating in greater than skipping if jobs equal to equal to go to utilize in response to thank you very much clothes is no compensation equal to compensation +10

APPENDIX G:

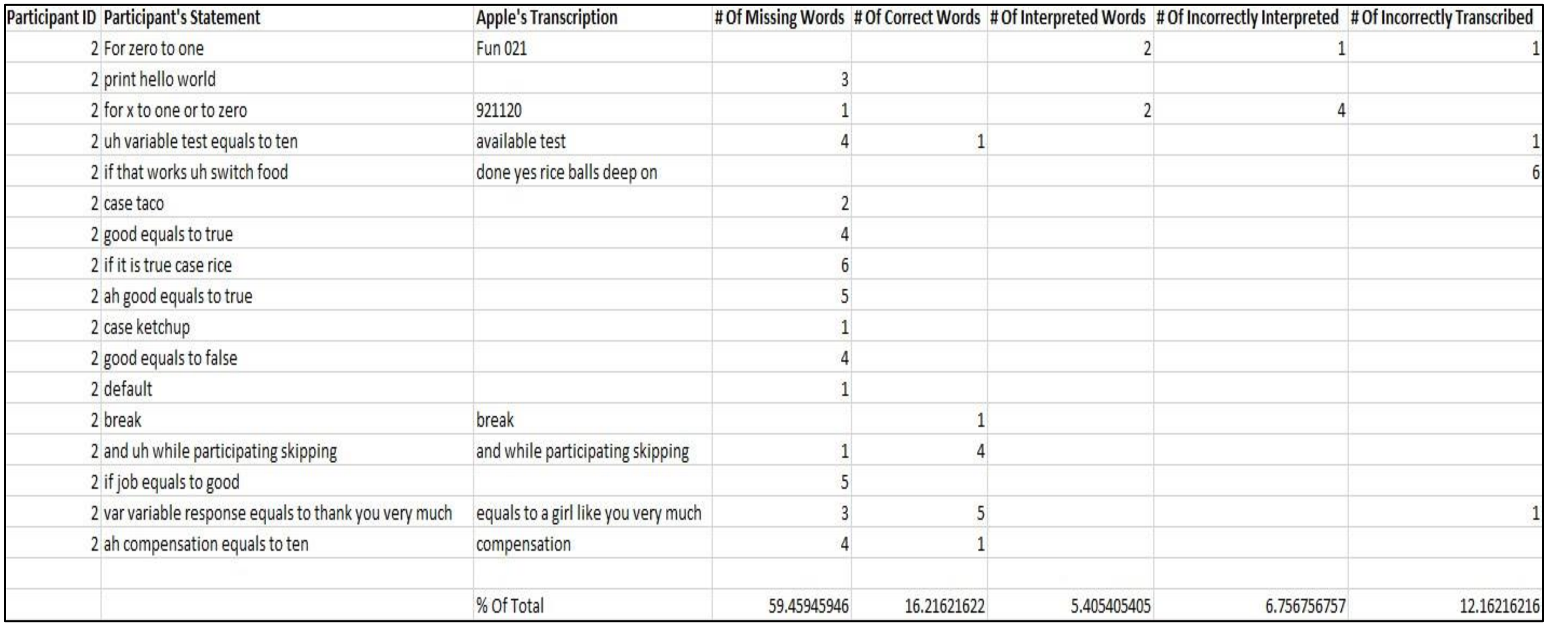
COMPARISON OF SPEECH TO APPLE’S TRANSCRIPTION

G.1) Participant 1



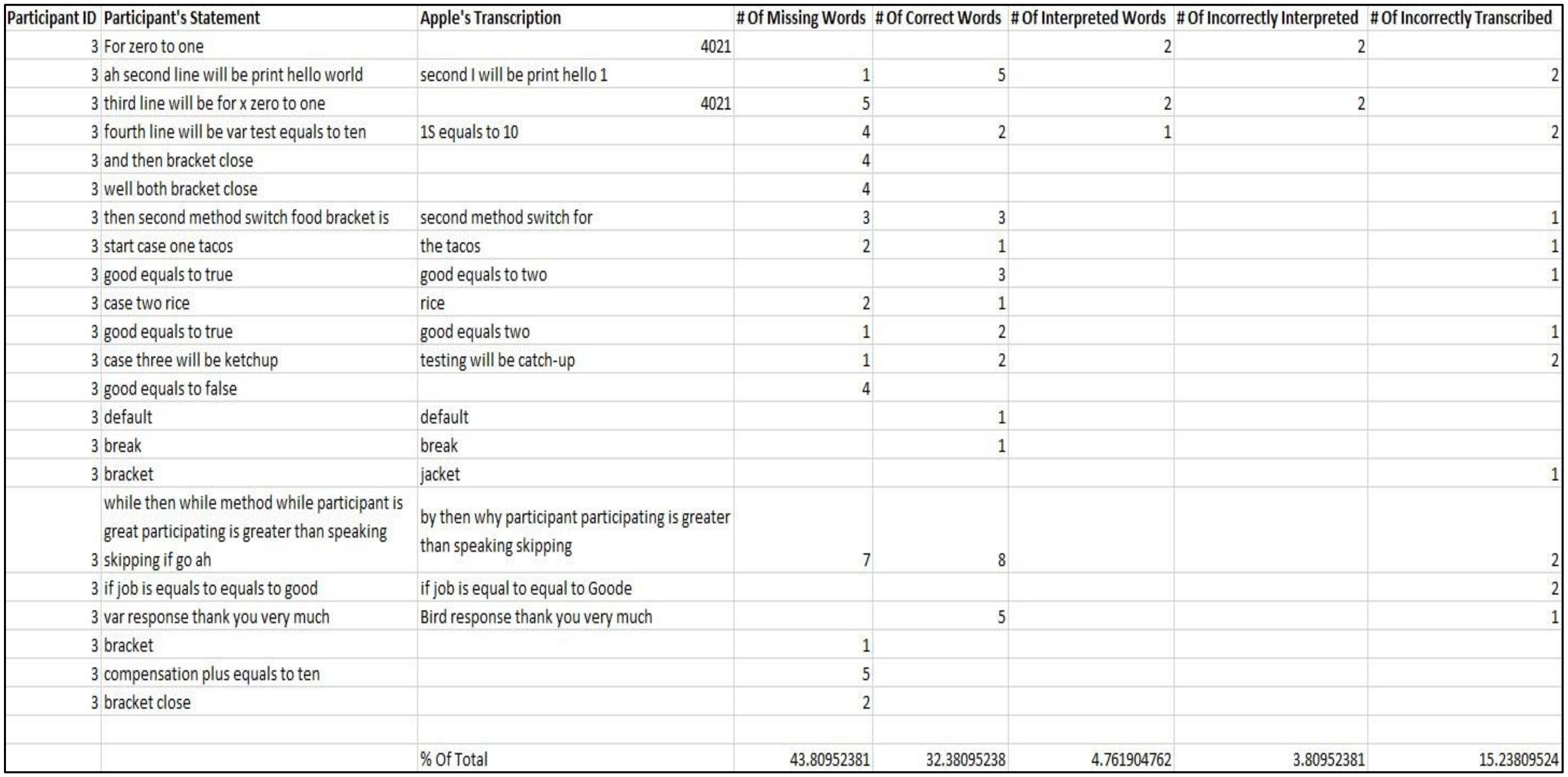
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
1	For zero to one open flower bracket	Zero one open flower bracket	2	5			
1	print open parenthesis double colon hello space world double quotes close parenthesis	print (double: hello space word double quotes)		7	5		
1	for x zero to one open flower bracket	4X021 open flat bracket		4	1	2	1
1	var space test is equal to ten	where space test is equal to 10		5	1		1
1	close flower bracket	+ lower bracket		1		2	
1	close flower bracket	close la bracket		2		1	
1	switch open parenthesis food close open parenthesis open flower bracket	switch (food clothes (open fly back in		1	2	3	
1	case taco colon	case taco:		2	1		
1	good is equal to true	good is equal to two		4			1
1	case rice colon	case price:		1	1		1
1	good is equal to true	good as equal to	1	3		1	
1	case ketchup	kiss catch up					1
1	good is equal to true	good as equal to	1	3		1	
1	default colon	default:		1	1		
1	break	break		1			
1	close flower bracket	close flop racket		1			2
1	while participating greater than skipping open flower bracket	while participating greater than skipping open flower packet		7			1
1	if job is equal to is equal to good open flower bracket	if job is equal to is equal to God open flower bracket		11			1
1	variable is equal to thank you very much		8				
1	close flower bracket		3				
1	compensation plus is equal to ten	compensation plus is equal to 10		5	1		
1	close flower bracket	clothes flat bracket		2			1
		% Of Total	13.15789474	57.89473684	11.40350877	8.771929825	8.771929825

G.2) Participant 2



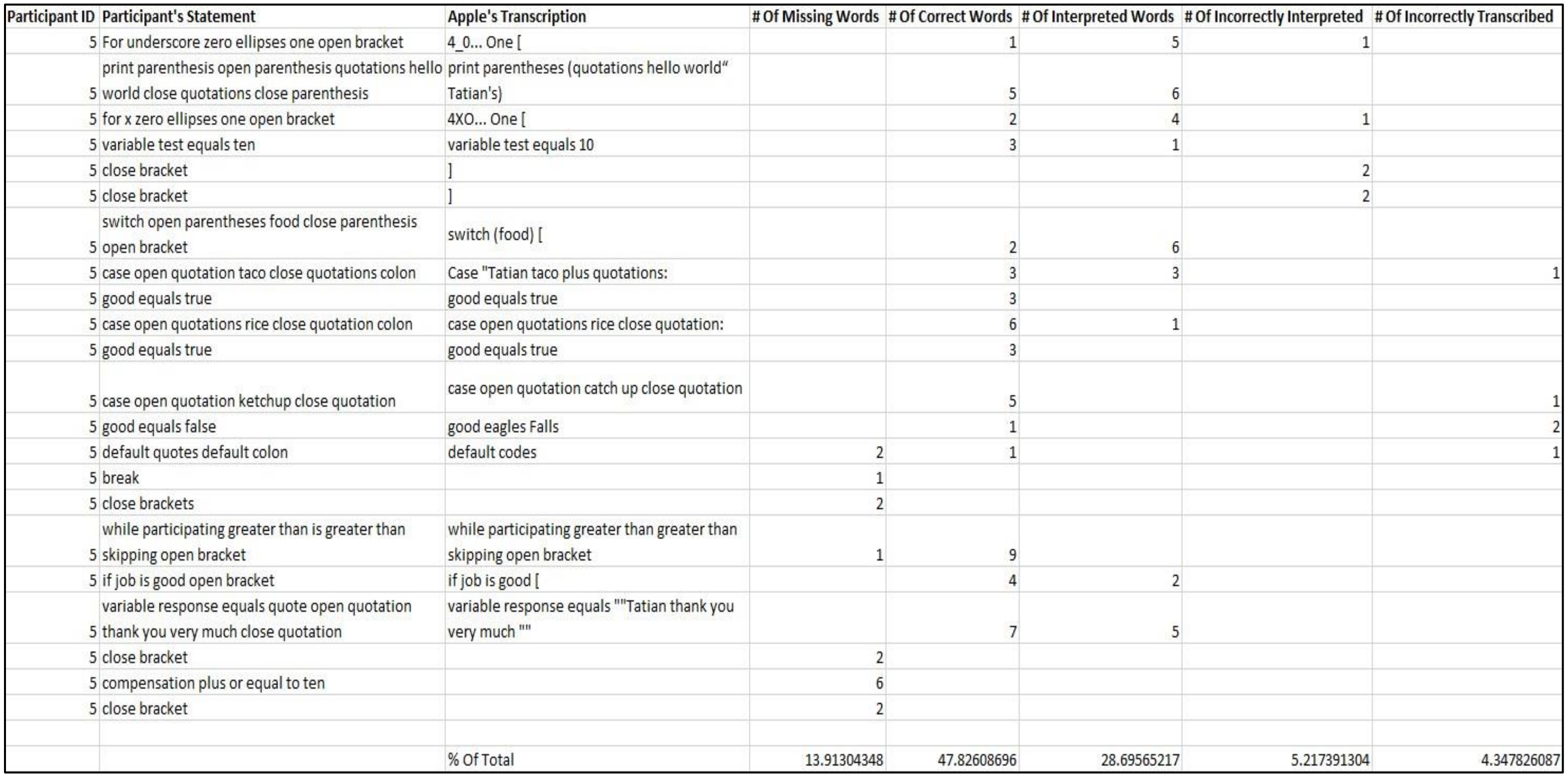
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
2	For zero to one	Fun 021			2	1	1
2	print hello world		3				
2	for x to one or to zero	921120	1		2	4	
2	uh variable test equals to ten	available test	4	1			1
2	if that works uh switch food	done yes rice balls deep on					6
2	case taco		2				
2	good equals to true		4				
2	if it is true case rice		6				
2	ah good equals to true		5				
2	case ketchup		1				
2	good equals to false		4				
2	default		1				
2	break	break		1			
2	and uh while participating skipping	and while participating skipping	1	4			
2	if job equals to good		5				
2	var variable response equals to thank you very much	equals to a girl like you very much	3	5			1
2	ah compensation equals to ten	compensation	4	1			
		% Of Total	59.45945946	16.21621622	5.405405405	6.756756757	12.16216216

G.3) Participant 3



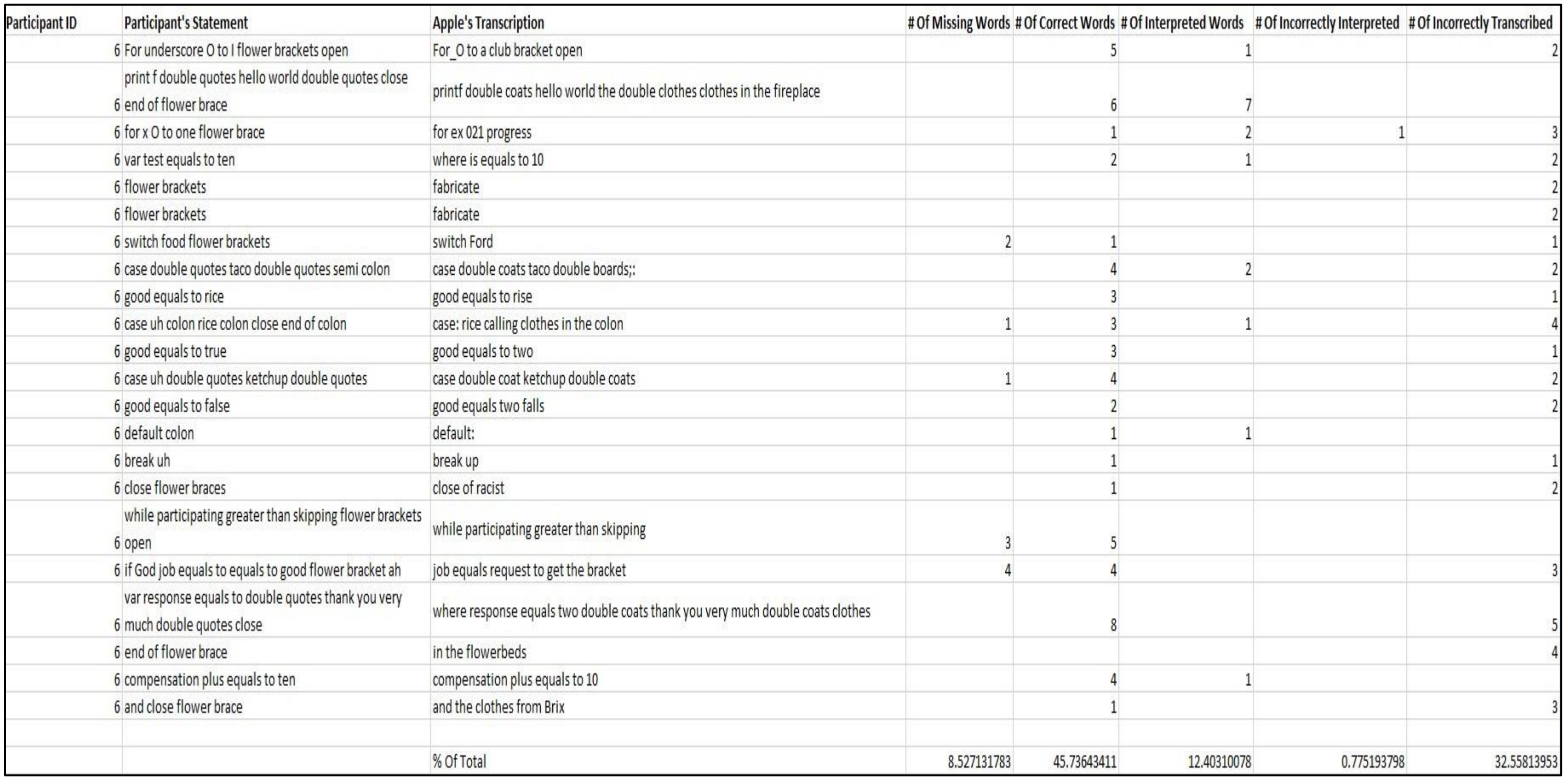
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
3	For zero to one	4021			2	2	
3	ah second line will be print hello world	second I will be print hello 1	1	5			2
3	third line will be for x zero to one	4021	5		2	2	
3	fourth line will be var test equals to ten	1S equals to 10	4	2	1		2
3	and then bracket close		4				
3	well both bracket close		4				
3	then second method switch food bracket is	second method switch for	3	3			1
3	start case one tacos	the tacos	2	1			1
3	good equals to true	good equals to two		3			1
3	case two rice	rice	2	1			
3	good equals to true	good equals two	1	2			1
3	case three will be ketchup	testing will be catch-up	1	2			2
3	good equals to false		4				
3	default	default		1			
3	break	break		1			
3	bracket	jacket					1
3	while then while method while participant is great participating is greater than speaking skipping if go ah	by then why participant participating is greater than speaking skipping	7	8			2
3	if job is equals to equals to good	if job is equal to equal to Goode					2
3	var response thank you very much	Bird response thank you very much		5			1
3	bracket		1				
3	compensation plus equals to ten		5				
3	bracket close		2				
		% Of Total	43.80952381	32.38095238	4.761904762	3.80952381	15.23809524

G.4) Participant 5



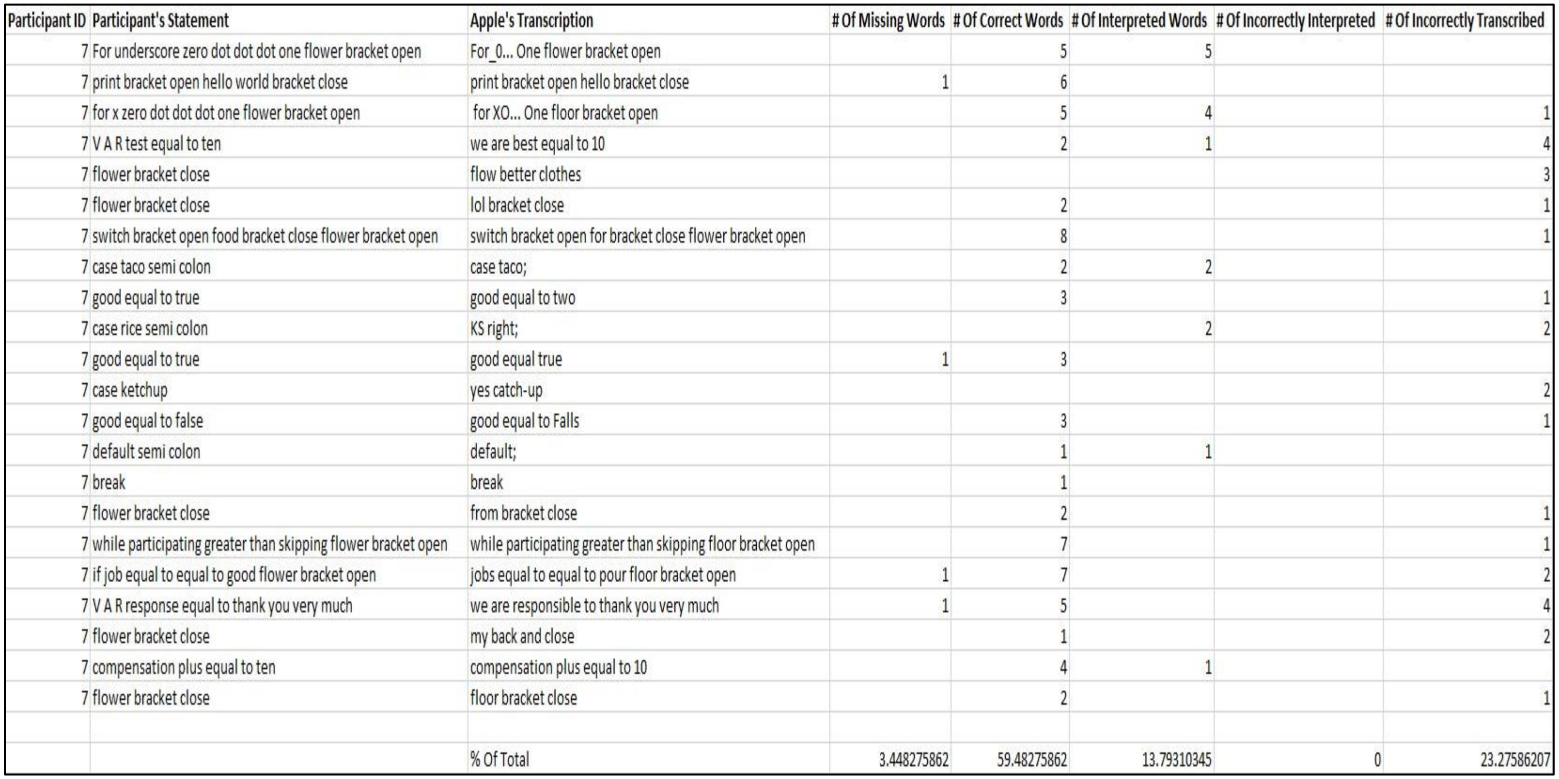
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
5	For underscore zero ellipses one open bracket	4_0... One [		1	5	1	
5	print parenthesis open parenthesis quotations hello world close quotations close parenthesis	print parentheses (quotations hello world" Tatian's)		5	6		
5	for x zero ellipses one open bracket	4XO... One [		2	4	1	
5	variable test equals ten	variable test equals 10		3	1		
5	close bracket	]				2	
5	close bracket	]				2	
5	switch open parentheses food close parenthesis open bracket	switch (food) [		2	6		
5	case open quotation taco close quotations colon	Case "Tatian taco plus quotations:		3	3		1
5	good equals true	good equals true		3			
5	case open quotations rice close quotation colon	case open quotations rice close quotation:		6	1		
5	good equals true	good equals true		3			
5	case open quotation ketchup close quotation	case open quotation catch up close quotation		5			1
5	good equals false	good eagles Falls		1			2
5	default quotes default colon	default codes	2	1			1
5	break		1				
5	close brackets		2				
5	while participating greater than is greater than skipping open bracket	while participating greater than greater than skipping open bracket	1	9			
5	if job is good open bracket	if job is good [		4	2		
5	variable response equals quote open quotation thank you very much close quotation	variable response equals ""Tatian thank you very much ""		7	5		
5	close bracket		2				
5	compensation plus or equal to ten		6				
5	close bracket		2				
		% Of Total	13.91304348	47.82608696	28.69565217	5.217391304	4.347826087

G.5) Participant 6



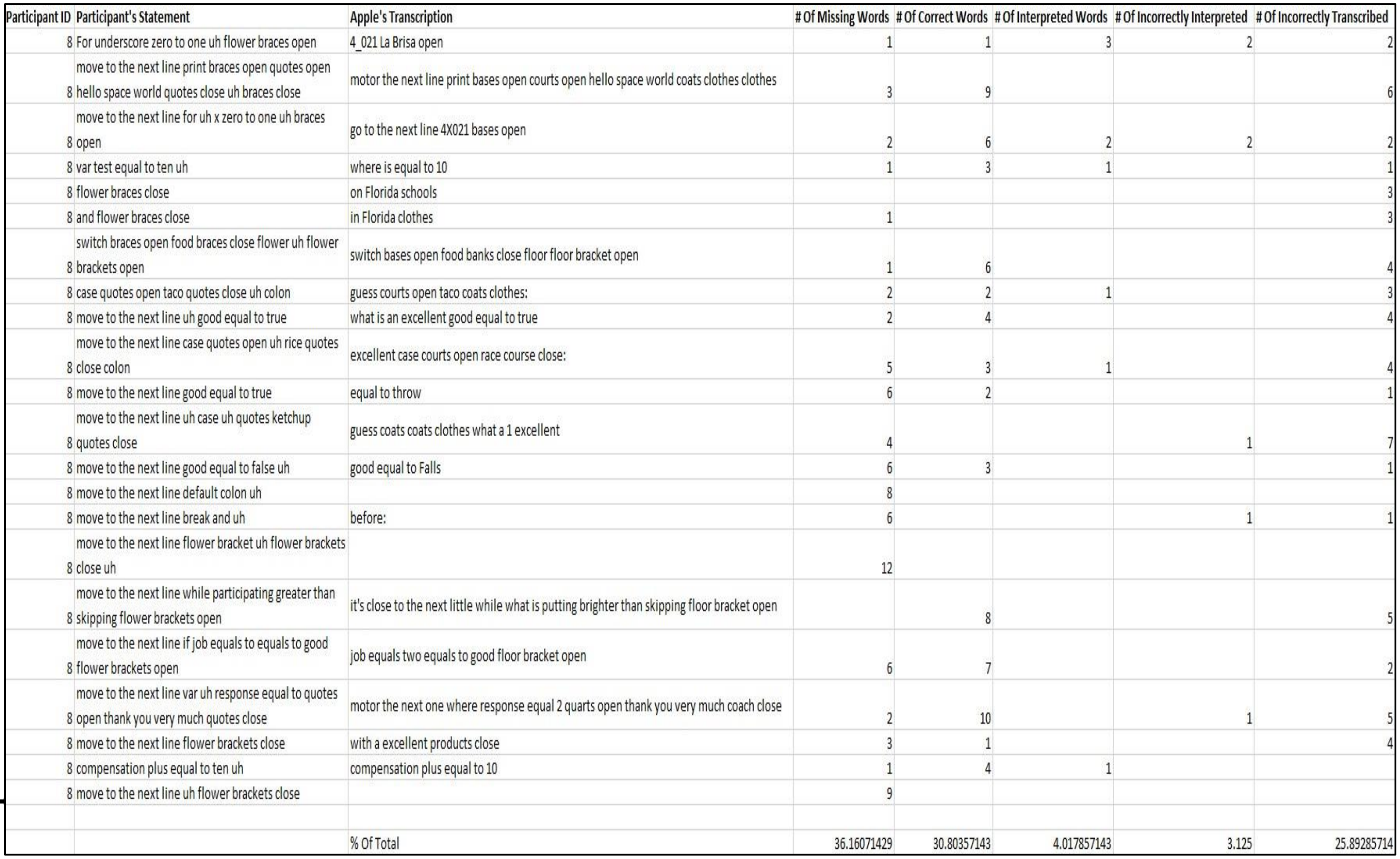
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
6	For underscore O to I flower brackets open	For_O to a club bracket open		5	1		2
6	print f double quotes hello world double quotes close end of flower brace	printf double coats hello world the double clothes clothes in the fireplace		6	7		
6	for x O to one flower brace	for ex 021 progress		1	2	1	3
6	var test equals to ten	where is equals to 10		2	1		2
6	flower brackets	fabricate					2
6	flower brackets	fabricate					2
6	switch food flower brackets	switch Ford	2	1			1
6	case double quotes taco double quotes semi colon	case double coats taco double boards;:		4	2		2
6	good equals to rice	good equals to rise		3			1
6	case uh colon rice colon close end of colon	case: rice calling clothes in the colon	1	3	1		4
6	good equals to true	good equals to two		3			1
6	case uh double quotes ketchup double quotes	case double coat ketchup double coats	1	4			2
6	good equals to false	good equals two falls		2			2
6	default colon	default:		1	1		
6	break uh	break up		1			1
6	close flower braces	close of racist		1			2
6	while participating greater than skipping flower brackets open	while participating greater than skipping	3	5			
6	if God job equals to equals to good flower bracket ah	job equals request to get the bracket	4	4			3
6	var response equals to double quotes thank you very much double quotes close	where response equals two double coats thank you very much double coats clothes		8			5
6	end of flower brace	in the flowerbeds					4
6	compensation plus equals to ten	compensation plus equals to 10		4	1		
6	and close flower brace	and the clothes from Brix		1			3
		% Of Total	8.527131783	45.73643411	12.40310078	0.775193798	32.55813953

G.6) Participant 7



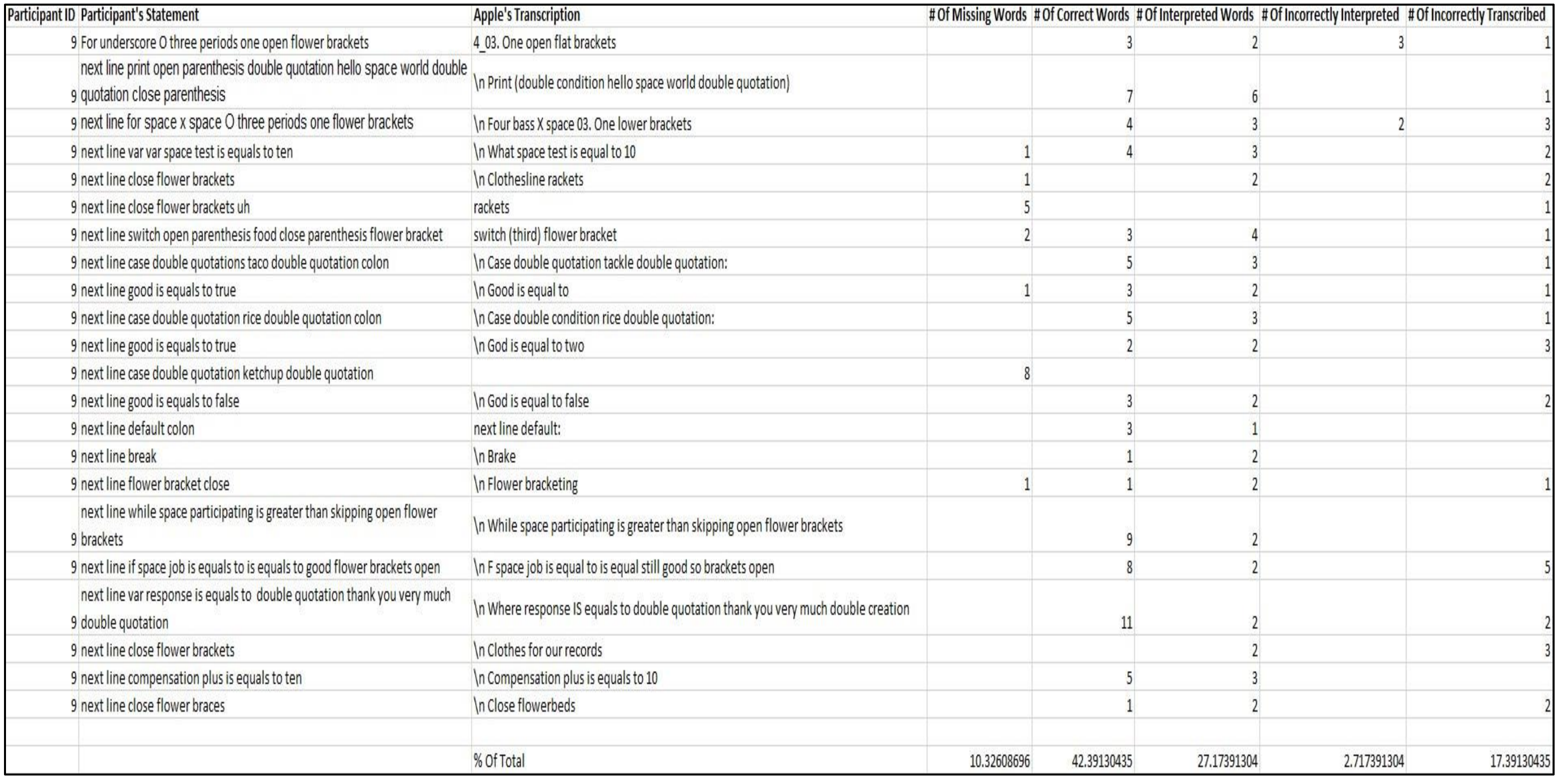
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
7	For underscore zero dot dot dot one flower bracket open	For_0... One flower bracket open		5	5		
7	print bracket open hello world bracket close	print bracket open hello bracket close	1	6			
7	for x zero dot dot dot one flower bracket open	for XO... One floor bracket open		5	4		1
7	V A R test equal to ten	we are best equal to 10		2	1		4
7	flower bracket close	flow better clothes					3
7	flower bracket close	lol bracket close		2			1
7	switch bracket open food bracket close flower bracket open	switch bracket open for bracket close flower bracket open		8			1
7	case taco semi colon	case taco;		2	2		
7	good equal to true	good equal to two		3			1
7	case rice semi colon	KS right;			2		2
7	good equal to true	good equal true	1	3			
7	case ketchup	yes catch-up					2
7	good equal to false	good equal to Falls		3			1
7	default semi colon	default;		1	1		
7	break	break		1			
7	flower bracket close	from bracket close		2			1
7	while participating greater than skipping flower bracket open	while participating greater than skipping floor bracket open		7			1
7	if job equal to equal to good flower bracket open	jobs equal to equal to pour floor bracket open	1	7			2
7	V A R response equal to thank you very much	we are responsible to thank you very much	1	5			4
7	flower bracket close	my back and close		1			2
7	compensation plus equal to ten	compensation plus equal to 10		4	1		
7	flower bracket close	floor bracket close		2			1
		% Of Total	3.448275862	59.48275862	13.79310345	0	23.27586207

G.7) Participant 8



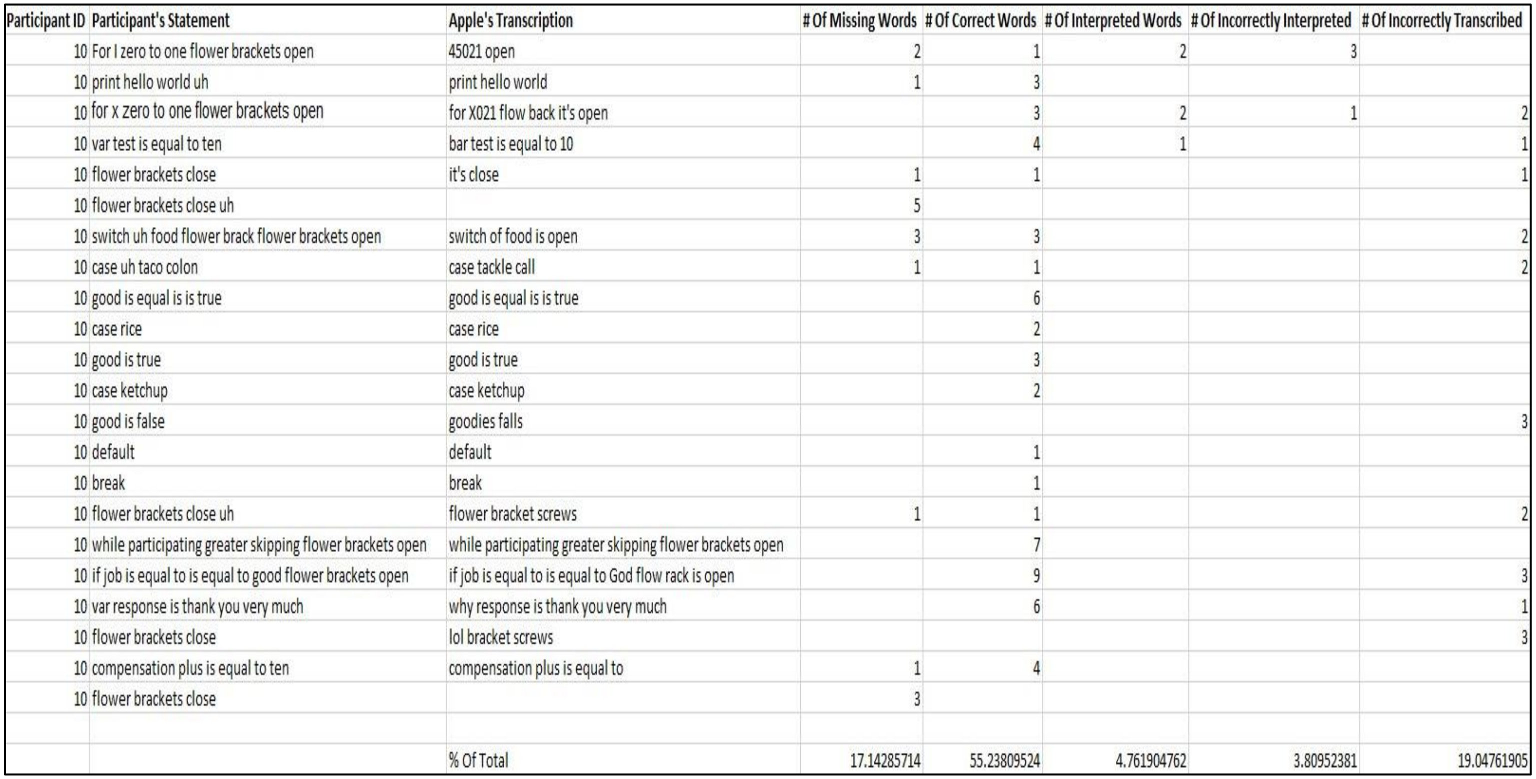
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
8	For underscore zero to one uh flower braces open	4_021 La Brisa open	1	1	3	2	2
8	move to the next line print braces open quotes open hello space world quotes close uh braces close	motor the next line print bases open courts open hello space world coats clothes clothes	3	9			6
8	move to the next line for uh x zero to one uh braces open	go to the next line 4X021 bases open	2	6	2	2	2
8	var test equal to ten uh	where is equal to 10	1	3	1		1
8	flower braces close	on Florida schools					3
8	and flower braces close	in Florida clothes	1				3
8	switch braces open food braces close flower uh flower brackets open	switch bases open food banks close floor floor bracket open	1	6			4
8	case quotes open taco quotes close uh colon	guess courts open taco coats clothes:	2	2	1		3
8	move to the next line uh good equal to true	what is an excellent good equal to true	2	4			4
8	move to the next line case quotes open uh rice quotes close colon	excellent case courts open race course close:	5	3	1		4
8	move to the next line good equal to true	equal to throw	6	2			1
8	move to the next line uh case uh quotes ketchup quotes close	guess coats coats clothes what a 1 excellent	4			1	7
8	move to the next line good equal to false uh	good equal to Falls	6	3			1
8	move to the next line default colon uh		8				
8	move to the next line break and uh	before:	6			1	1
8	move to the next line flower bracket uh flower brackets close uh		12				
8	move to the next line while participating greater than skipping flower brackets open	it's close to the next little while what is putting brighter than skipping floor bracket open		8			5
8	move to the next line if job equals to equals to good flower brackets open	job equals two equals to good floor bracket open	6	7			2
8	move to the next line var uh response equal to quotes open thank you very much quotes close	motor the next one where response equal 2 quarts open thank you very much coach close	2	10		1	5
8	move to the next line flower brackets close	with a excellent products close	3	1			4
8	compensation plus equal to ten uh	compensation plus equal to 10	1	4	1		
8	move to the next line uh flower brackets close		9				
		% Of Total	36.16071429	30.80357143	4.017857143	3.125	25.89285714

G.8) Participant 9



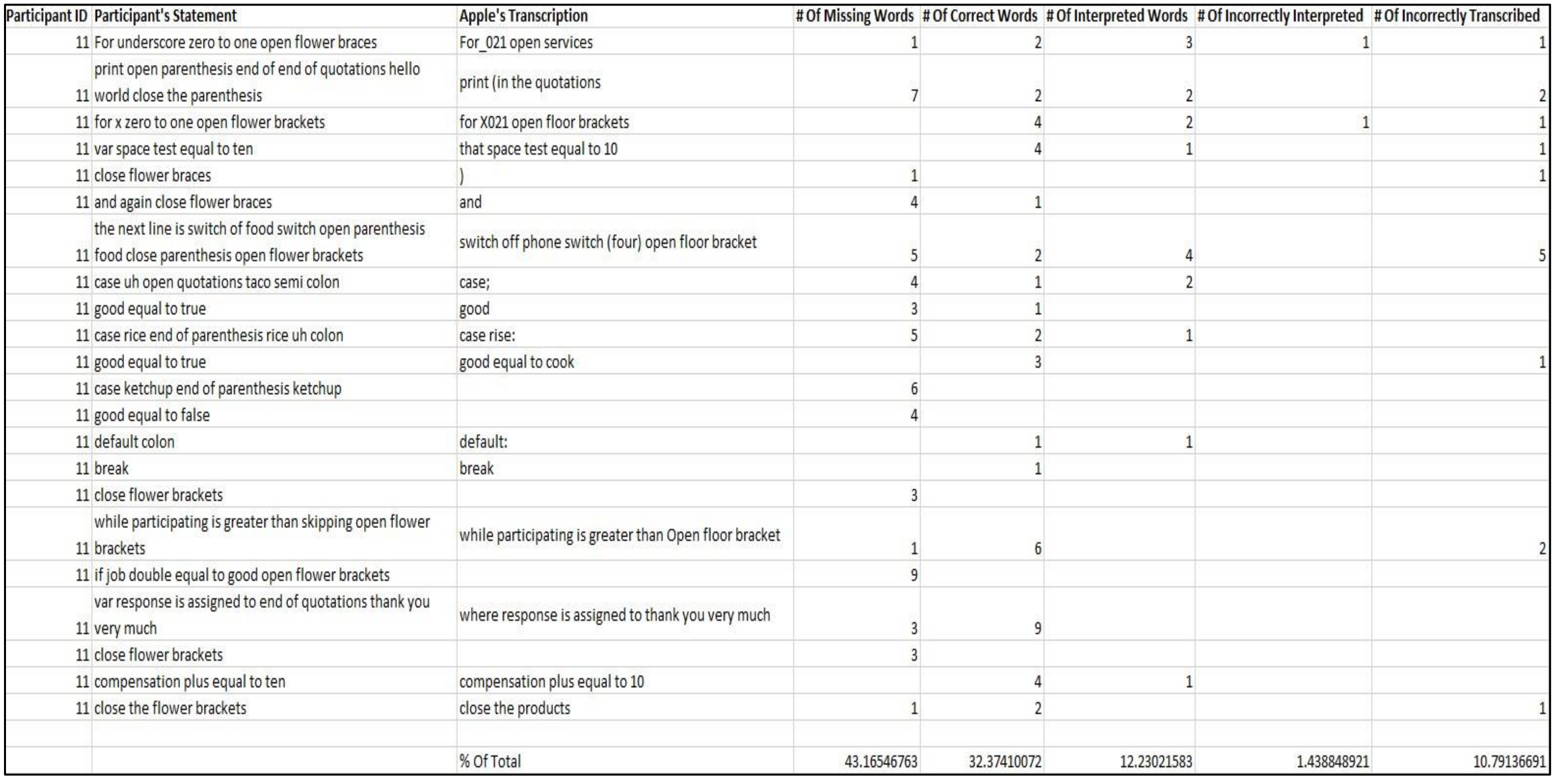
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
9	For underscore O three periods one open flower brackets	4_03. One open flat brackets		3	2	3	1
9	next line print open parenthesis double quotation hello space world double quotation close parenthesis	\n Print (double condition hello space world double quotation)		7	6		1
9	next line for space x space O three periods one flower brackets	\n Four bass X space 03. One lower brackets		4	3	2	3
9	next line var var space test is equals to ten	\n What space test is equal to 10	1	4	3		2
9	next line close flower brackets	\n Clothesline rackets	1		2		2
9	next line close flower brackets uh	rackets	5				1
9	next line switch open parenthesis food close parenthesis flower bracket	switch (third) flower bracket	2	3	4		1
9	next line case double quotations taco double quotation colon	\n Case double quotation tackle double quotation:		5	3		1
9	next line good is equals to true	\n Good is equal to	1	3	2		1
9	next line case double quotation rice double quotation colon	\n Case double condition rice double quotation:		5	3		1
9	next line good is equals to true	\n God is equal to two		2	2		3
9	next line case double quotation ketchup double quotation		8				
9	next line good is equals to false	\n God is equal to false		3	2		2
9	next line default colon	next line default:		3	1		
9	next line break	\n Brake		1	2		
9	next line flower bracket close	\n Flower bracketing	1	1	2		1
9	next line while space participating is greater than skipping open flower brackets	\n While space participating is greater than skipping open flower brackets		9	2		
9	next line if space job is equals to is equals to good flower brackets open	\n F space job is equal to is equal still good so brackets open		8	2		5
9	next line var response is equals to double quotation thank you very much double quotation	\n Where response IS equals to double quotation thank you very much double creation		11	2		2
9	next line close flower brackets	\n Clothes for our records			2		3
9	next line compensation plus is equals to ten	\n Compensation plus is equals to 10		5	3		
9	next line close flower braces	\n Close flowerbeds		1	2		2
		% Of Total	10.32608696	42.39130435	27.17391304	2.717391304	17.39130435

G.9) Participant 10



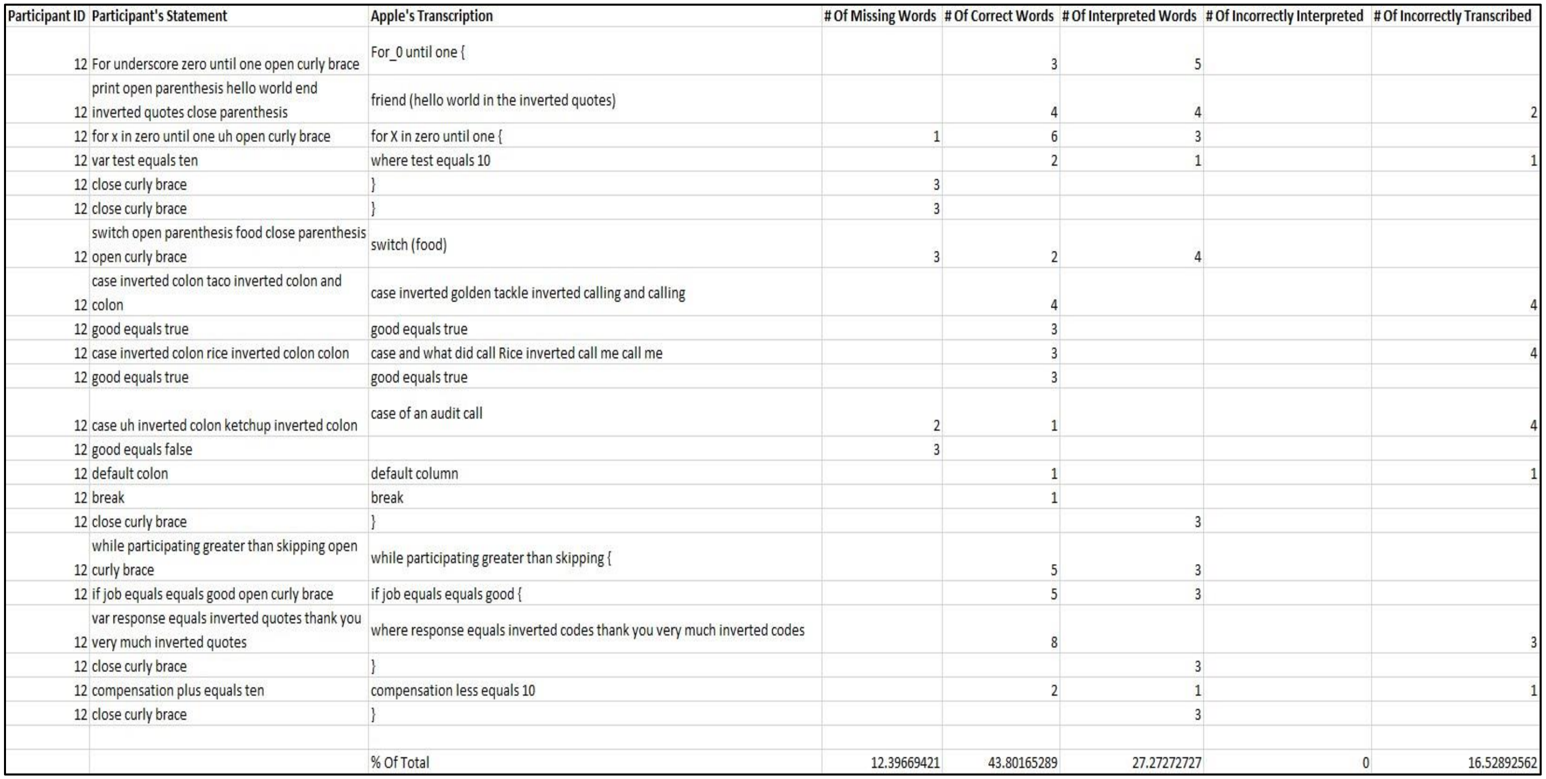
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
10	For I zero to one flower brackets open	45021 open	2	1	2	3	
10	print hello world uh	print hello world	1	3			
10	for x zero to one flower brackets open	for X021 flow back it's open		3	2	1	2
10	var test is equal to ten	bar test is equal to 10		4	1		1
10	flower brackets close	it's close	1	1			1
10	flower brackets close uh		5				
10	switch uh food flower brack flower brackets open	switch of food is open	3	3			2
10	case uh taco colon	case tackle call	1	1			2
10	good is equal is is true	good is equal is is true		6			
10	case rice	case rice		2			
10	good is true	good is true		3			
10	case ketchup	case ketchup		2			
10	good is false	goodies falls					3
10	default	default		1			
10	break	break		1			
10	flower brackets close uh	flower bracket screws	1	1			2
10	while participating greater skipping flower brackets open	while participating greater skipping flower brackets open		7			
10	if job is equal to is equal to good flower brackets open	if job is equal to is equal to God flow rack is open		9			3
10	var response is thank you very much	why response is thank you very much		6			1
10	flower brackets close	lol bracket screws					3
10	compensation plus is equal to ten	compensation plus is equal to	1	4			
10	flower brackets close		3				
		% Of Total	17.14285714	55.23809524	4.761904762	3.80952381	19.04761905

G.10) Participant 11



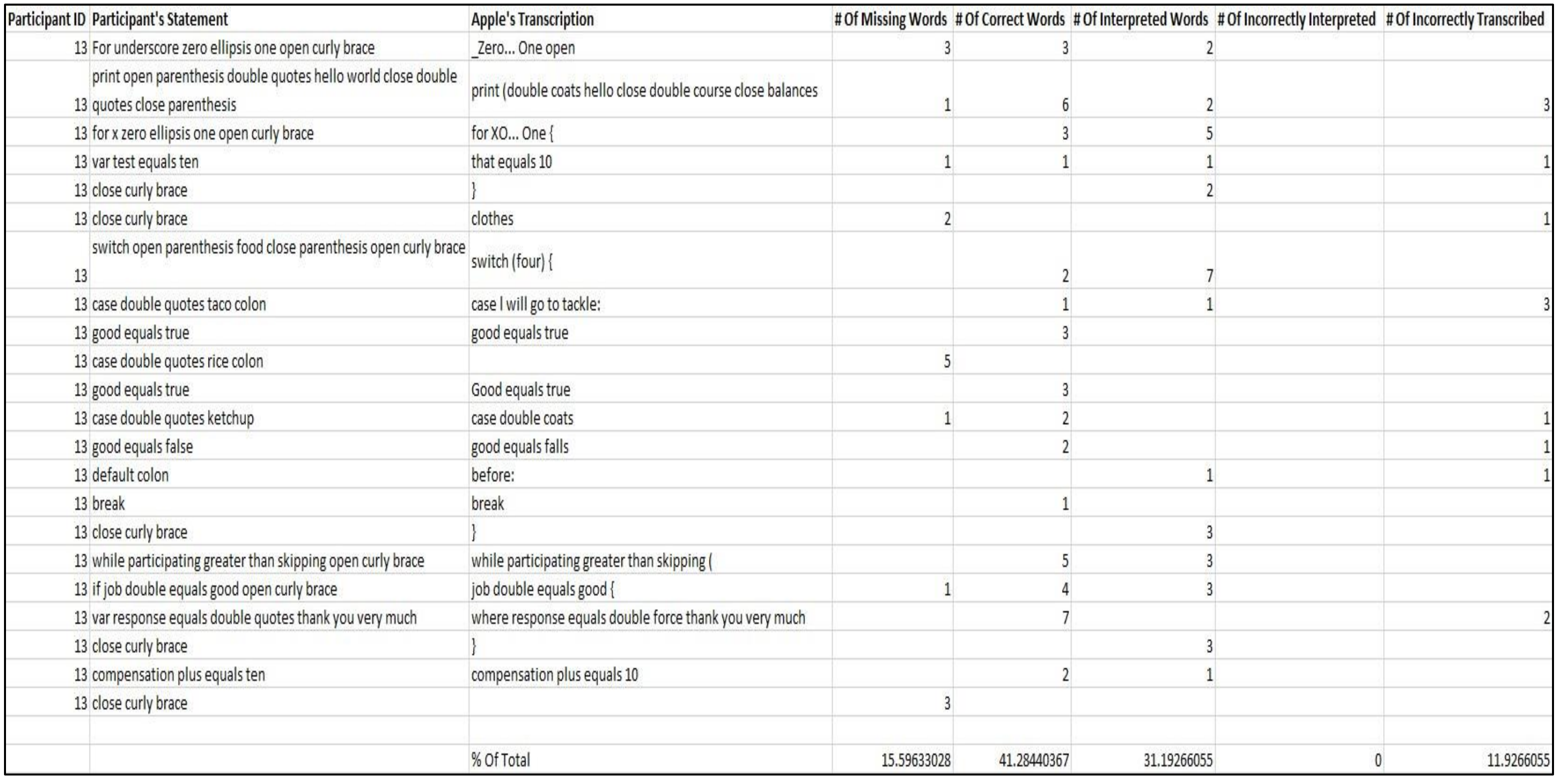
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
11	For underscore zero to one open flower braces	For_021 open services	1	2	3	1	1
11	print open parenthesis end of end of quotations hello world close the parenthesis	print (in the quotations	7	2	2		2
11	for x zero to one open flower brackets	for X021 open floor brackets		4	2	1	1
11	var space test equal to ten	that space test equal to 10		4	1		1
11	close flower braces	)	1				1
11	and again close flower braces	and	4	1			
11	the next line is switch of food switch open parenthesis food close parenthesis open flower brackets	switch off phone switch (four) open floor bracket	5	2	4		5
11	case uh open quotations taco semi colon	case;	4	1	2		
11	good equal to true	good	3	1			
11	case rice end of parenthesis rice uh colon	case rise:	5	2	1		
11	good equal to true	good equal to cook		3			1
11	case ketchup end of parenthesis ketchup		6				
11	good equal to false		4				
11	default colon	default:		1	1		
11	break	break		1			
11	close flower brackets		3				
11	while participating is greater than skipping open flower brackets	while participating is greater than Open floor bracket	1	6			2
11	if job double equal to good open flower brackets		9				
11	var response is assigned to end of quotations thank you very much	where response is assigned to thank you very much	3	9			
11	close flower brackets		3				
11	compensation plus equal to ten	compensation plus equal to 10		4	1		
11	close the flower brackets	close the products	1	2			1
		% Of Total	43.16546763	32.37410072	12.23021583	1.438848921	10.79136691

G.11) Participant 12



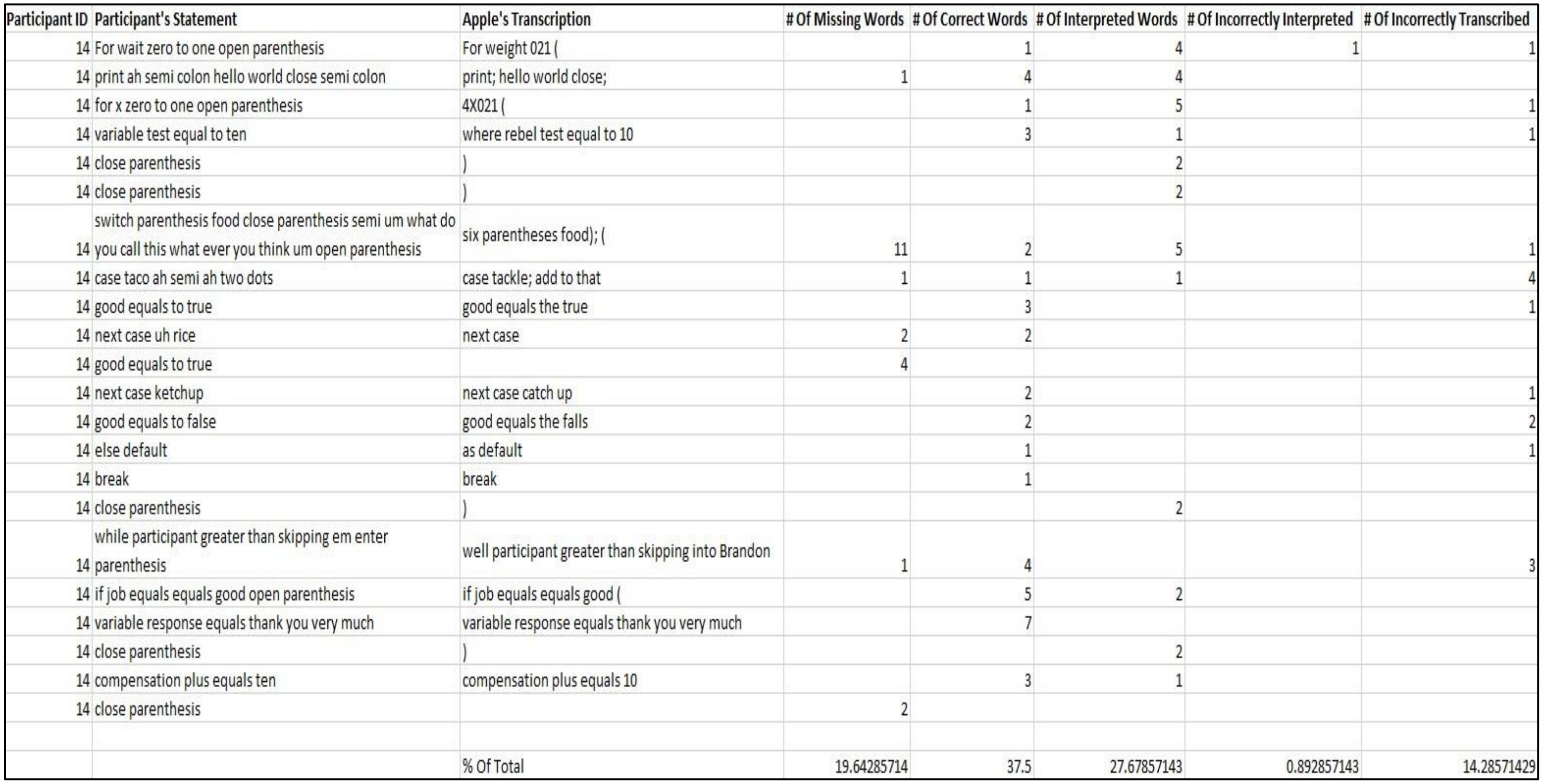
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
12	For underscore zero until one open curly brace	For_0 until one {		3	5		
12	print open parenthesis hello world end inverted quotes close parenthesis	friend (hello world in the inverted quotes)		4	4		2
12	for x in zero until one uh open curly brace	for X in zero until one {	1	6	3		
12	var test equals ten	where test equals 10		2	1		1
12	close curly brace	}	3				
12	close curly brace	}	3				
12	switch open parenthesis food close parenthesis open curly brace	switch (food)	3	2	4		
12	case inverted colon taco inverted colon and colon	case inverted golden tackle inverted calling and calling		4			4
12	good equals true	good equals true		3			
12	case inverted colon rice inverted colon colon	case and what did call Rice inverted call me call me		3			4
12	good equals true	good equals true		3			
12	case uh inverted colon ketchup inverted colon	case of an audit call	2	1			4
12	good equals false		3				
12	default colon	default column		1			1
12	break	break		1			
12	close curly brace	}			3		
12	while participating greater than skipping open curly brace	while participating greater than skipping {		5	3		
12	if job equals equals good open curly brace	if job equals equals good {		5	3		
12	var response equals inverted quotes thank you very much inverted quotes	where response equals inverted codes thank you very much inverted codes		8			3
12	close curly brace	}			3		
12	compensation plus equals ten	compensation less equals 10		2	1		1
12	close curly brace	}			3		
		% Of Total	12.39669421	43.80165289	27.27272727	0	16.52892562

G.12) Participant 13



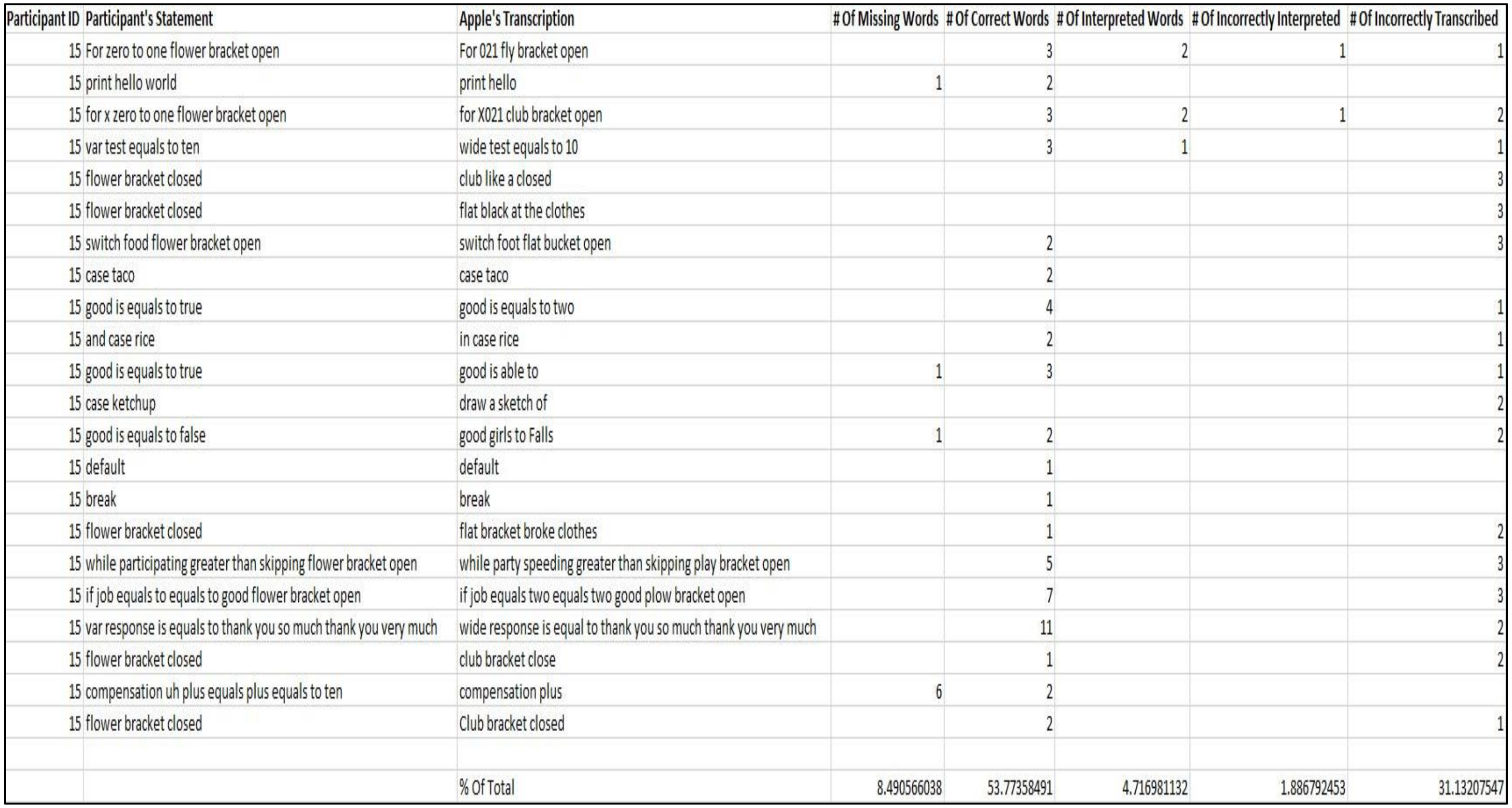
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
13	For underscore zero ellipsis one open curly brace	_Zero... One open	3	3	2		
13	print open parenthesis double quotes hello world close double quotes close parenthesis	print (double coats hello close double course close balances	1	6	2		3
13	for x zero ellipsis one open curly brace	for XO... One {		3	5		
13	var test equals ten	that equals 10	1	1	1		1
13	close curly brace	}			2		
13	close curly brace	clothes	2				1
13	switch open parenthesis food close parenthesis open curly brace	switch (four) {		2	7		
13	case double quotes taco colon	case I will go to tackle:		1	1		3
13	good equals true	good equals true		3			
13	case double quotes rice colon		5				
13	good equals true	Good equals true		3			
13	case double quotes ketchup	case double coats	1	2			1
13	good equals false	good equals falls		2			1
13	default colon	before:			1		1
13	break	break		1			
13	close curly brace	}			3		
13	while participating greater than skipping open curly brace	while participating greater than skipping (		5	3		
13	if job double equals good open curly brace	job double equals good {	1	4	3		
13	var response equals double quotes thank you very much	where response equals double force thank you very much		7			2
13	close curly brace	}			3		
13	compensation plus equals ten	compensation plus equals 10		2	1		
13	close curly brace		3				
		% Of Total	15.59633028	41.28440367	31.19266055	0	11.9266055

G.13) Participant 14



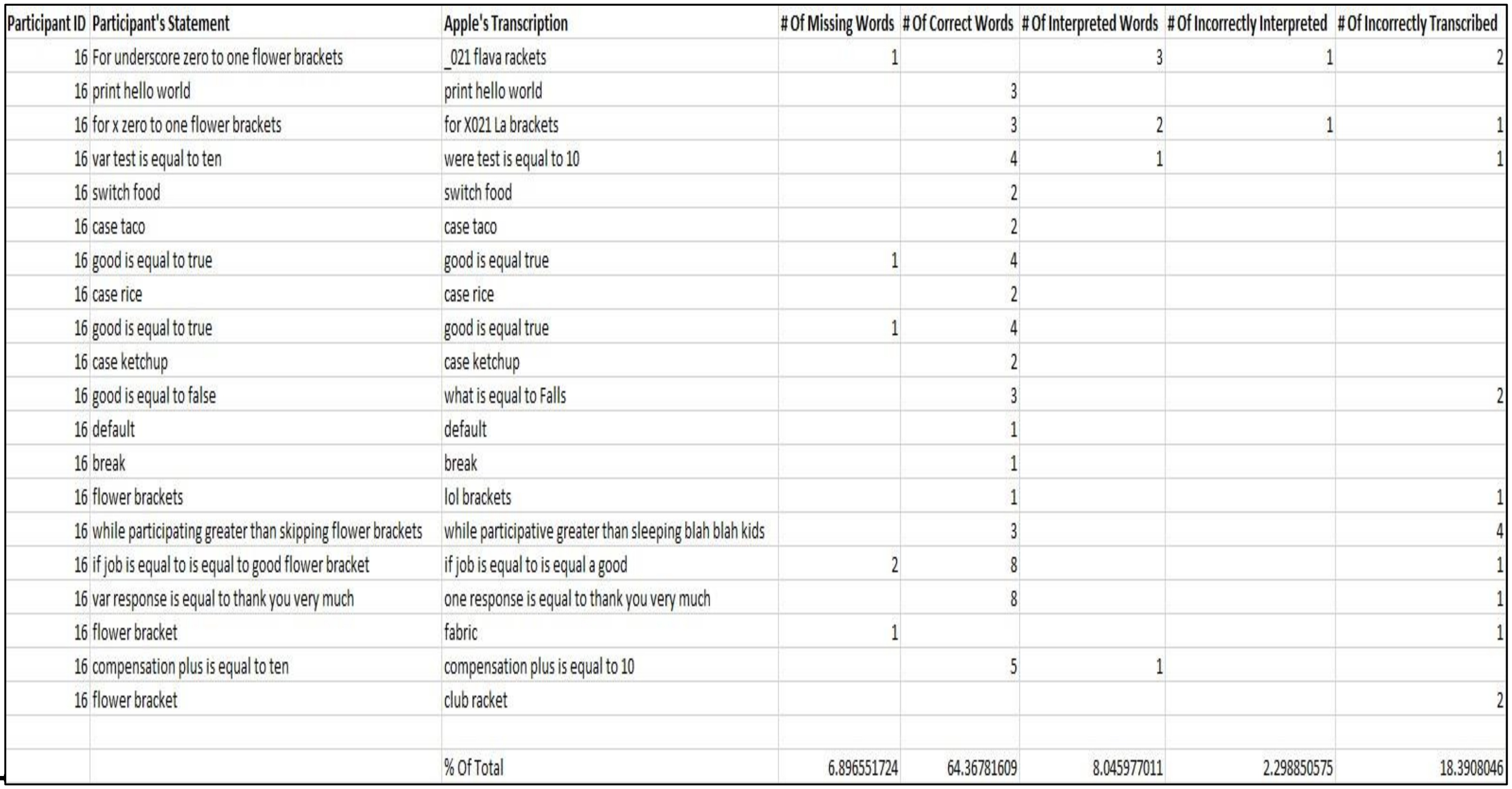
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
14	For wait zero to one open parenthesis	For weight 021 (		1	4	1	1
14	print ah semi colon hello world close semi colon	print; hello world close;	1	4	4		
14	for x zero to one open parenthesis	4X021 (		1	5		1
14	variable test equal to ten	where rebel test equal to 10		3	1		1
14	close parenthesis	)			2		
14	close parenthesis	)			2		
14	switch parenthesis food close parenthesis semi um what do you call this what ever you think um open parenthesis	six parentheses food); (	11	2	5		1
14	case taco ah semi ah two dots	case tackle; add to that	1	1	1		4
14	good equals to true	good equals the true		3			1
14	next case uh rice	next case	2	2			
14	good equals to true		4				
14	next case ketchup	next case catch up		2			1
14	good equals to false	good equals the falls		2			2
14	else default	as default		1			1
14	break	break		1			
14	close parenthesis	)			2		
14	while participant greater than skipping em enter parenthesis	well participant greater than skipping into Brandon	1	4			3
14	if job equals equals good open parenthesis	if job equals equals good (		5	2		
14	variable response equals thank you very much	variable response equals thank you very much		7			
14	close parenthesis	)			2		
14	compensation plus equals ten	compensation plus equals 10		3	1		
14	close parenthesis		2				
		% Of Total	19.64285714	37.5	27.67857143	0.892857143	14.28571429

G.14) Participant 15



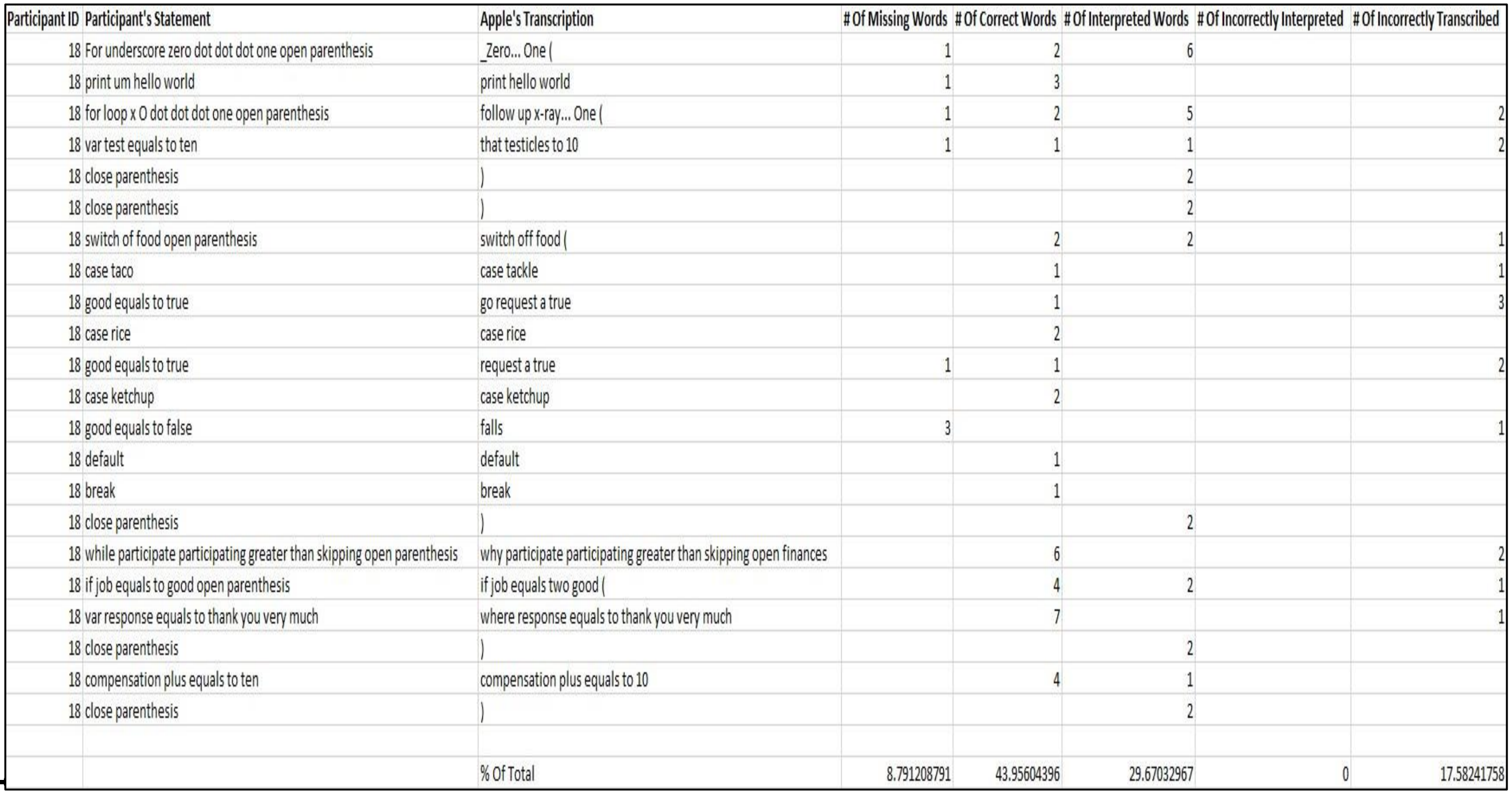
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
15	For zero to one flower bracket open	For 021 fly bracket open		3	2	1	1
15	print hello world	print hello	1	2			
15	for x zero to one flower bracket open	for X021 club bracket open		3	2	1	2
15	var test equals to ten	wide test equals to 10		3	1		1
15	flower bracket closed	club like a closed					3
15	flower bracket closed	flat black at the clothes					3
15	switch food flower bracket open	switch foot flat bucket open		2			3
15	case taco	case taco		2			
15	good is equals to true	good is equals to two		4			1
15	and case rice	in case rice		2			1
15	good is equals to true	good is able to	1	3			1
15	case ketchup	draw a sketch of					2
15	good is equals to false	good girls to Falls	1	2			2
15	default	default		1			
15	break	break		1			
15	flower bracket closed	flat bracket broke clothes		1			2
15	while participating greater than skipping flower bracket open	while party speeding greater than skipping play bracket open		5			3
15	if job equals to equals to good flower bracket open	if job equals two equals two good plow bracket open		7			3
15	var response is equals to thank you so much thank you very much	wide response is equal to thank you so much thank you very much		11			2
15	flower bracket closed	club bracket close		1			2
15	compensation uh plus equals plus equals to ten	compensation plus	6	2			
15	flower bracket closed	Club bracket closed		2			1
		% Of Total	8.490566038	53.77358491	4.716981132	1.886792453	31.13207547

G.15) Participant 16



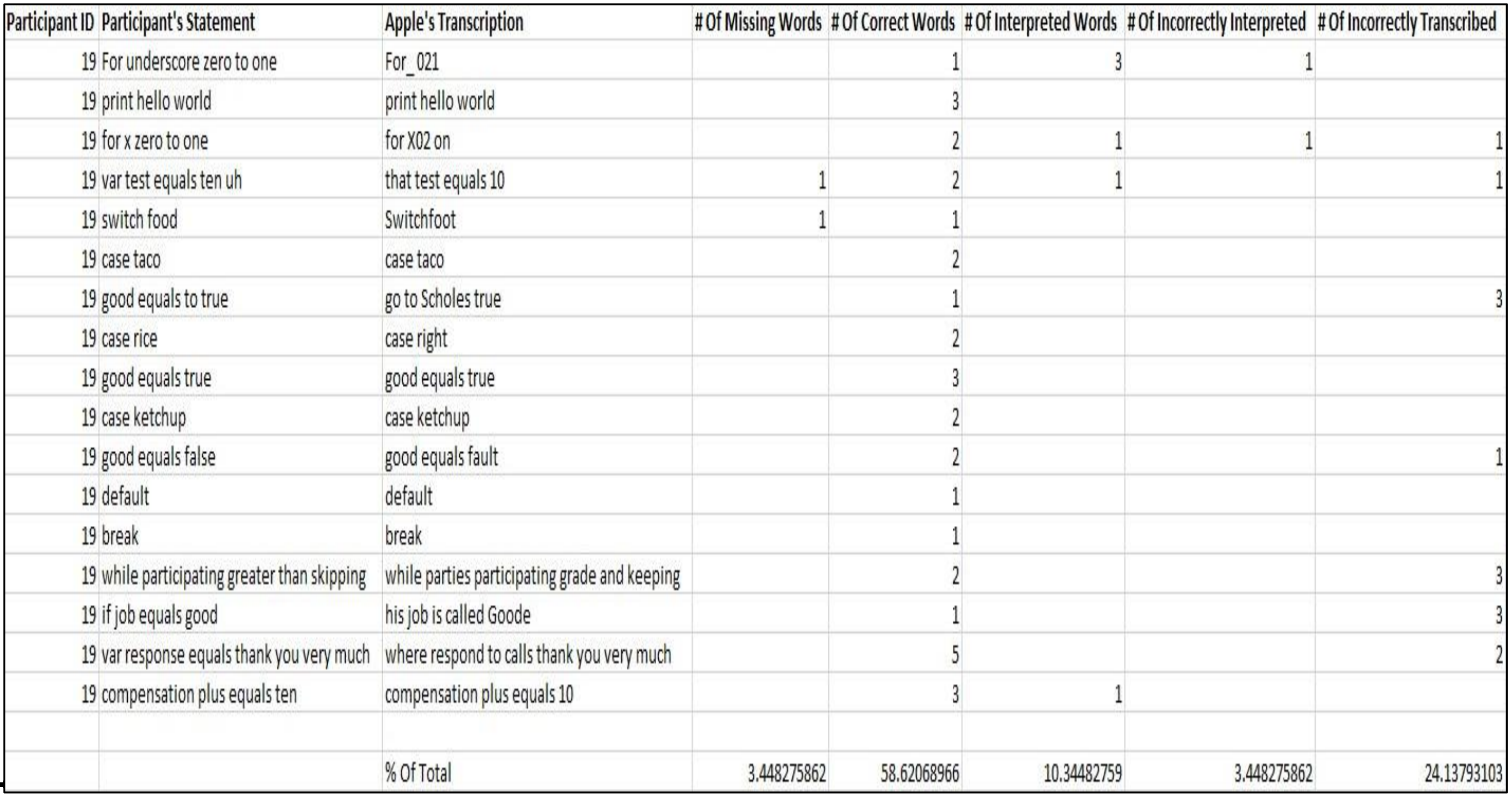
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
16	For underscore zero to one flower brackets	_021 flava rackets	1		3	1	2
16	print hello world	print hello world		3			
16	for x zero to one flower brackets	for X021 La brackets		3	2	1	1
16	var test is equal to ten	were test is equal to 10		4	1		1
16	switch food	switch food		2			
16	case taco	case taco		2			
16	good is equal to true	good is equal true	1	4			
16	case rice	case rice		2			
16	good is equal to true	good is equal true	1	4			
16	case ketchup	case ketchup		2			
16	good is equal to false	what is equal to Falls		3			2
16	default	default		1			
16	break	break		1			
16	flower brackets	lol brackets		1			1
16	while participating greater than skipping flower brackets	while participative greater than sleeping blah blah kids		3			4
16	if job is equal to is equal to good flower bracket	if job is equal to is equal a good	2	8			1
16	var response is equal to thank you very much	one response is equal to thank you very much		8			1
16	flower bracket	fabric	1				1
16	compensation plus is equal to ten	compensation plus is equal to 10		5	1		
16	flower bracket	club racket					2
		% Of Total	6.896551724	64.36781609	8.045977011	2.298850575	18.3908046

G.16) Participant 18



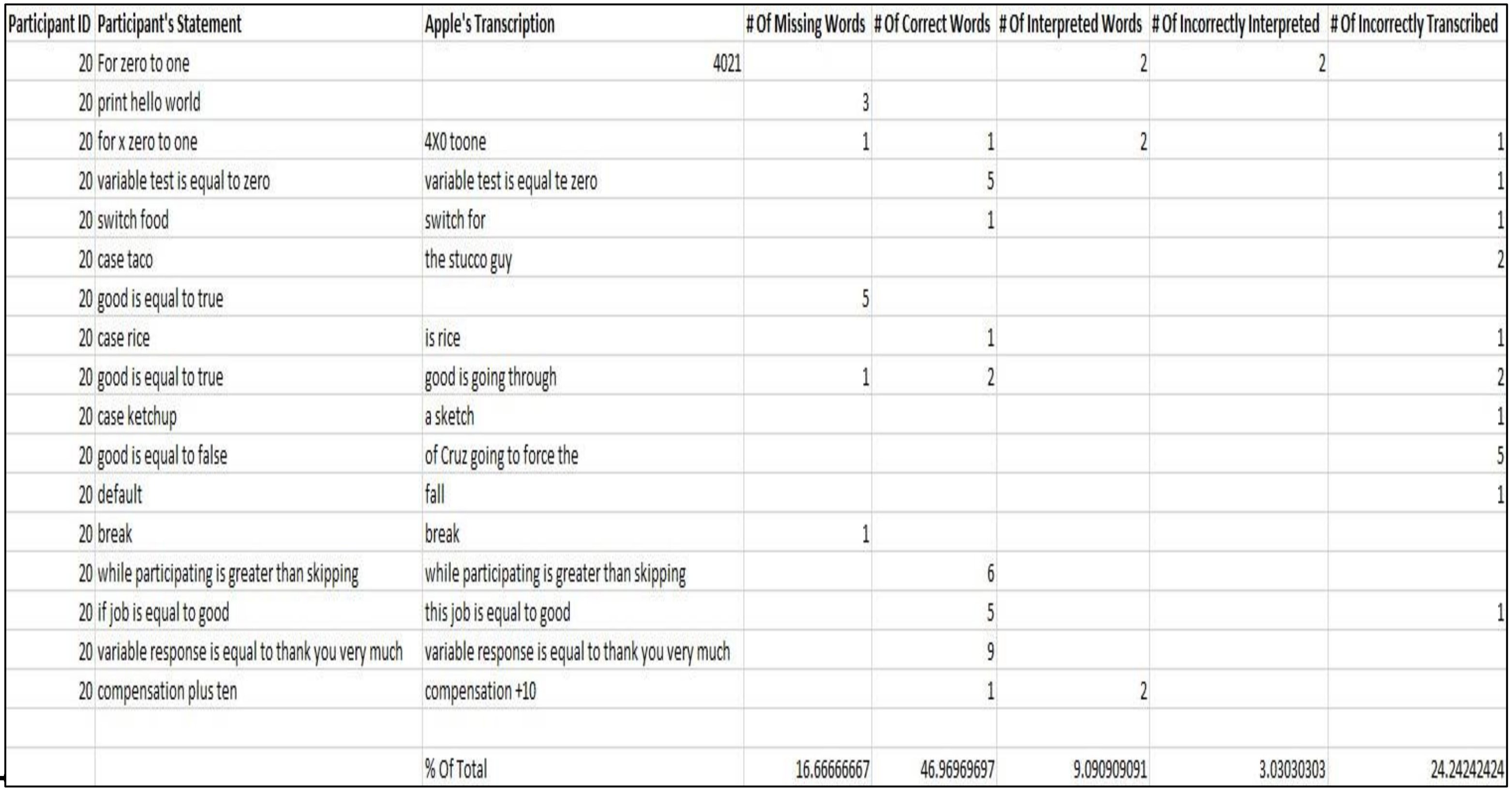
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
18	For underscore zero dot dot dot one open parenthesis	_Zero... One (	1	2	6		
18	print um hello world	print hello world	1	3			
18	for loop x O dot dot dot one open parenthesis	follow up x-ray... One (	1	2	5		2
18	var test equals to ten	that testicles to 10	1	1	1		2
18	close parenthesis	)			2		
18	close parenthesis	)			2		
18	switch of food open parenthesis	switch off food (		2	2		1
18	case taco	case tackle		1			1
18	good equals to true	go request a true		1			3
18	case rice	case rice		2			
18	good equals to true	request a true	1	1			2
18	case ketchup	case ketchup		2			
18	good equals to false	falls	3				1
18	default	default		1			
18	break	break		1			
18	close parenthesis	)			2		
18	while participate participating greater than skipping open parenthesis	why participate participating greater than skipping open finances		6			2
18	if job equals to good open parenthesis	if job equals two good (		4	2		1
18	var response equals to thank you very much	where response equals to thank you very much		7			1
18	close parenthesis	)			2		
18	compensation plus equals to ten	compensation plus equals to 10		4	1		
18	close parenthesis	)			2		
		% Of Total	8.791208791	43.95604396	29.67032967	0	17.58241758

G.17) Participant 19



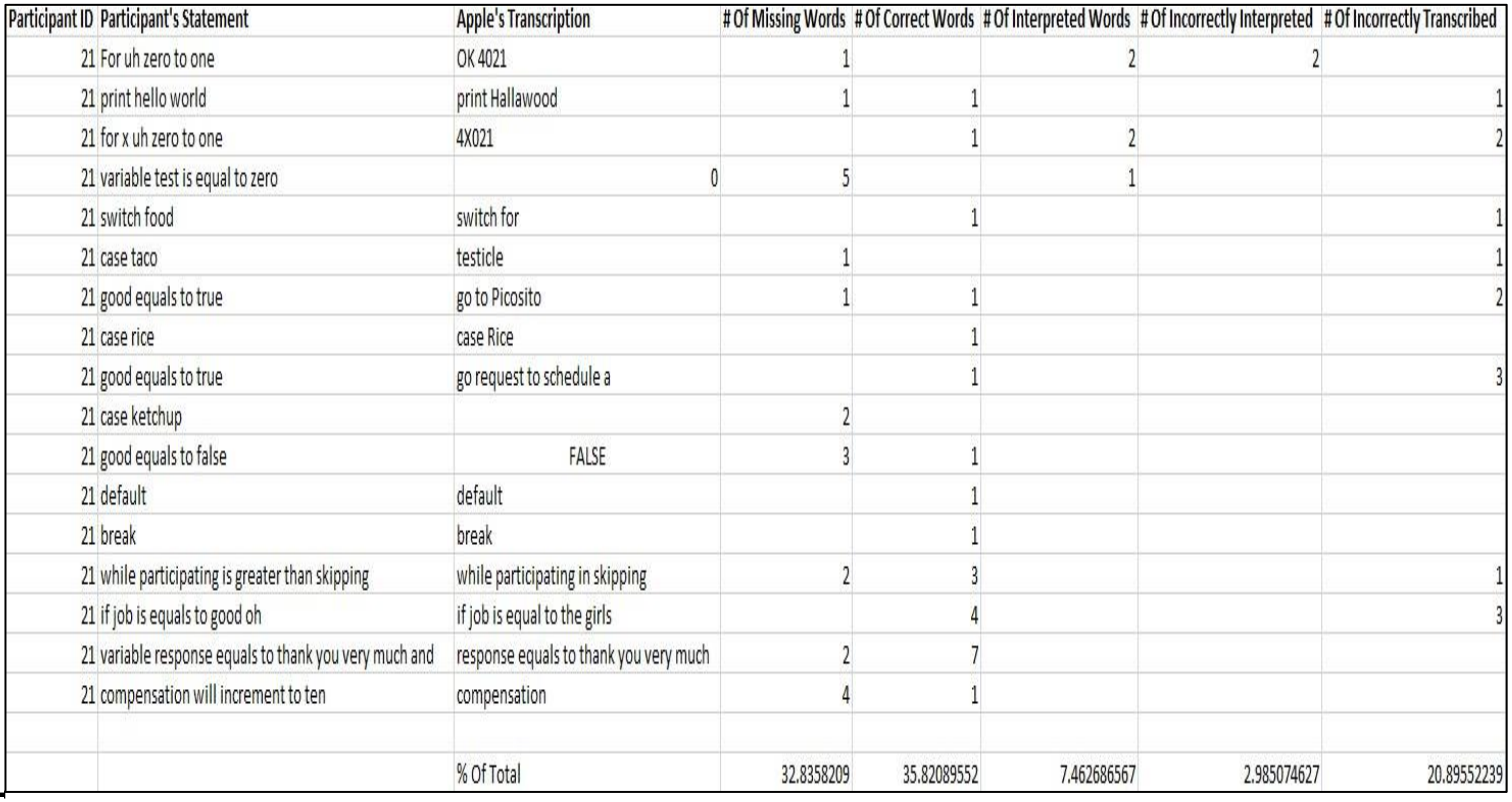
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
19	For underscore zero to one	For_021		1	3	1	
19	print hello world	print hello world		3			
19	for x zero to one	for X02 on		2	1	1	1
19	var test equals ten uh	that test equals 10	1	2	1		1
19	switch food	Switchfoot	1	1			
19	case taco	case taco		2			
19	good equals to true	go to Scholes true		1			3
19	case rice	case right		2			
19	good equals true	good equals true		3			
19	case ketchup	case ketchup		2			
19	good equals false	good equals fault		2			1
19	default	default		1			
19	break	break		1			
19	while participating greater than skipping	while parties participating grade and keeping		2			3
19	if job equals good	his job is called Goode		1			3
19	var response equals thank you very much	where respond to calls thank you very much		5			2
19	compensation plus equals ten	compensation plus equals 10		3	1		
		% Of Total	3.448275862	58.62068966	10.34482759	3.448275862	24.13793103

G.18) Participant 20



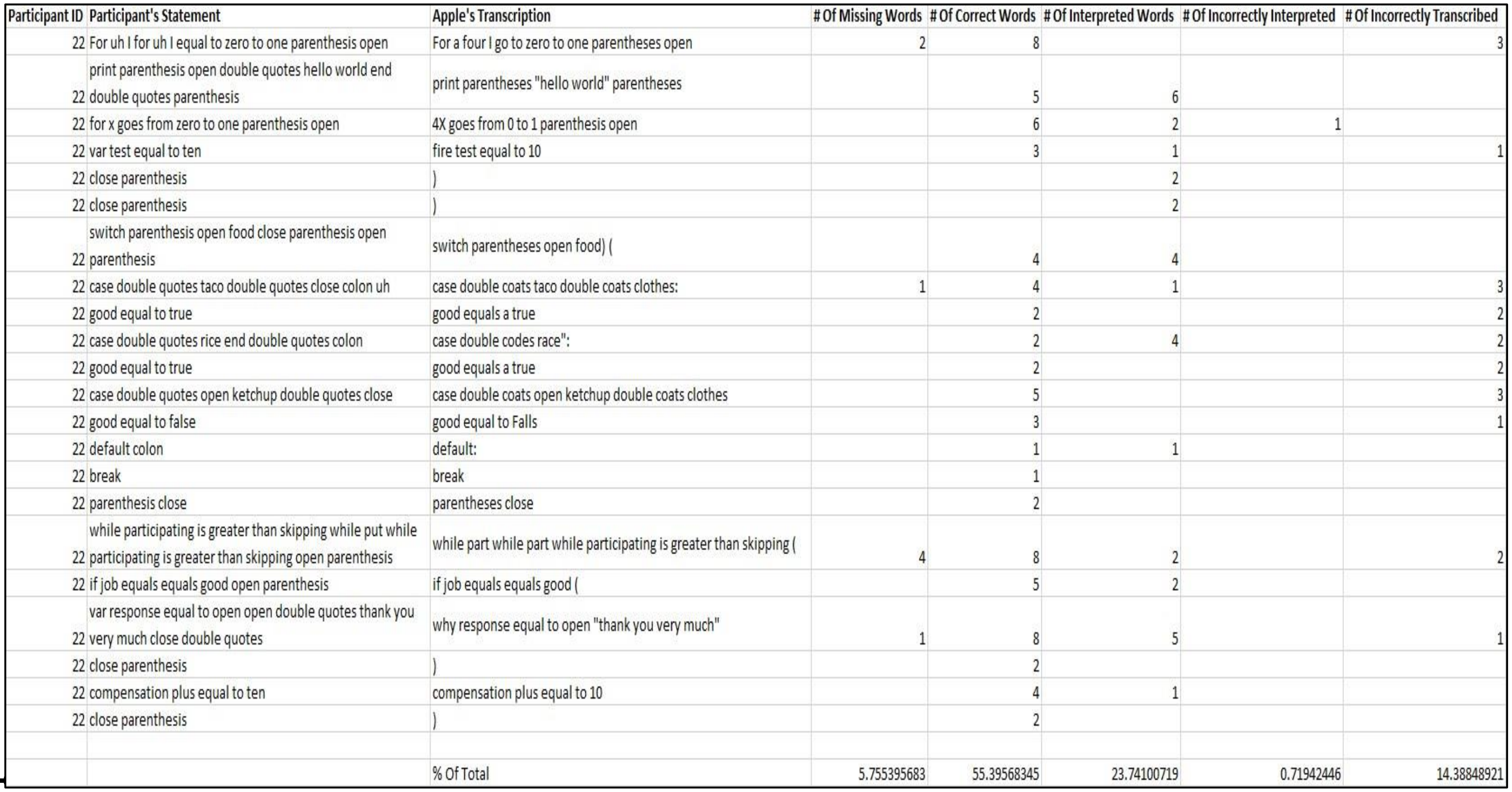
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
20	For zero to one	4021			2	2	
20	print hello world		3				
20	for x zero to one	4X0 toone	1	1	2		1
20	variable test is equal to zero	variable test is equal te zero		5			1
20	switch food	switch for		1			1
20	case taco	the stucco guy					2
20	good is equal to true		5				
20	case rice	is rice		1			1
20	good is equal to true	good is going through	1	2			2
20	case ketchup	a sketch					1
20	good is equal to false	of Cruz going to force the					5
20	default	fall					1
20	break	break	1				
20	while participating is greater than skipping	while participating is greater than skipping		6			
20	if job is equal to good	this job is equal to good		5			1
20	variable response is equal to thank you very much	variable response is equal to thank you very much		9			
20	compensation plus ten	compensation +10		1	2		
		% Of Total	16.66666667	46.96969697	9.090909091	3.03030303	24.24242424

G.19) Participant 21



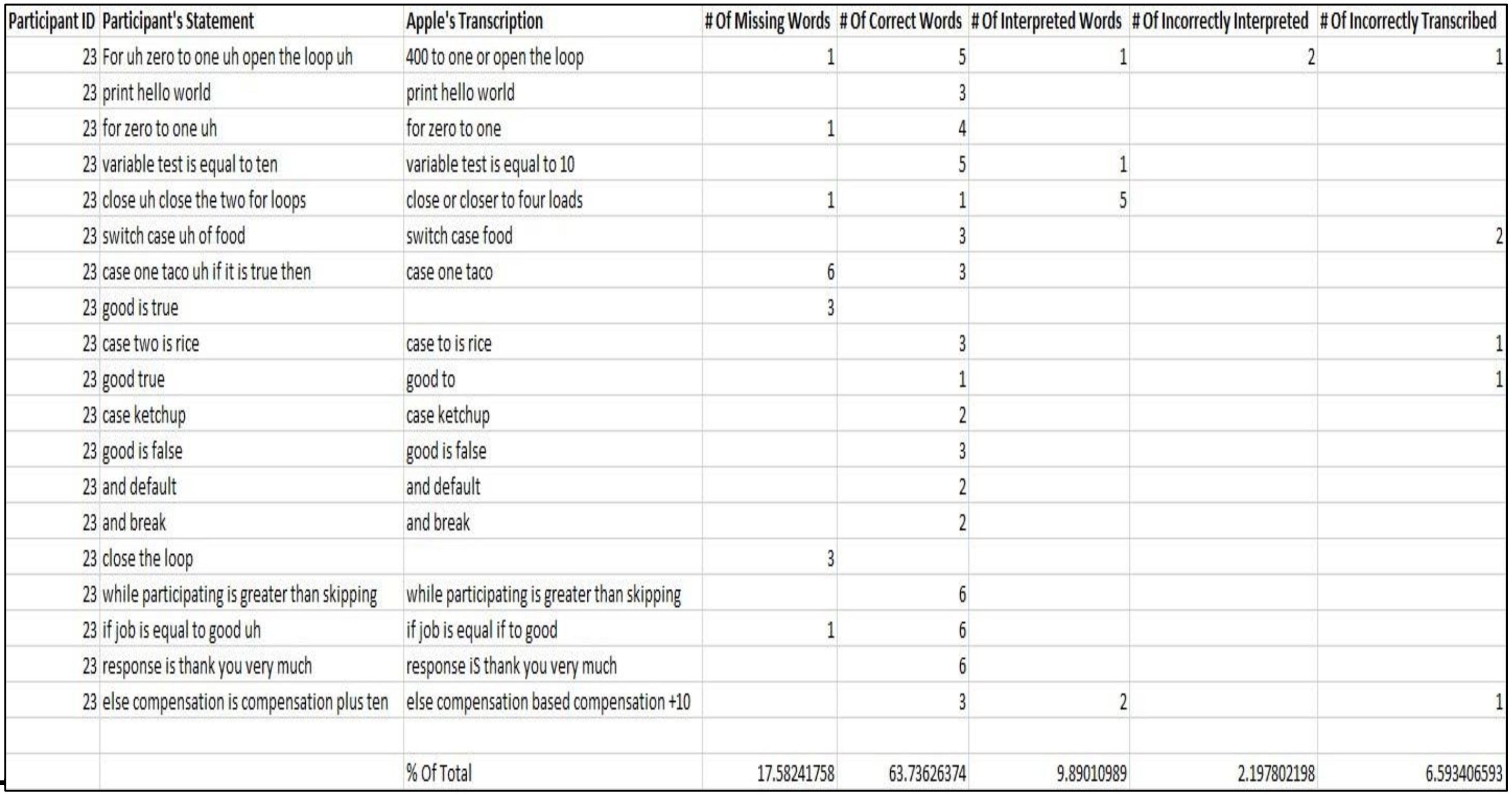
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
21	For uh zero to one	OK 4021	1		2	2	
21	print hello world	print Hallawood	1	1			1
21	for x uh zero to one	4X021		1	2		2
21	variable test is equal to zero	0	5		1		
21	switch food	switch for		1			1
21	case taco	testicle	1				1
21	good equals to true	go to Picosito	1	1			2
21	case rice	case Rice		1			
21	good equals to true	go request to schedule a		1			3
21	case ketchup		2				
21	good equals to false	FALSE	3	1			
21	default	default		1			
21	break	break		1			
21	while participating is greater than skipping	while participating in skipping	2	3			1
21	if job is equals to good oh	if job is equal to the girls		4			3
21	variable response equals to thank you very much and	response equals to thank you very much	2	7			
21	compensation will increment to ten	compensation	4	1			
		% Of Total	32.8358209	35.82089552	7.462686567	2.985074627	20.89552239

G.20) Participant 22



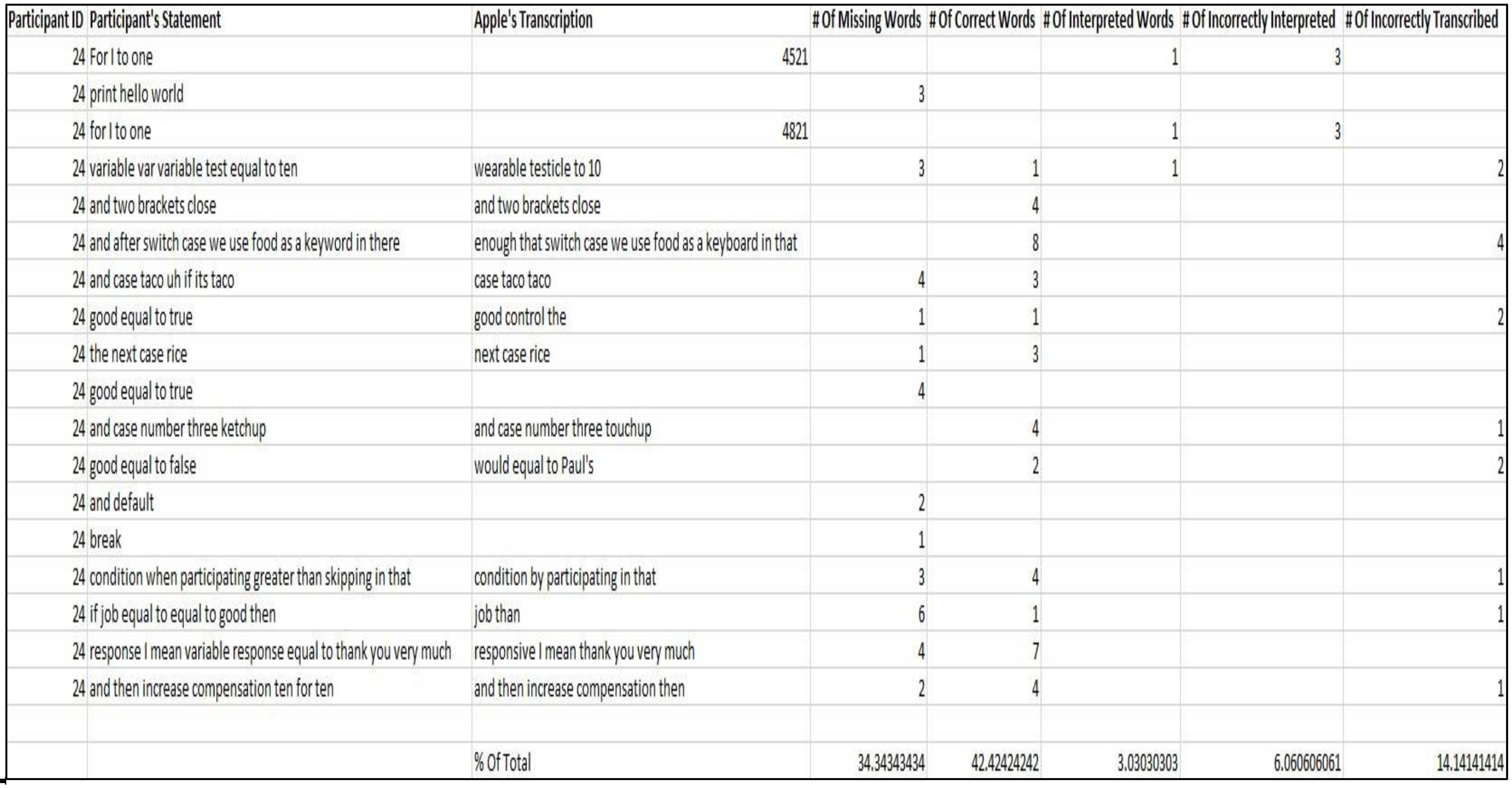
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
22	For uh I for uh I equal to zero to one parenthesis open	For a four I go to zero to one parentheses open	2	8			3
22	print parenthesis open double quotes hello world end double quotes parenthesis	print parentheses "hello world" parentheses		5	6		
22	for x goes from zero to one parenthesis open	4X goes from 0 to 1 parenthesis open		6	2	1	
22	var test equal to ten	fire test equal to 10		3	1		1
22	close parenthesis	)			2		
22	close parenthesis	)			2		
22	switch parenthesis open food close parenthesis open parenthesis	switch parentheses open food) (		4	4		
22	case double quotes taco double quotes close colon uh	case double coats taco double coats clothes:	1	4	1		3
22	good equal to true	good equals a true		2			2
22	case double quotes rice end double quotes colon	case double codes race":		2	4		2
22	good equal to true	good equals a true		2			2
22	case double quotes open ketchup double quotes close	case double coats open ketchup double coats clothes		5			3
22	good equal to false	good equal to Falls		3			1
22	default colon	default:		1	1		
22	break	break		1			
22	parenthesis close	parentheses close		2			
22	while participating is greater than skipping while put while participating is greater than skipping open parenthesis	while part while part while participating is greater than skipping (	4	8	2		2
22	if job equals equals good open parenthesis	if job equals equals good (		5	2		
22	var response equal to open open double quotes thank you very much close double quotes	why response equal to open "thank you very much"	1	8	5		1
22	close parenthesis	)		2			
22	compensation plus equal to ten	compensation plus equal to 10		4	1		
22	close parenthesis	)		2			
		% Of Total	5.755395683	55.39568345	23.74100719	0.71942446	14.38848921

G.21) Participant 23



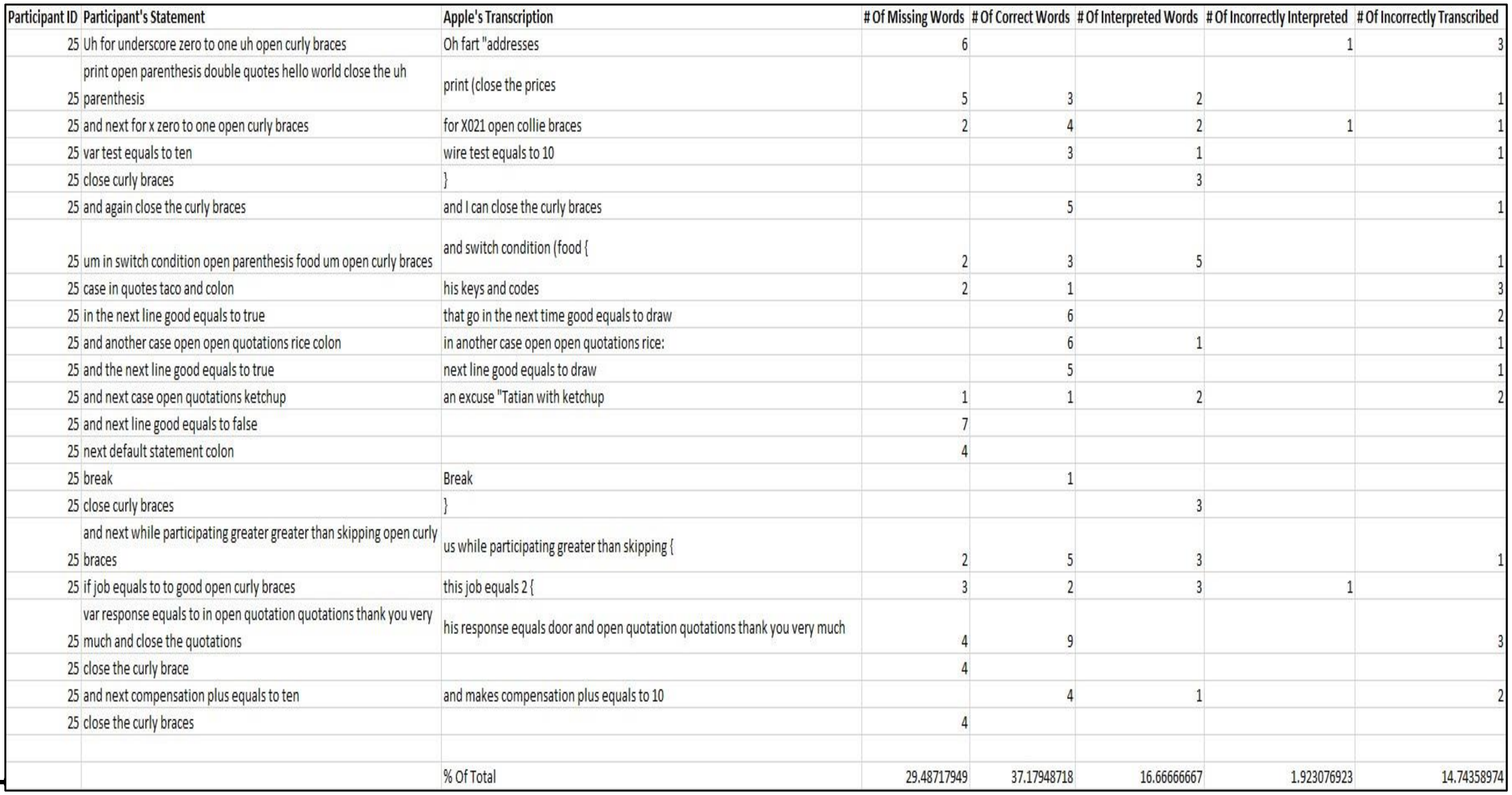
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
23	For uh zero to one uh open the loop uh	400 to one or open the loop	1	5	1	2	1
23	print hello world	print hello world		3			
23	for zero to one uh	for zero to one	1	4			
23	variable test is equal to ten	variable test is equal to 10		5	1		
23	close uh close the two for loops	close or closer to four loads	1	1	5		
23	switch case uh of food	switch case food		3			2
23	case one taco uh if it is true then	case one taco	6	3			
23	good is true		3				
23	case two is rice	case to is rice		3			1
23	good true	good to		1			1
23	case ketchup	case ketchup		2			
23	good is false	good is false		3			
23	and default	and default		2			
23	and break	and break		2			
23	close the loop		3				
23	while participating is greater than skipping	while participating is greater than skipping		6			
23	if job is equal to good uh	if job is equal if to good	1	6			
23	response is thank you very much	response iS thank you very much		6			
23	else compensation is compensation plus ten	else compensation based compensation +10		3	2		1
		% Of Total	17.58241758	63.73626374	9.89010989	2.197802198	6.593406593

G.22) Participant 24



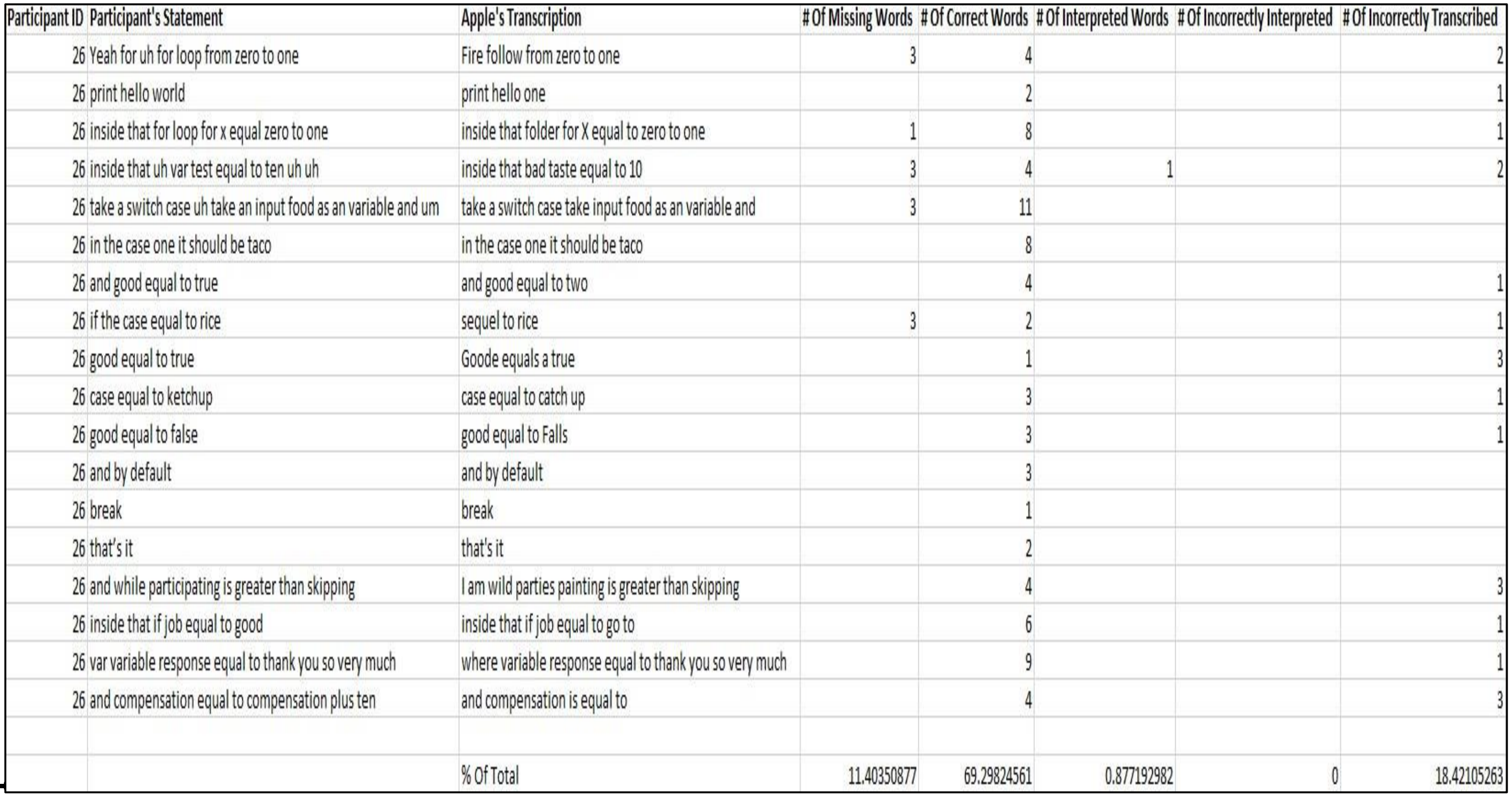
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
24	For I to one	4521			1	3	
24	print hello world		3				
24	for I to one	4821			1	3	
24	variable var variable test equal to ten	wearable testicle to 10	3	1	1		2
24	and two brackets close	and two brackets close		4			
24	and after switch case we use food as a keyword in there	enough that switch case we use food as a keyboard in that		8			4
24	and case taco uh if its taco	case taco taco	4	3			
24	good equal to true	good control the	1	1			2
24	the next case rice	next case rice	1	3			
24	good equal to true		4				
24	and case number three ketchup	and case number three touchup		4			1
24	good equal to false	would equal to Paul's		2			2
24	and default		2				
24	break		1				
24	condition when participating greater than skipping in that	condition by participating in that	3	4			1
24	if job equal to equal to good then	job than	6	1			1
24	response I mean variable response equal to thank you very much	responsive I mean thank you very much	4	7			
24	and then increase compensation ten for ten	and then increase compensation then	2	4			1
		% Of Total	34.34343434	42.42424242	3.03030303	6.060606061	14.14141414

G.23) Participant 25



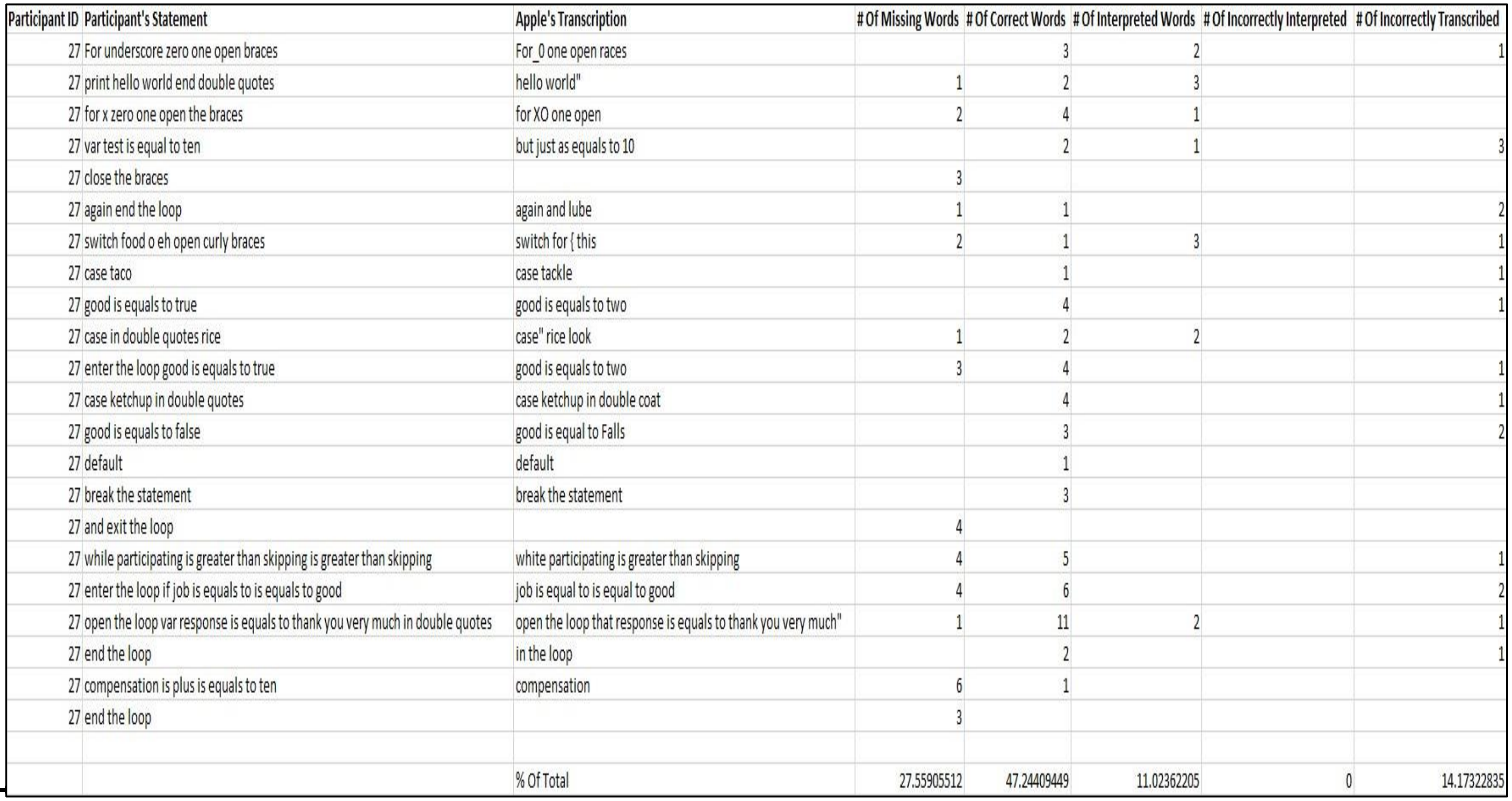
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
25	Uh for underscore zero to one uh open curly braces	Oh fart "addresses	6			1	3
25	print open parenthesis double quotes hello world close the uh parenthesis	print (close the prices	5	3	2		1
25	and next for x zero to one open curly braces	for X021 open collie braces	2	4	2	1	1
25	var test equals to ten	wire test equals to 10		3	1		1
25	close curly braces	}			3		
25	and again close the curly braces	and I can close the curly braces		5			1
25	um in switch condition open parenthesis food um open curly braces	and switch condition (food {	2	3	5		1
25	case in quotes taco and colon	his keys and codes	2	1			3
25	in the next line good equals to true	that go in the next time good equals to draw		6			2
25	and another case open open quotations rice colon	in another case open open quotations rice:		6	1		1
25	and the next line good equals to true	next line good equals to draw		5			1
25	and next case open quotations ketchup	an excuse "Tatian with ketchup	1	1	2		2
25	and next line good equals to false		7				
25	next default statement colon		4				
25	break	Break		1			
25	close curly braces	}			3		
25	and next while participating greater greater than skipping open curly braces	us while participating greater than skipping {	2	5	3		1
25	if job equals to to good open curly braces	this job equals 2 {	3	2	3	1	
25	var response equals to in open quotation quotations thank you very much and close the quotations	his response equals door and open quotation quotations thank you very much	4	9			3
25	close the curly brace		4				
25	and next compensation plus equals to ten	and makes compensation plus equals to 10		4	1		2
25	close the curly braces		4				
		% Of Total	29.48717949	37.17948718	16.66666667	1.923076923	14.74358974

G.24) Participant 26



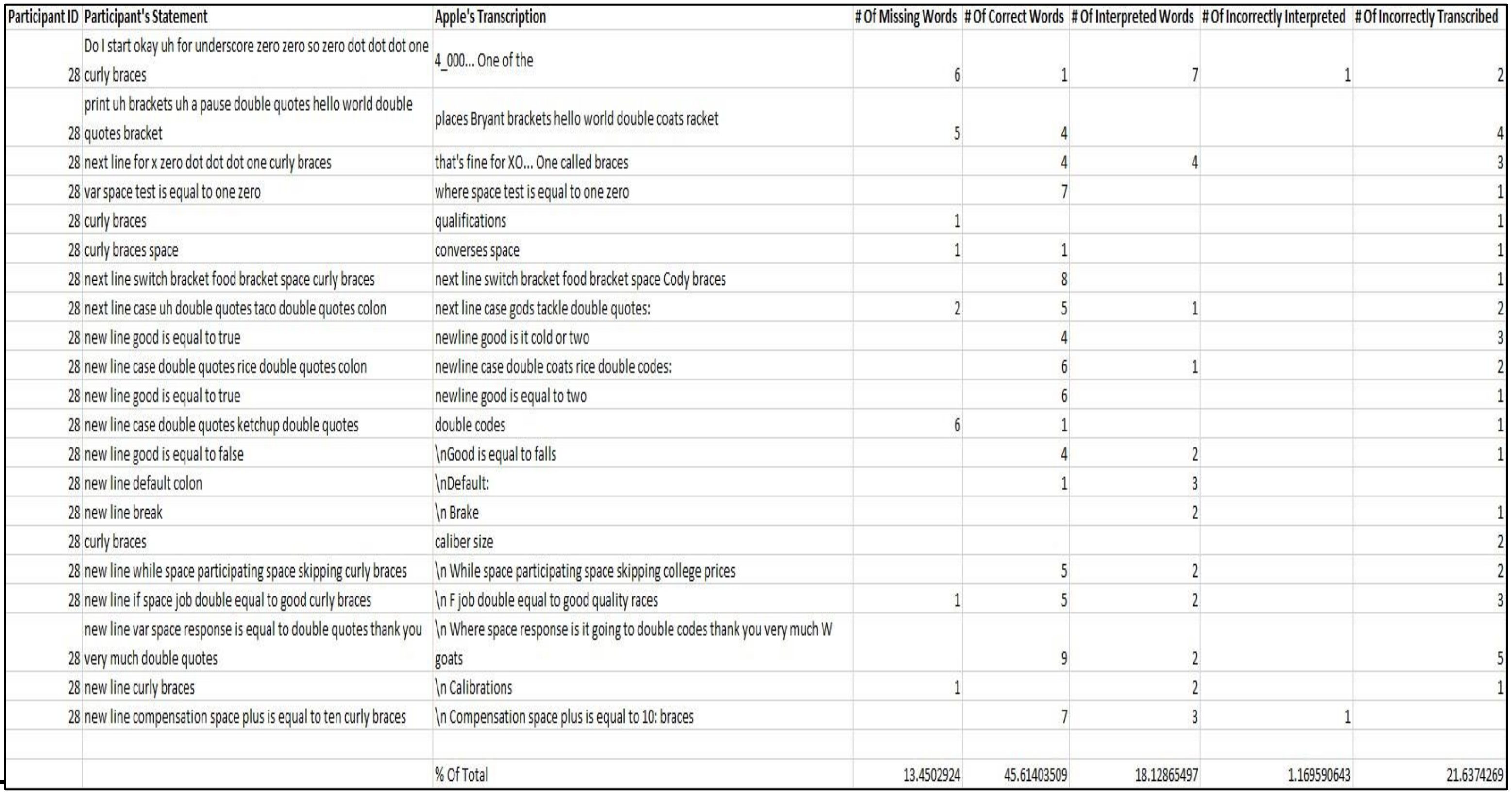
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
26	Yeah for uh for loop from zero to one	Fire follow from zero to one	3	4			2
26	print hello world	print hello one		2			1
26	inside that for loop for x equal zero to one	inside that folder for X equal to zero to one	1	8			1
26	inside that uh var test equal to ten uh uh	inside that bad taste equal to 10	3	4	1		2
26	take a switch case uh take an input food as an variable and um	take a switch case take input food as an variable and	3	11			
26	in the case one it should be taco	in the case one it should be taco		8			
26	and good equal to true	and good equal to two		4			1
26	if the case equal to rice	sequel to rice	3	2			1
26	good equal to true	Goode equals a true		1			3
26	case equal to ketchup	case equal to catch up		3			1
26	good equal to false	good equal to Falls		3			1
26	and by default	and by default		3			
26	break	break		1			
26	that's it	that's it		2			
26	and while participating is greater than skipping	I am wild parties painting is greater than skipping		4			3
26	inside that if job equal to good	inside that if job equal to go to		6			1
26	var variable response equal to thank you so very much	where variable response equal to thank you so very much		9			1
26	and compensation equal to compensation plus ten	and compensation is equal to		4			3
		% Of Total	11.40350877	69.29824561	0.877192982	0	18.42105263

G.25) Participant 27



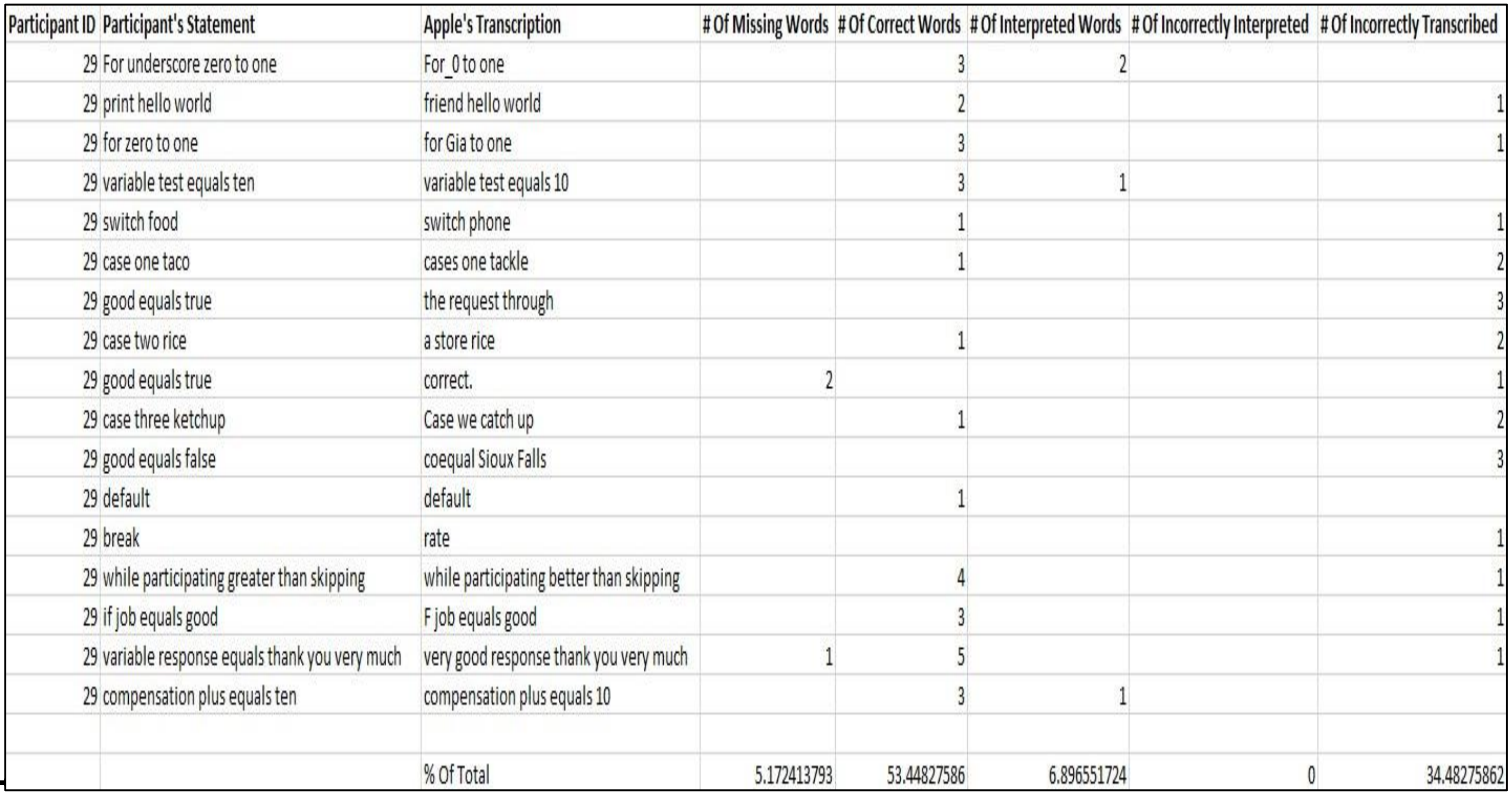
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
27	For underscore zero one open braces	For_0 one open races		3	2		1
27	print hello world end double quotes	hello world"	1	2	3		
27	for x zero one open the braces	for XO one open	2	4	1		
27	var test is equal to ten	but just as equals to 10		2	1		3
27	close the braces		3				
27	again end the loop	again and lube	1	1			2
27	switch food o eh open curly braces	switch for { this	2	1	3		1
27	case taco	case tackle		1			1
27	good is equals to true	good is equals to two		4			1
27	case in double quotes rice	case" rice look	1	2	2		
27	enter the loop good is equals to true	good is equals to two	3	4			1
27	case ketchup in double quotes	case ketchup in double coat		4			1
27	good is equals to false	good is equal to Falls		3			2
27	default	default		1			
27	break the statement	break the statement		3			
27	and exit the loop		4				
27	while participating is greater than skipping is greater than skipping	white participating is greater than skipping	4	5			1
27	enter the loop if job is equals to is equals to good	job is equal to is equal to good	4	6			2
27	open the loop var response is equals to thank you very much in double quotes	open the loop that response is equals to thank you very much"	1	11	2		1
27	end the loop	in the loop		2			1
27	compensation is plus is equals to ten	compensation	6	1			
27	end the loop		3				
		% Of Total	27.55905512	47.24409449	11.02362205	0	14.17322835

G.26) Participant 28



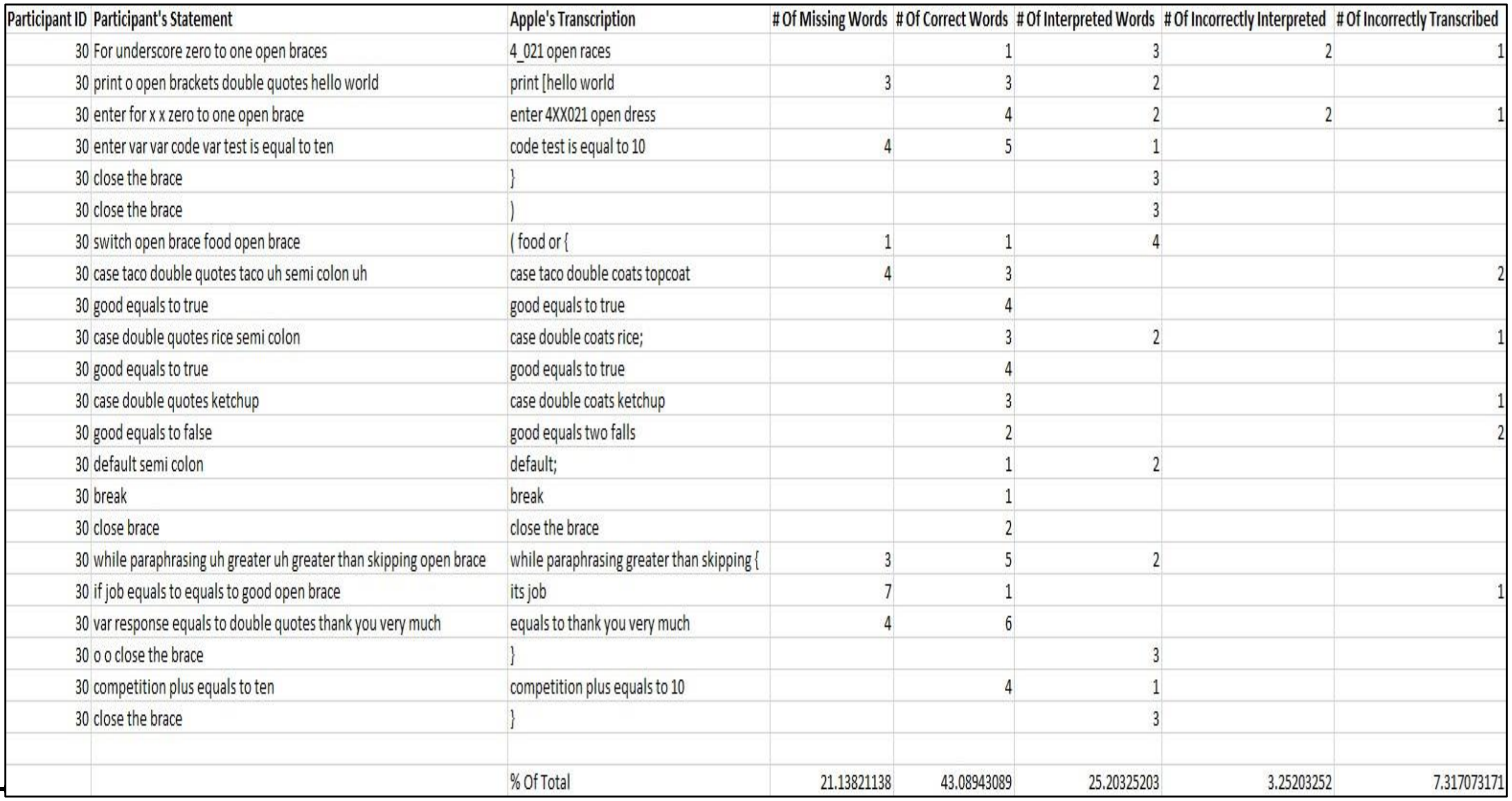
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
28	Do I start okay uh for underscore zero zero so zero dot dot dot one curly braces	4_000... One of the	6	1	7	1	2
28	print uh brackets uh a pause double quotes hello world double quotes bracket	places Bryant brackets hello world double coats racket	5	4			4
28	next line for x zero dot dot dot one curly braces	that's fine for XO... One called braces		4	4		3
28	var space test is equal to one zero	where space test is equal to one zero		7			1
28	curly braces	qualifications	1				1
28	curly braces space	converses space	1	1			1
28	next line switch bracket food bracket space curly braces	next line switch bracket food bracket space Cody braces		8			1
28	next line case uh double quotes taco double quotes colon	next line case gods tackle double quotes:	2	5	1		2
28	new line good is equal to true	newline good is it cold or two		4			3
28	new line case double quotes rice double quotes colon	newline case double coats rice double codes:		6	1		2
28	new line good is equal to true	newline good is equal to two		6			1
28	new line case double quotes ketchup double quotes	double codes	6	1			1
28	new line good is equal to false	\nGood is equal to falls		4	2		1
28	new line default colon	\nDefault:		1	3		
28	new line break	\n Brake			2		1
28	curly braces	caliber size					2
28	new line while space participating space skipping curly braces	\n While space participating space skipping college prices		5	2		2
28	new line if space job double equal to good curly braces	\n F job double equal to good quality races	1	5	2		3
28	new line var space response is equal to double quotes thank you very much double quotes	\n Where space response is it going to double codes thank you very much W goats		9	2		5
28	new line curly braces	\n Calibrations	1		2		1
28	new line compensation space plus is equal to ten curly braces	\n Compensation space plus is equal to 10: braces		7	3	1	
		% Of Total	13.4502924	45.61403509	18.12865497	1.169590643	21.6374269

G.27) Participant 29



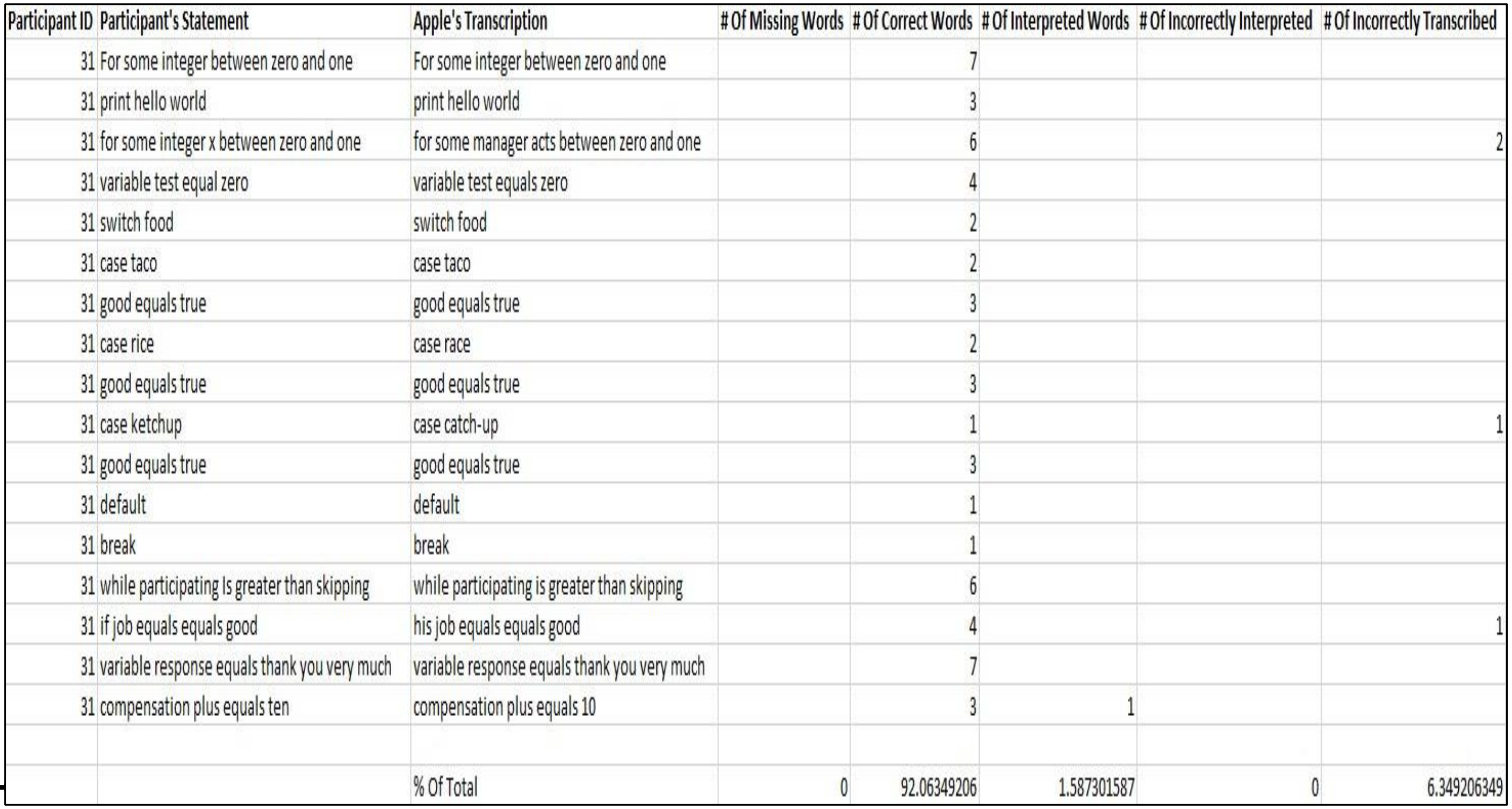
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
29	For underscore zero to one	For_0 to one		3	2		
29	print hello world	friend hello world		2			1
29	for zero to one	for Gia to one		3			1
29	variable test equals ten	variable test equals 10		3	1		
29	switch food	switch phone		1			1
29	case one taco	cases one tackle		1			2
29	good equals true	the request through					3
29	case two rice	a store rice		1			2
29	good equals true	correct.	2				1
29	case three ketchup	Case we catch up		1			2
29	good equals false	coequal Sioux Falls					3
29	default	default		1			
29	break	rate					1
29	while participating greater than skipping	while participating better than skipping		4			1
29	if job equals good	F job equals good		3			1
29	variable response equals thank you very much	very good response thank you very much	1	5			1
29	compensation plus equals ten	compensation plus equals 10		3	1		
		% Of Total	5.172413793	53.44827586	6.896551724	0	34.48275862

G.28) Participant 30



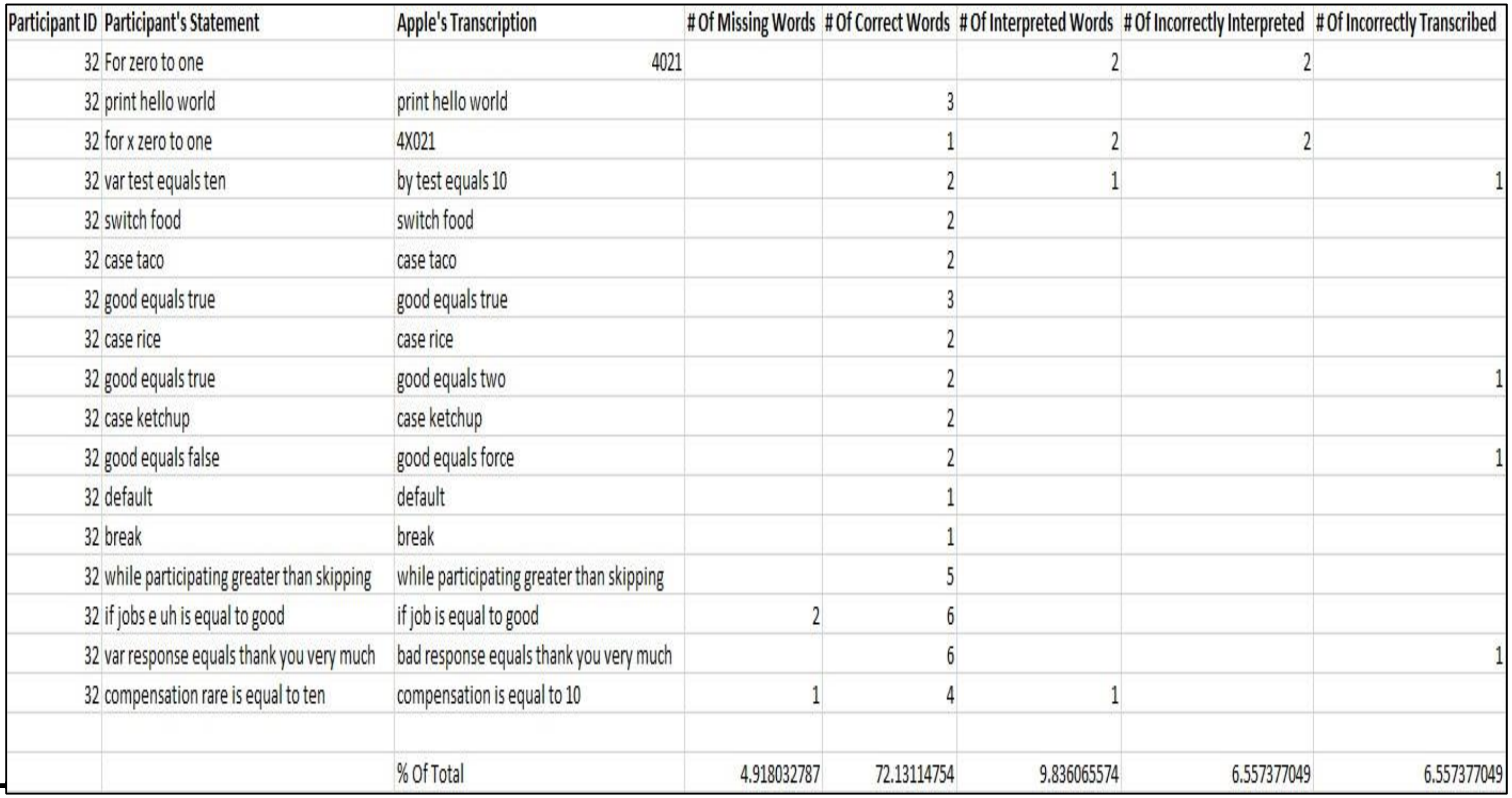
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
30	For underscore zero to one open braces	4_021 open races		1	3	2	1
30	print o open brackets double quotes hello world	print [hello world	3	3	2		
30	enter for x x zero to one open brace	enter 4XX021 open dress		4	2	2	1
30	enter var var code var test is equal to ten	code test is equal to 10	4	5	1		
30	close the brace	}			3		
30	close the brace	)			3		
30	switch open brace food open brace	(food or {	1	1	4		
30	case taco double quotes taco uh semi colon uh	case taco double coats topcoat	4	3			2
30	good equals to true	good equals to true		4			
30	case double quotes rice semi colon	case double coats rice;		3	2		1
30	good equals to true	good equals to true		4			
30	case double quotes ketchup	case double coats ketchup		3			1
30	good equals to false	good equals two falls		2			2
30	default semi colon	default;		1	2		
30	break	break		1			
30	close brace	close the brace		2			
30	while paraphrasing uh greater uh greater than skipping open brace	while paraphrasing greater than skipping {	3	5	2		
30	if job equals to equals to good open brace	its job	7	1			1
30	var response equals to double quotes thank you very much	equals to thank you very much	4	6			
30	o o close the brace	}			3		
30	competition plus equals to ten	competition plus equals to 10		4	1		
30	close the brace	}			3		
		% Of Total	21.13821138	43.08943089	25.20325203	3.25203252	7.317073171

G.29) Participant 31



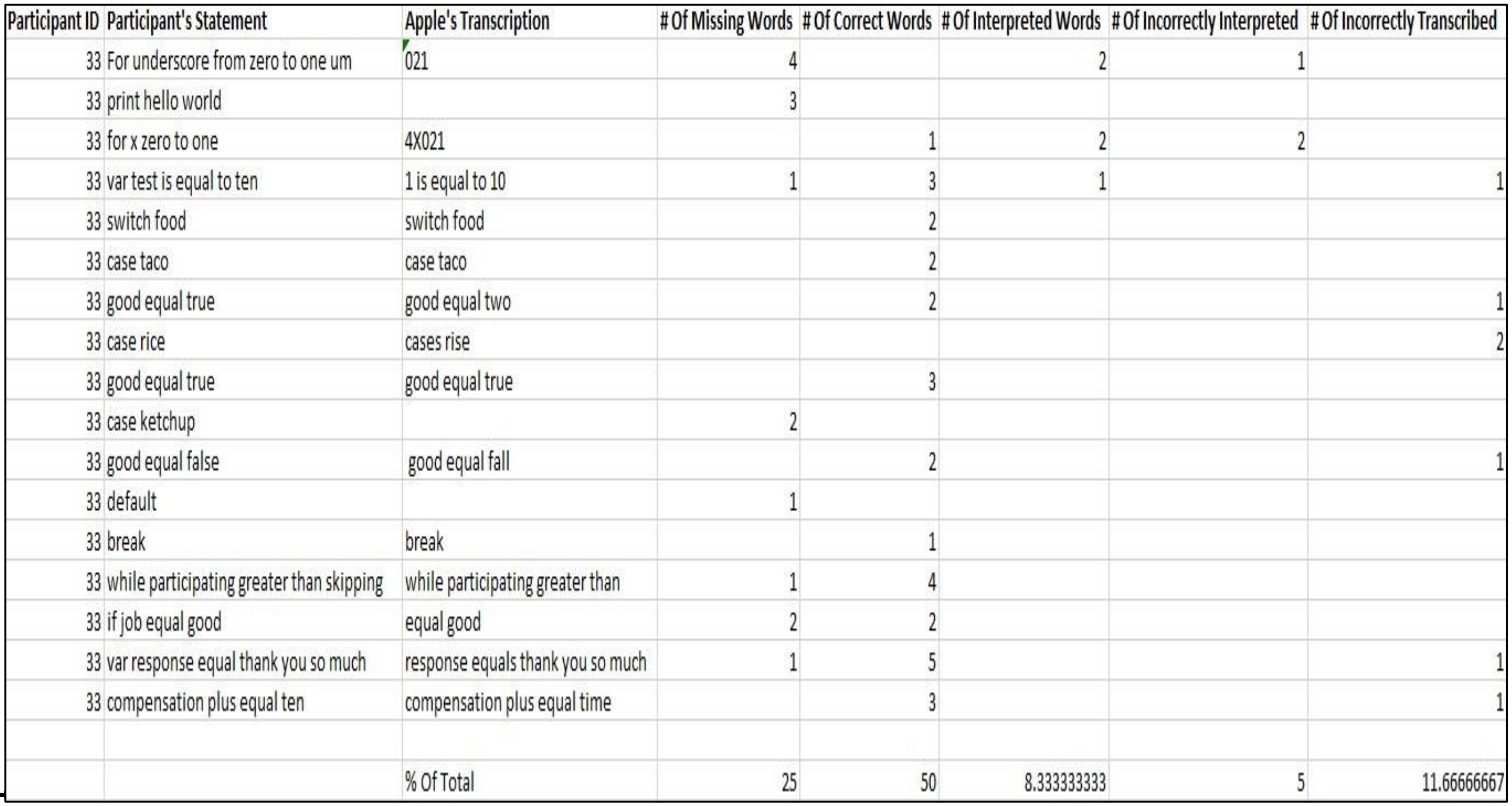
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
31	For some integer between zero and one	For some integer between zero and one		7			
31	print hello world	print hello world		3			
31	for some integer x between zero and one	for some manager acts between zero and one		6			2
31	variable test equal zero	variable test equals zero		4			
31	switch food	switch food		2			
31	case taco	case taco		2			
31	good equals true	good equals true		3			
31	case rice	case race		2			
31	good equals true	good equals true		3			
31	case ketchup	case catch-up		1			1
31	good equals true	good equals true		3			
31	default	default		1			
31	break	break		1			
31	while participating Is greater than skipping	while participating is greater than skipping		6			
31	if job equals equals good	his job equals equals good		4			1
31	variable response equals thank you very much	variable response equals thank you very much		7			
31	compensation plus equals ten	compensation plus equals 10		3	1		
		% Of Total	0	92.06349206	1.587301587	0	6.349206349

G.30) Participant 32



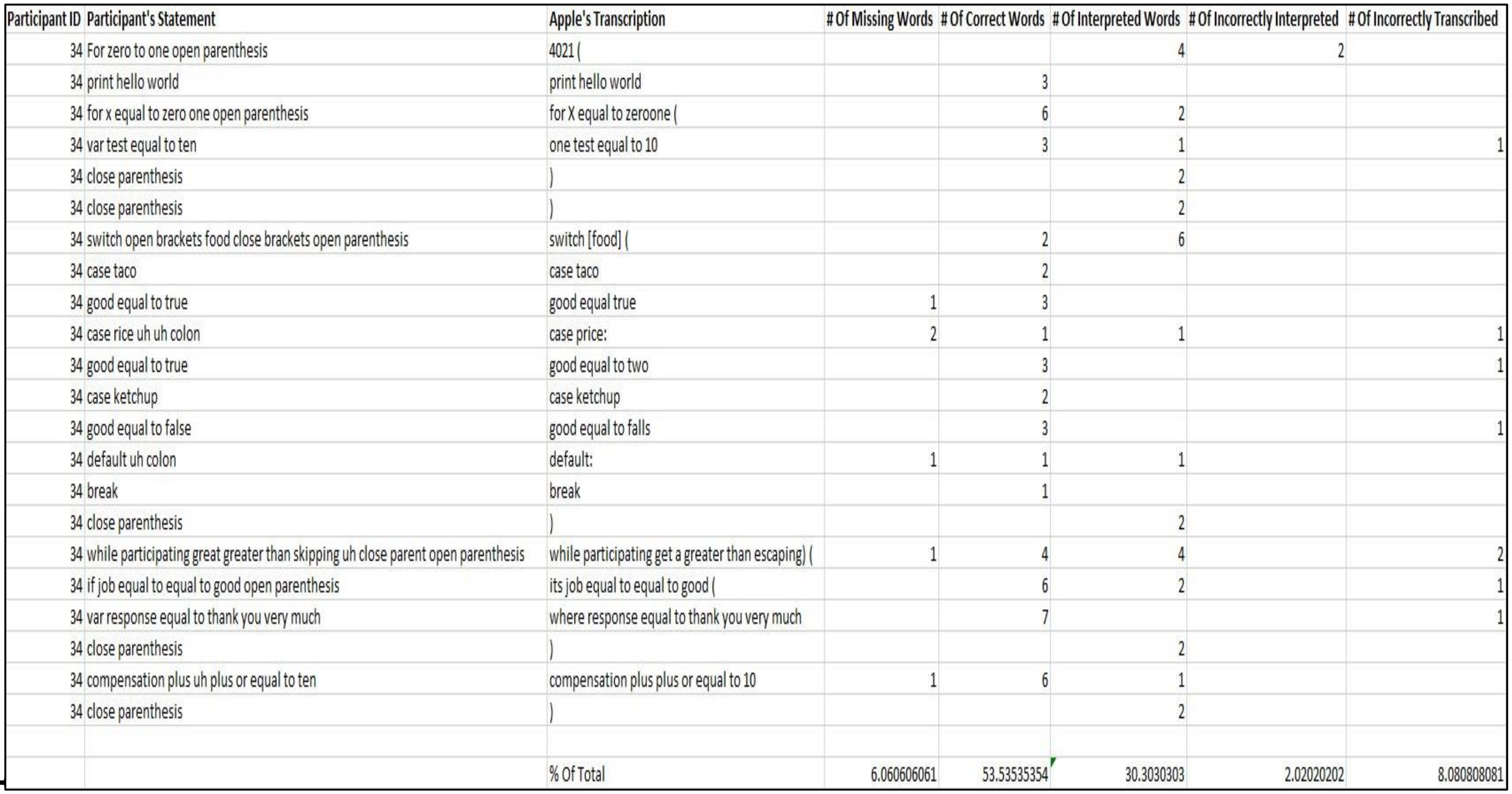
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
32	For zero to one	4021			2	2	
32	print hello world	print hello world		3			
32	for x zero to one	4X021		1	2	2	
32	var test equals ten	by test equals 10		2	1		1
32	switch food	switch food		2			
32	case taco	case taco		2			
32	good equals true	good equals true		3			
32	case rice	case rice		2			
32	good equals true	good equals two		2			1
32	case ketchup	case ketchup		2			
32	good equals false	good equals force		2			1
32	default	default		1			
32	break	break		1			
32	while participating greater than skipping	while participating greater than skipping		5			
32	if jobs e uh is equal to good	if job is equal to good	2	6			
32	var response equals thank you very much	bad response equals thank you very much		6			1
32	compensation rare is equal to ten	compensation is equal to 10	1	4	1		
		% Of Total	4.918032787	72.13114754	9.836065574	6.557377049	6.557377049

G.31) Participant 33



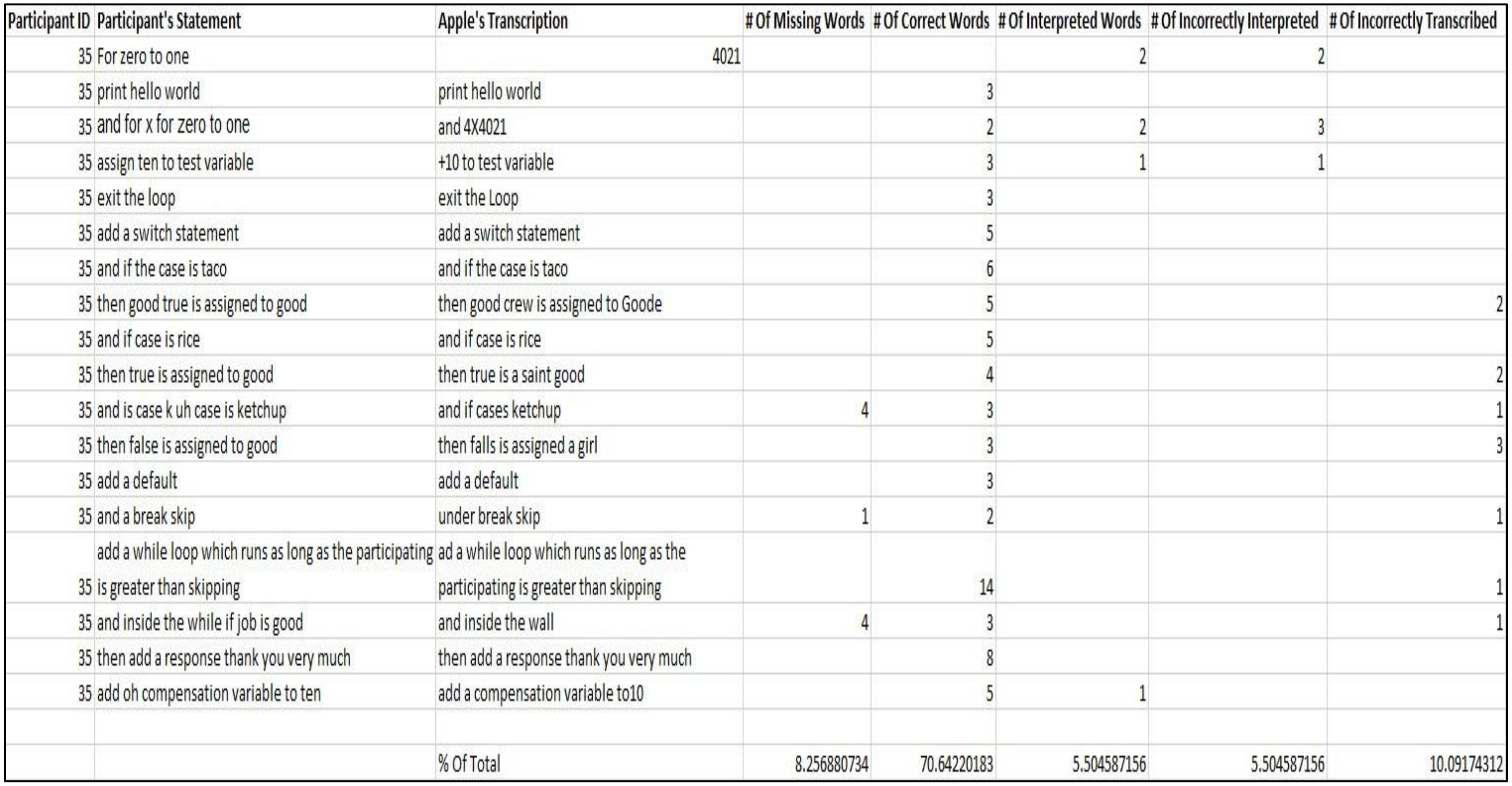
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
33	For underscore from zero to one um	021	4		2	1	
33	print hello world		3				
33	for x zero to one	4X021		1	2	2	
33	var test is equal to ten	1 is equal to 10	1	3	1		1
33	switch food	switch food		2			
33	case taco	case taco		2			
33	good equal true	good equal two		2			1
33	case rice	cases rise					2
33	good equal true	good equal true		3			
33	case ketchup		2				
33	good equal false	good equal fall		2			1
33	default		1				
33	break	break		1			
33	while participating greater than skipping	while participating greater than	1	4			
33	if job equal good	equal good	2	2			
33	var response equal thank you so much	response equals thank you so much	1	5			1
33	compensation plus equal ten	compensation plus equal time		3			1
		% Of Total	25	50	8.333333333	5	11.66666667

G.32) Participant 34



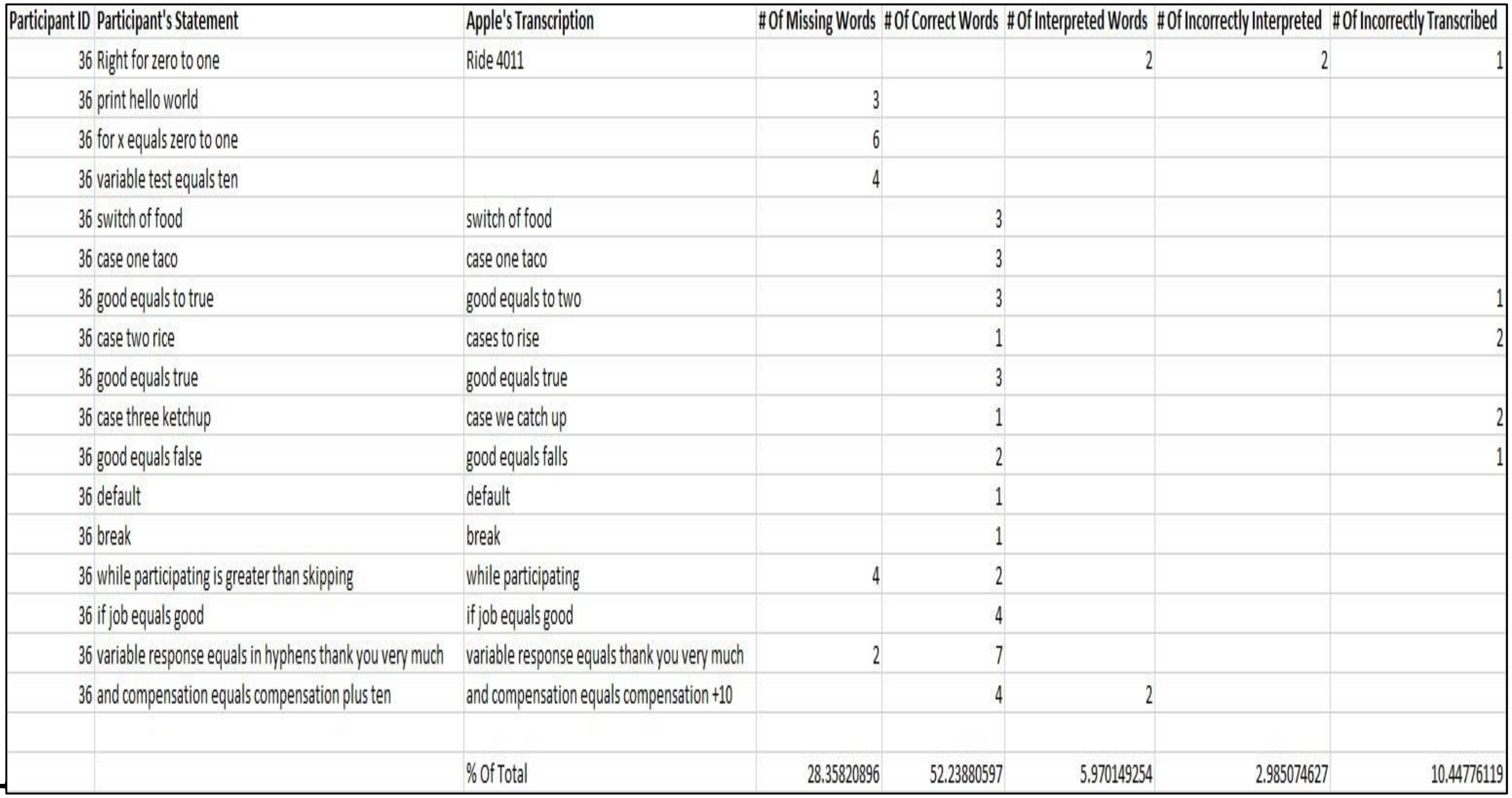
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
34	For zero to one open parenthesis	4021 (			4	2	
34	print hello world	print hello world		3			
34	for x equal to zero one open parenthesis	for X equal to zeroone (		6	2		
34	var test equal to ten	one test equal to 10		3	1		1
34	close parenthesis	)			2		
34	close parenthesis	)			2		
34	switch open brackets food close brackets open parenthesis	switch [food] (		2	6		
34	case taco	case taco		2			
34	good equal to true	good equal true	1	3			
34	case rice uh uh colon	case price:	2	1	1		1
34	good equal to true	good equal to two		3			1
34	case ketchup	case ketchup		2			
34	good equal to false	good equal to falls		3			1
34	default uh colon	default:	1	1	1		
34	break	break		1			
34	close parenthesis	)			2		
34	while participating great greater than skipping uh close parent open parenthesis	while participating get a greater than escaping) (	1	4	4		2
34	if job equal to equal to good open parenthesis	its job equal to equal to good (		6	2		1
34	var response equal to thank you very much	where response equal to thank you very much		7			1
34	close parenthesis	)			2		
34	compensation plus uh plus or equal to ten	compensation plus plus or equal to 10	1	6	1		
34	close parenthesis	)			2		
		% Of Total	6.060606061	53.53535354	30.3030303	2.02020202	8.080808081

G.33) Participant 35



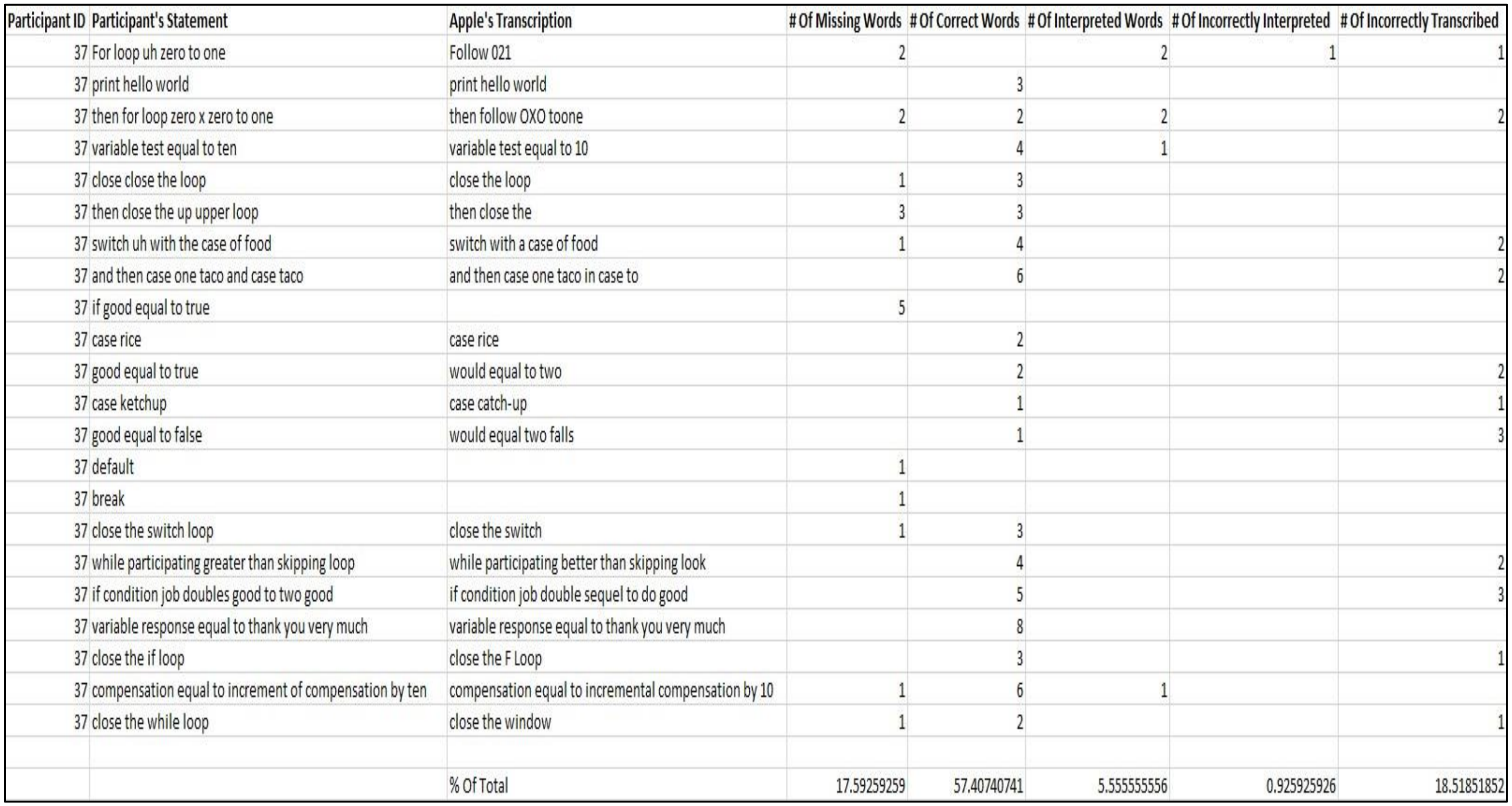
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
35	For zero to one	4021			2	2	
35	print hello world	print hello world		3			
35	and for x for zero to one	and 4X4021		2	2	3	
35	assign ten to test variable	+10 to test variable		3	1	1	
35	exit the loop	exit the Loop		3			
35	add a switch statement	add a switch statement		5			
35	and if the case is taco	and if the case is taco		6			
35	then good true is assigned to good	then good crew is assigned to Goode		5			2
35	and if case is rice	and if case is rice		5			
35	then true is assigned to good	then true is a saint good		4			2
35	and is case k uh case is ketchup	and if cases ketchup	4	3			1
35	then false is assigned to good	then falls is assigned a girl		3			3
35	add a default	add a default		3			
35	and a break skip	under break skip	1	2			1
35	add a while loop which runs as long as the participating is greater than skipping	ad a while loop which runs as long as the participating is greater than skipping		14			1
35	and inside the while if job is good	and inside the wall	4	3			1
35	then add a response thank you very much	then add a response thank you very much		8			
35	add oh compensation variable to ten	add a compensation variable to10		5	1		
		% Of Total	8.256880734	70.64220183	5.504587156	5.504587156	10.09174312

G.34) Participant 36



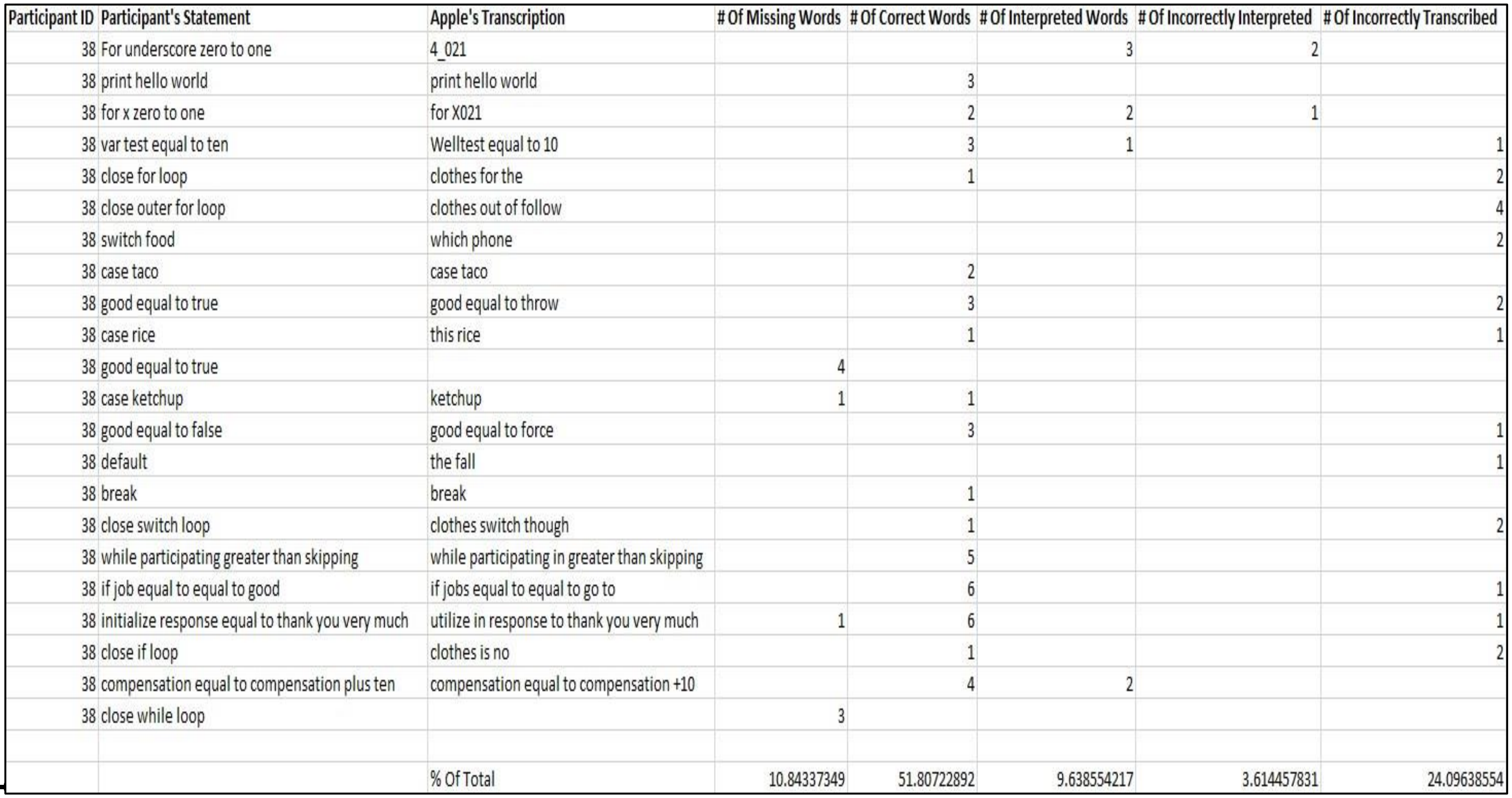
Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
36	Right for zero to one	Ride 4011			2	2	1
36	print hello world		3				
36	for x equals zero to one		6				
36	variable test equals ten		4				
36	switch of food	switch of food		3			
36	case one taco	case one taco		3			
36	good equals to true	good equals to two		3			1
36	case two rice	cases to rise		1			2
36	good equals true	good equals true		3			
36	case three ketchup	case we catch up		1			2
36	good equals false	good equals falls		2			1
36	default	default		1			
36	break	break		1			
36	while participating is greater than skipping	while participating	4	2			
36	if job equals good	if job equals good		4			
36	variable response equals in hyphens thank you very much	variable response equals thank you very much	2	7			
36	and compensation equals compensation plus ten	and compensation equals compensation +10		4	2		
		% Of Total	28.35820896	52.23880597	5.970149254	2.985074627	10.44776119

G.35) Participant 37



Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
37	For loop uh zero to one	Follow 021	2		2	1	1
37	print hello world	print hello world		3			
37	then for loop zero x zero to one	then follow OXO toone	2	2	2		2
37	variable test equal to ten	variable test equal to 10		4	1		
37	close close the loop	close the loop	1	3			
37	then close the up upper loop	then close the	3	3			
37	switch uh with the case of food	switch with a case of food	1	4			2
37	and then case one taco and case taco	and then case one taco in case to		6			2
37	if good equal to true		5				
37	case rice	case rice		2			
37	good equal to true	would equal to two		2			2
37	case ketchup	case catch-up		1			1
37	good equal to false	would equal two falls		1			3
37	default		1				
37	break		1				
37	close the switch loop	close the switch	1	3			
37	while participating greater than skipping loop	while participating better than skipping look		4			2
37	if condition job doubles good to two good	if condition job double sequel to do good		5			3
37	variable response equal to thank you very much	variable response equal to thank you very much		8			
37	close the if loop	close the F Loop		3			1
37	compensation equal to increment of compensation by ten	compensation equal to incremental compensation by 10	1	6	1		
37	close the while loop	close the window	1	2			1
		% Of Total	17.59259259	57.40740741	5.555555556	0.925925926	18.51851852

G.36) Participant 38

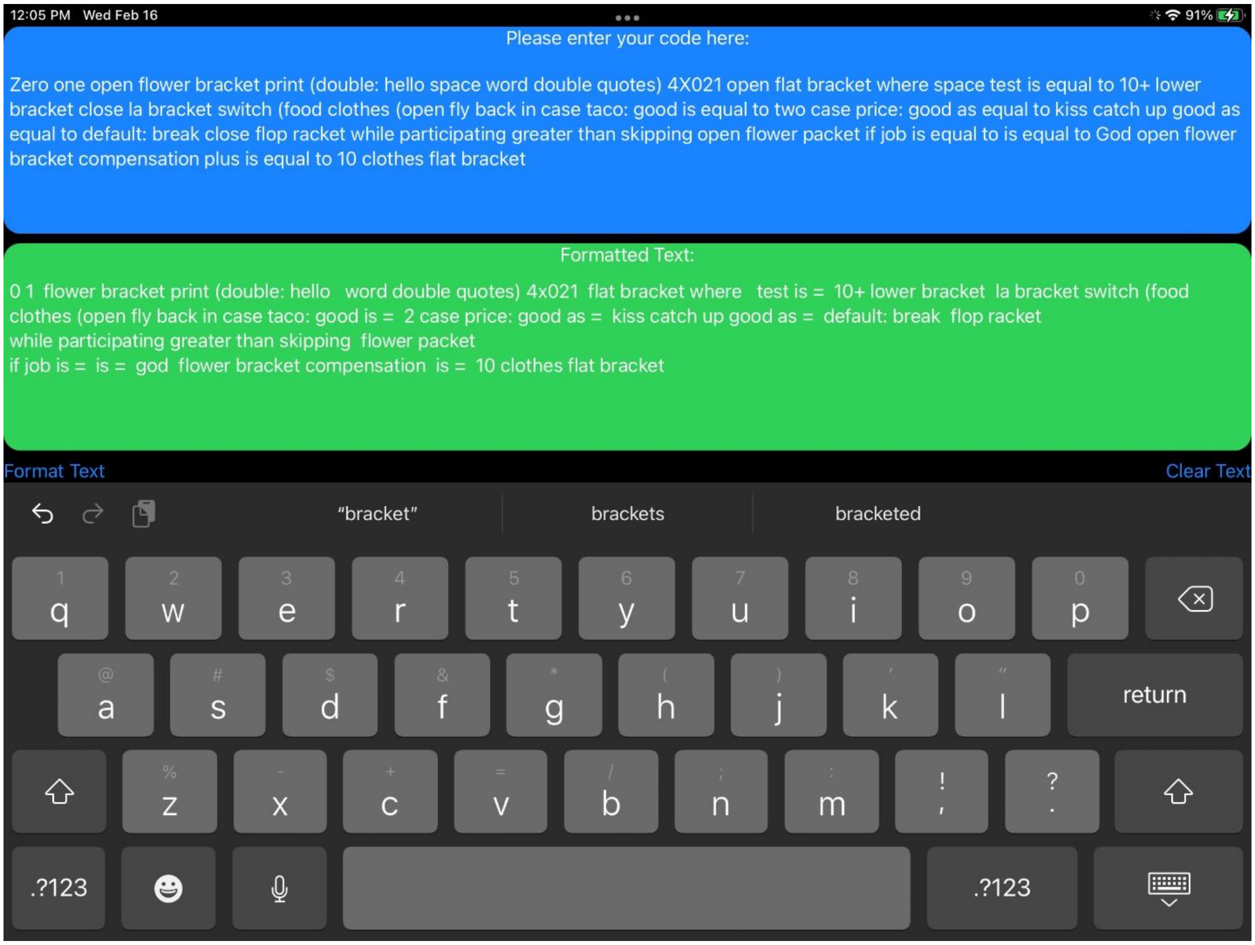


Participant ID	Participant's Statement	Apple's Transcription	# Of Missing Words	# Of Correct Words	# Of Interpreted Words	# Of Incorrectly Interpreted	# Of Incorrectly Transcribed
38	For underscore zero to one	4_021			3	2	
38	print hello world	print hello world		3			
38	for x zero to one	for X021		2	2	1	
38	var test equal to ten	Welltest equal to 10		3	1		1
38	close for loop	clothes for the		1			2
38	close outer for loop	clothes out of follow					4
38	switch food	which phone					2
38	case taco	case taco		2			
38	good equal to true	good equal to throw		3			2
38	case rice	this rice		1			1
38	good equal to true		4				
38	case ketchup	ketchup	1	1			
38	good equal to false	good equal to force		3			1
38	default	the fall					1
38	break	break		1			
38	close switch loop	clothes switch though		1			2
38	while participating greater than skipping	while participating in greater than skipping		5			
38	if job equal to equal to good	if jobs equal to equal to go to		6			1
38	initialize response equal to thank you very much	utilize in response to thank you very much	1	6			1
38	close if loop	clothes is no		1			2
38	compensation equal to compensation plus ten	compensation equal to compensation +10		4	2		
38	close while loop		3				
		% Of Total	10.84337349	51.80722892	9.638554217	3.614457831	24.09638554

APPENDIX H:

SCREENSHOT OF APPLICATION PER PARTICIPANT

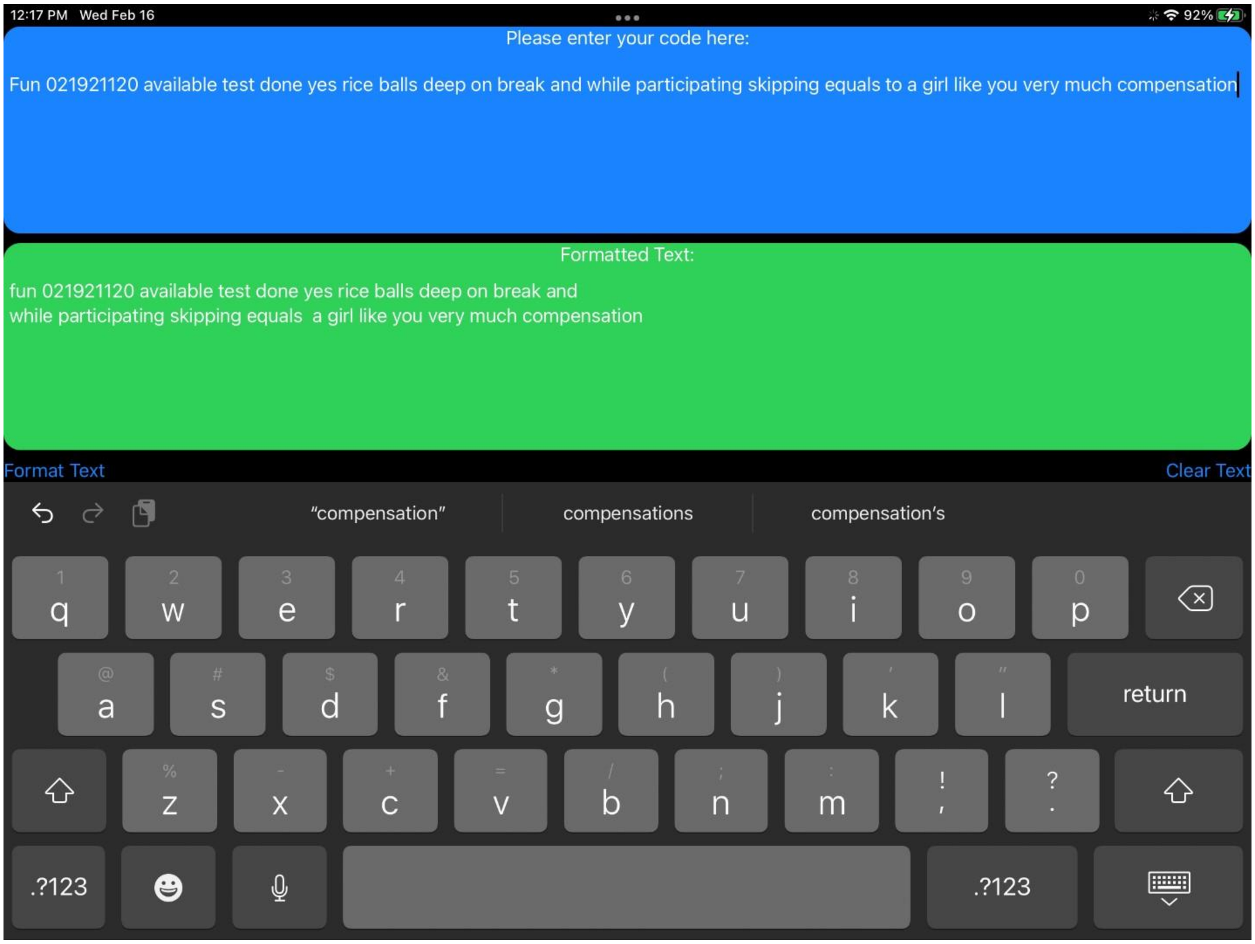
H.1) Participant 1



Zero one open flower bracket print (double: hello space word double quotes) 4X021 open flat bracket where space test is equal to 10+ lower bracket close la bracket switch (food clothes (open fly back in case taco: good is equal to two case price: good as equal to kiss catch up good as equal to default: break close flop racket while participating greater than skipping open flower packet if job is equal to is equal to God open flower bracket compensation plus is equal to 10 clothes flat bracket

```
0 1  flower bracket print (double: hello   word double quotes) 4x021  flat bracket where   test is =  10+ lower bracket  la bracket switch (food
clothes (open fly back in case taco: good is =  2 case price: good as =  kiss catch up good as =  default: break  flop racket
while participating greater than skipping  flower packet
if job is =  is =  god  flower bracket compensation  is =  10 clothes flat bracket
```

H.2) Participant 2



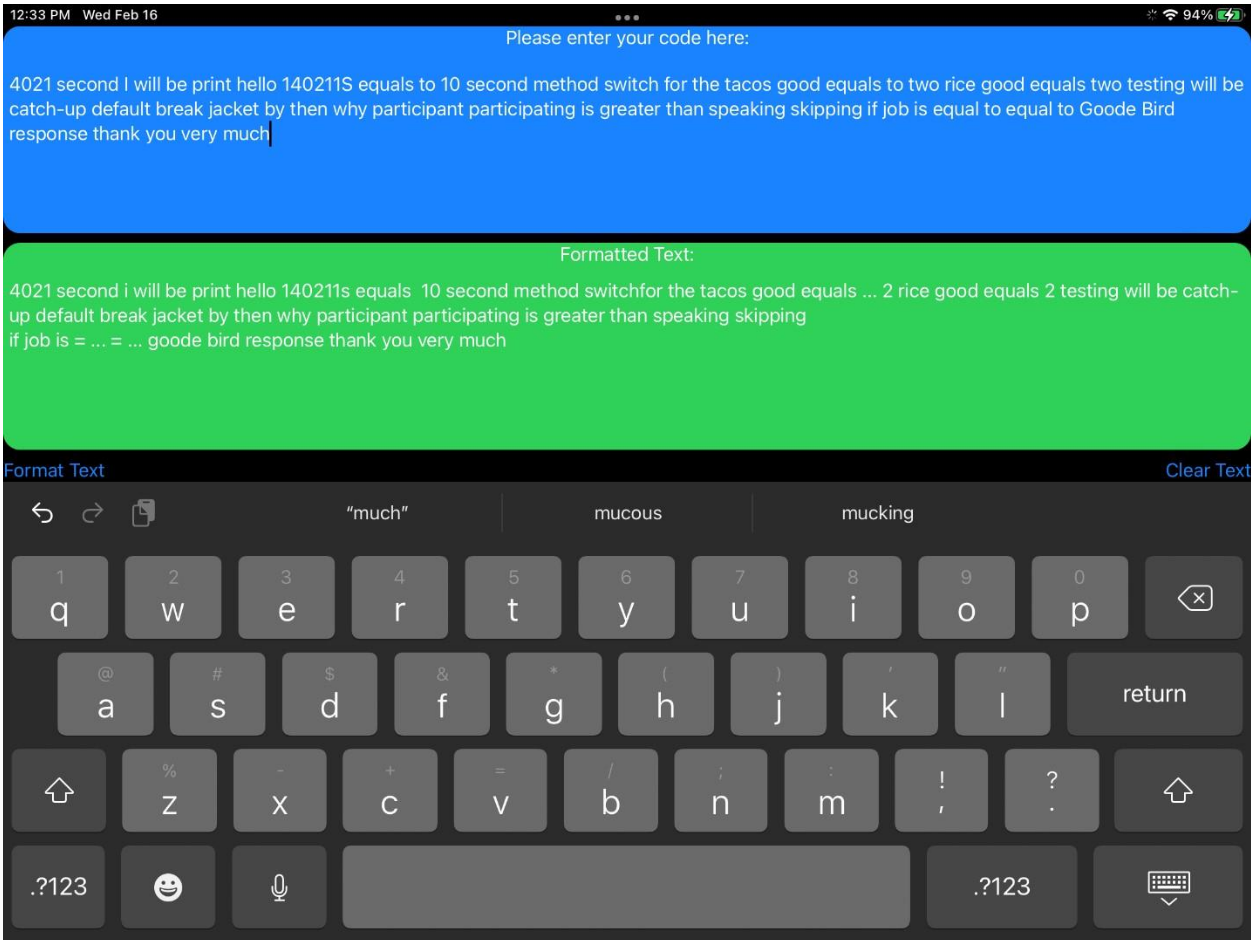
Please enter your code here:

Fun 021921120 available test done yes rice balls deep on break and while participating skipping equals to a girl like you very much compensation

Formatted Text:

fun 021921120 available test done yes rice balls deep on break and
while participating skipping equals a girl like you very much compensation

H.3) Participant 3



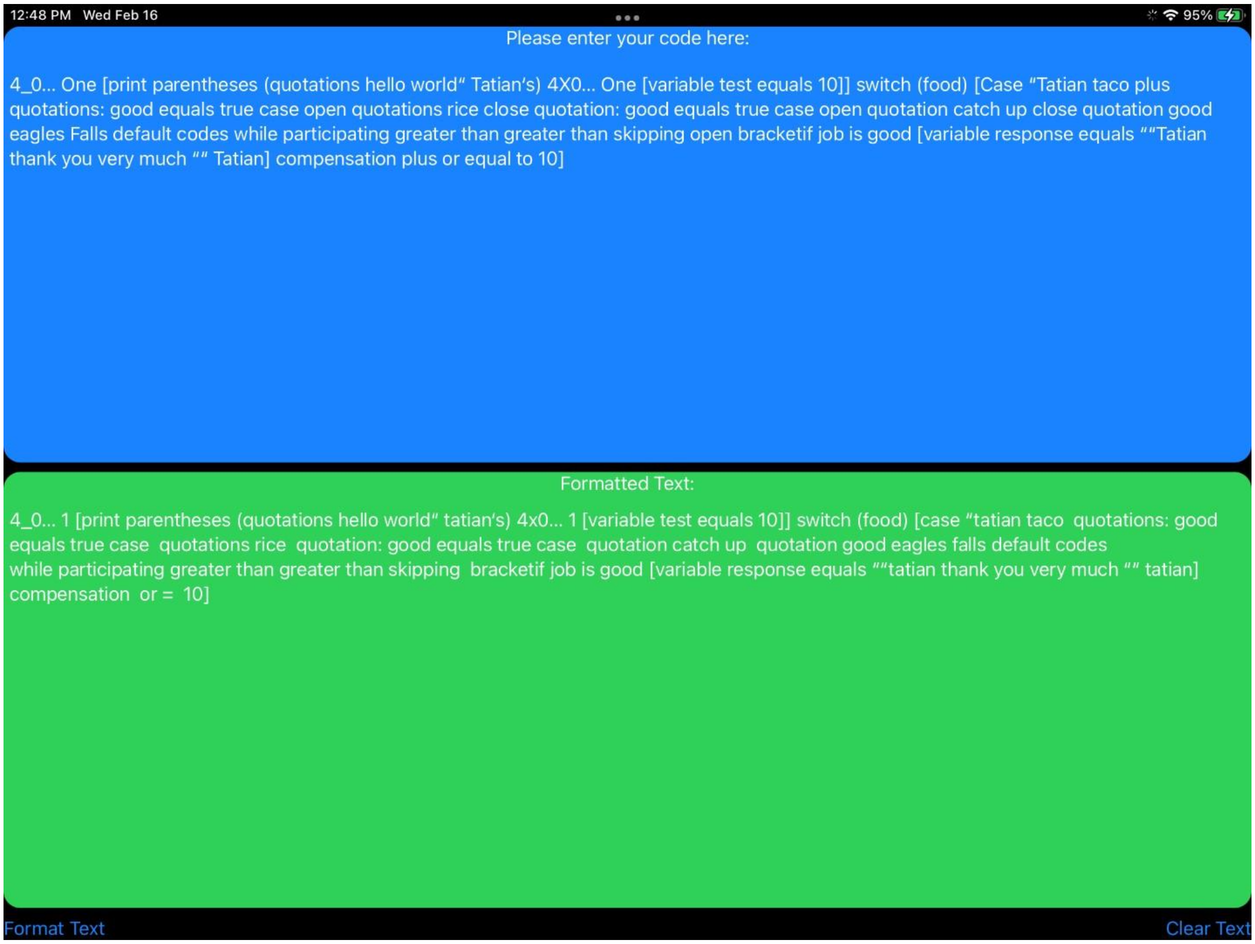
Please enter your code here:

4021 second I will be print hello 140211S equals to 10 second method switch for the tacos good equals to two rice good equals two testing will be catch-up default break jacket by then why participant participating is greater than speaking skipping if job is equal to equal to Goode Bird response thank you very much

Formatted Text:

```
4021 second i will be print hello 140211s equals  10 second method switchfor the tacos good equals ... 2 rice good equals 2 testing will be catch-up default break jacket by then why participant participating is greater than speaking skipping
if job is = ... = ... goode bird response thank you very much
```

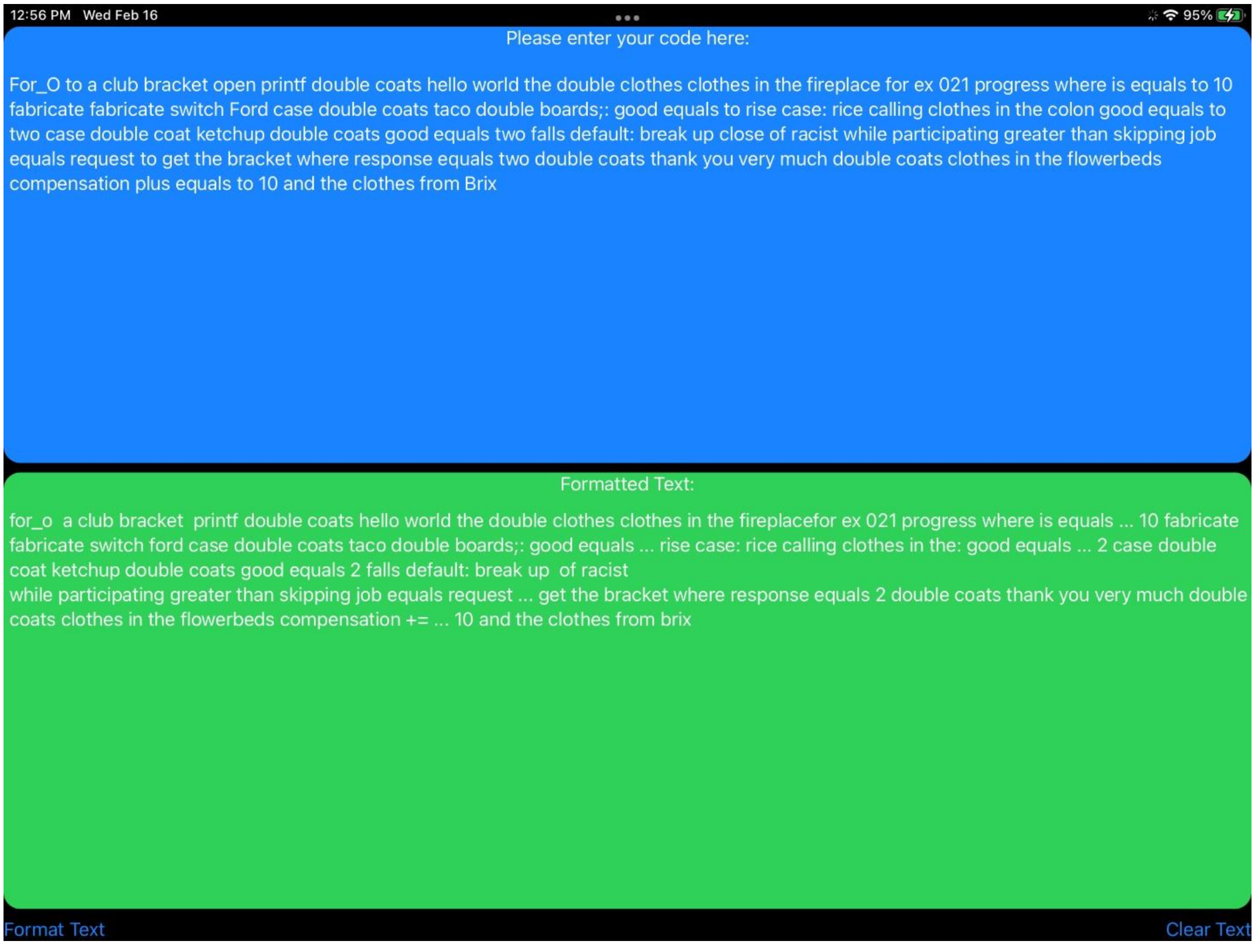
H.4) Participant 5



```
4_0... One [print parentheses (quotations hello world“ Tatian’s) 4X0... One [variable test equals 10]] switch (food) [Case “Tatian taco plus
quotations: good equals true case open quotations rice close quotation: good equals true case open quotation catch up close quotation good
eagles Falls default codes while participating greater than greater than skipping open bracketif job is good [variable response equals ““Tatian
thank you very much ““ Tatian] compensation plus or equal to 10]
```

```
4_0... 1 [print parentheses (quotations hello world“ tatian’s) 4x0... 1 [variable test equals 10]] switch (food) [case “tatian taco  quotations: good
equals true case  quotations rice  quotation: good equals true case  quotation catch up  quotation good eagles falls default codes
while participating greater than greater than skipping  bracketif job is good [variable response equals ““tatian thank you very much ““ tatian]
compensation  or =  10]
```

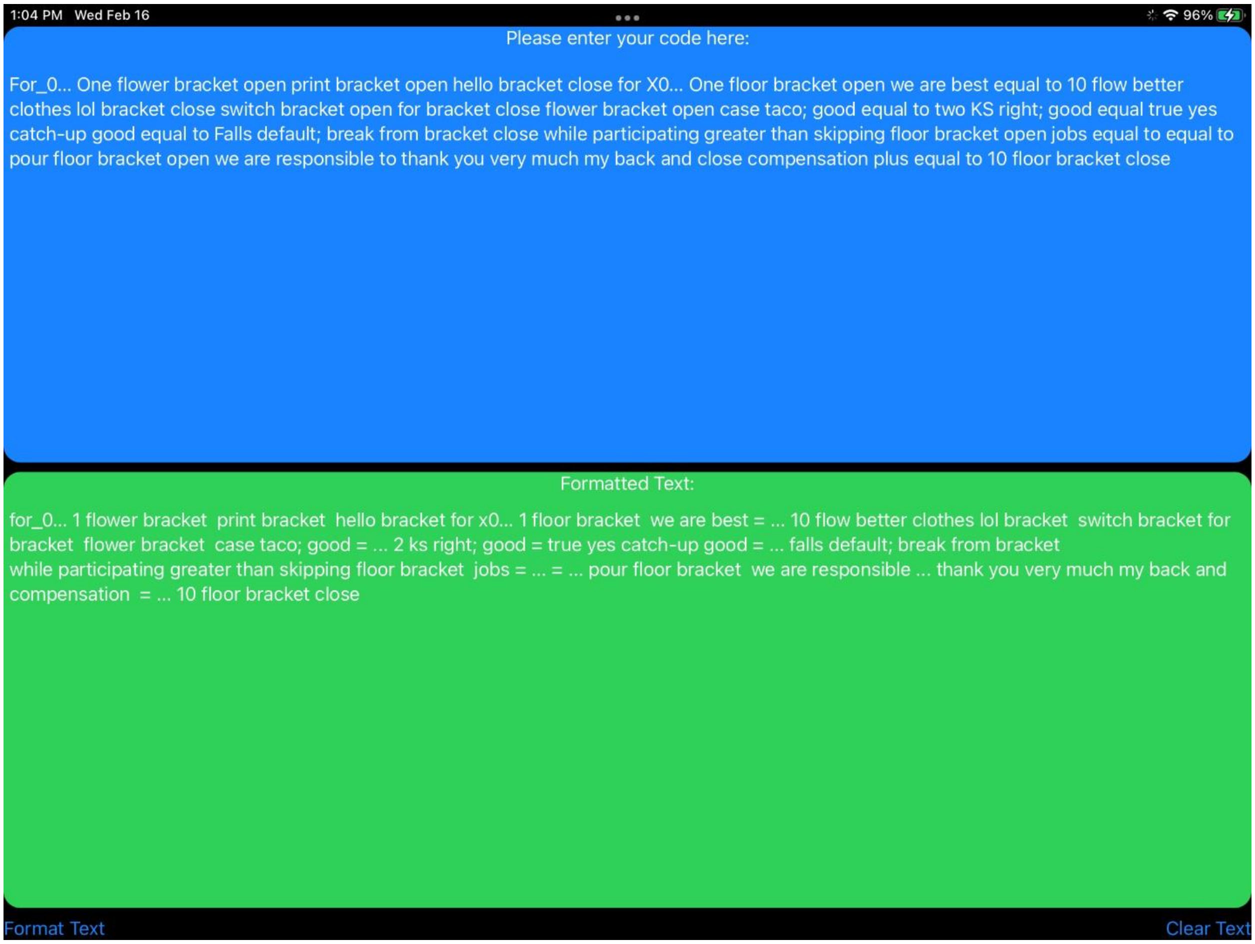
H.5) Participant 6



For_O to a club bracket open printf double coats hello world the double clothes clothes in the fireplace for ex 021 progress where is equals to 10 fabricate fabricate switch Ford case double coats taco double boards;: good equals to rise case: rice calling clothes in the colon good equals to two case double coat ketchup double coats good equals two falls default: break up close of racist while participating greater than skipping job equals request to get the bracket where response equals two double coats thank you very much double coats clothes in the flowerbeds compensation plus equals to 10 and the clothes from Brix

```
for_o  a club bracket  printf double coats hello world the double clothes clothes in the fireplacefor ex 021 progress where is equals ... 10 fabricate fabricate switch ford case double coats taco double boards;: good equals ... rise case: rice calling clothes in the: good equals ... 2 case double coat ketchup double coats good equals 2 falls default: break up  of racist
while participating greater than skipping job equals request ... get the bracket where response equals 2 double coats thank you very much double coats clothes in the flowerbeds compensation += ... 10 and the clothes from brix
```

H.6) Participant 7



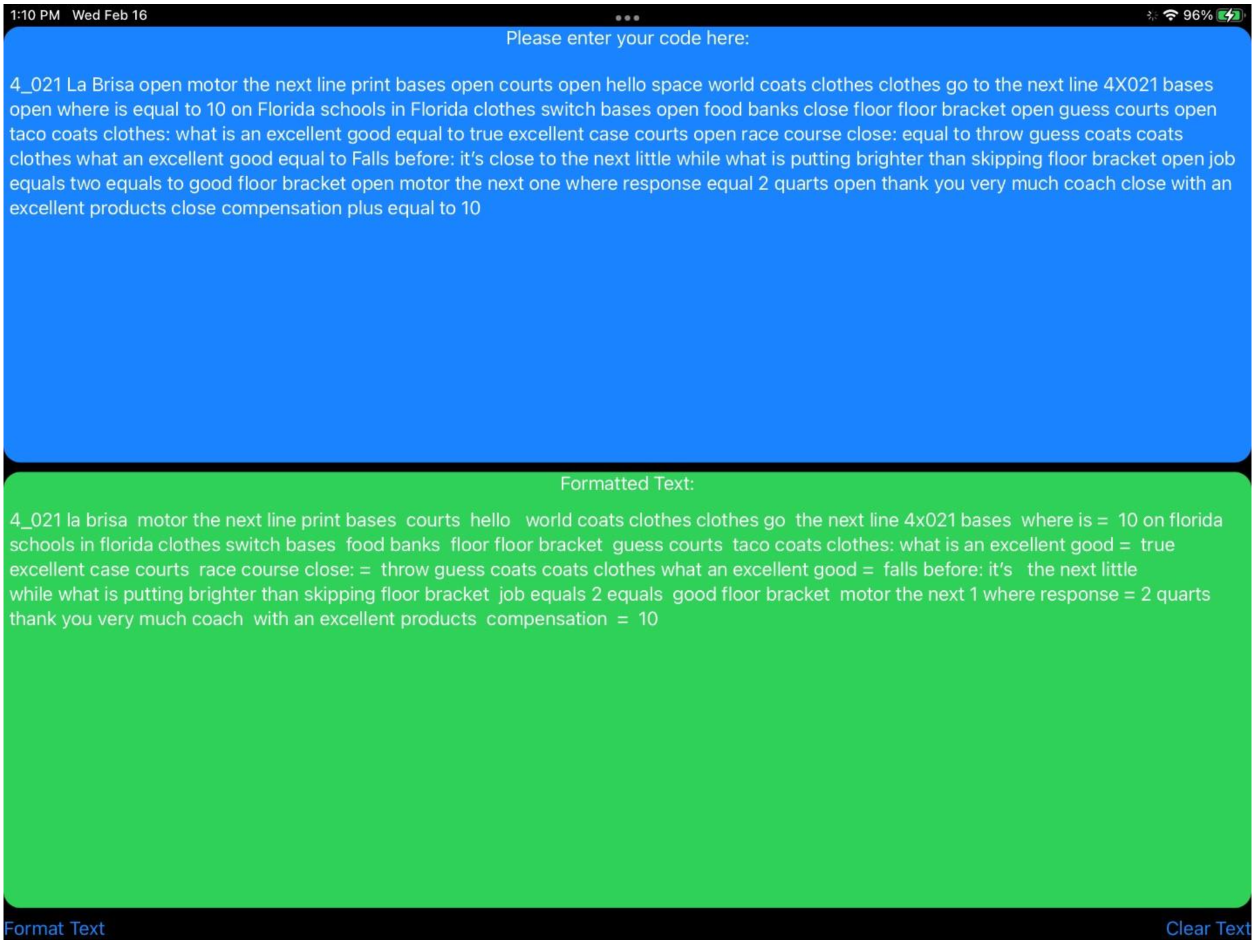
Please enter your code here:

For_0... One flower bracket open print bracket open hello bracket close for X0... One floor bracket open we are best equal to 10 flow better clothes lol bracket close switch bracket open for bracket close flower bracket open case taco; good equal to two KS right; good equal true yes catch-up good equal to Falls default; break from bracket close while participating greater than skipping floor bracket open jobs equal to equal to pour floor bracket open we are responsible to thank you very much my back and close compensation plus equal to 10 floor bracket close

Formatted Text:

```
for_0... 1 flower bracket  print bracket  hello bracket for x0... 1 floor bracket  we are best = ... 10 flow better clothes lol bracket  switch bracket for bracket  flower bracket  case taco; good = ... 2 ks right; good = true yes catch-up good = ... falls default; break from bracket
while participating greater than skipping floor bracket  jobs = ... = ... pour floor bracket  we are responsible ... thank you very much my back and compensation  = ... 10 floor bracket close
```

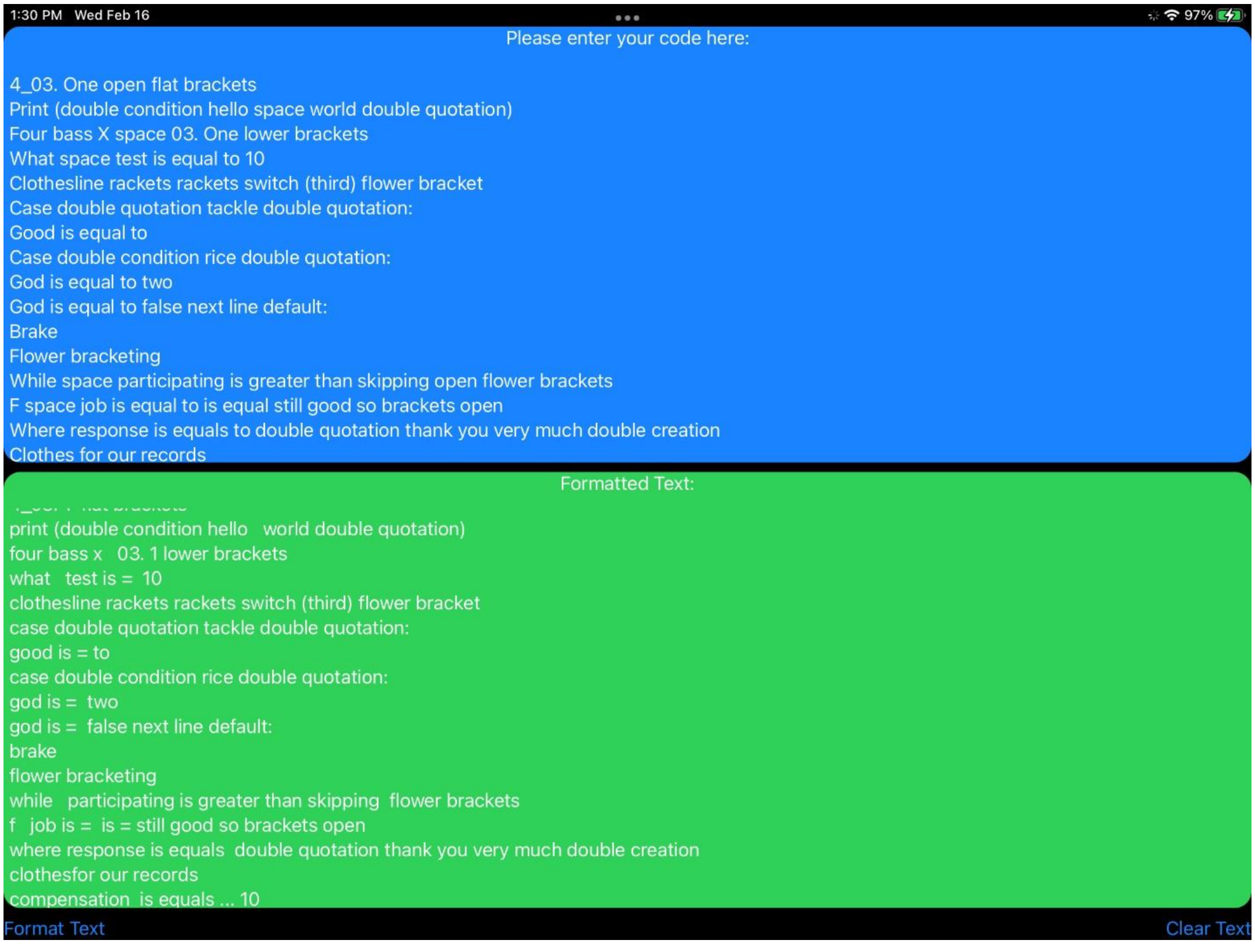
H.7) Participant 8



4_021 La Brisa open motor the next line print bases open courts open hello space world coats clothes clothes go to the next line 4X021 bases open where is equal to 10 on Florida schools in Florida clothes switch bases open food banks close floor floor bracket open guess courts open taco coats clothes: what is an excellent good equal to true excellent case courts open race course close: equal to throw guess coats coats clothes what an excellent good equal to Falls before: it's close to the next little while what is putting brighter than skipping floor bracket open job equals two equals to good floor bracket open motor the next one where response equal 2 quarts open thank you very much coach close with an excellent products close compensation plus equal to 10

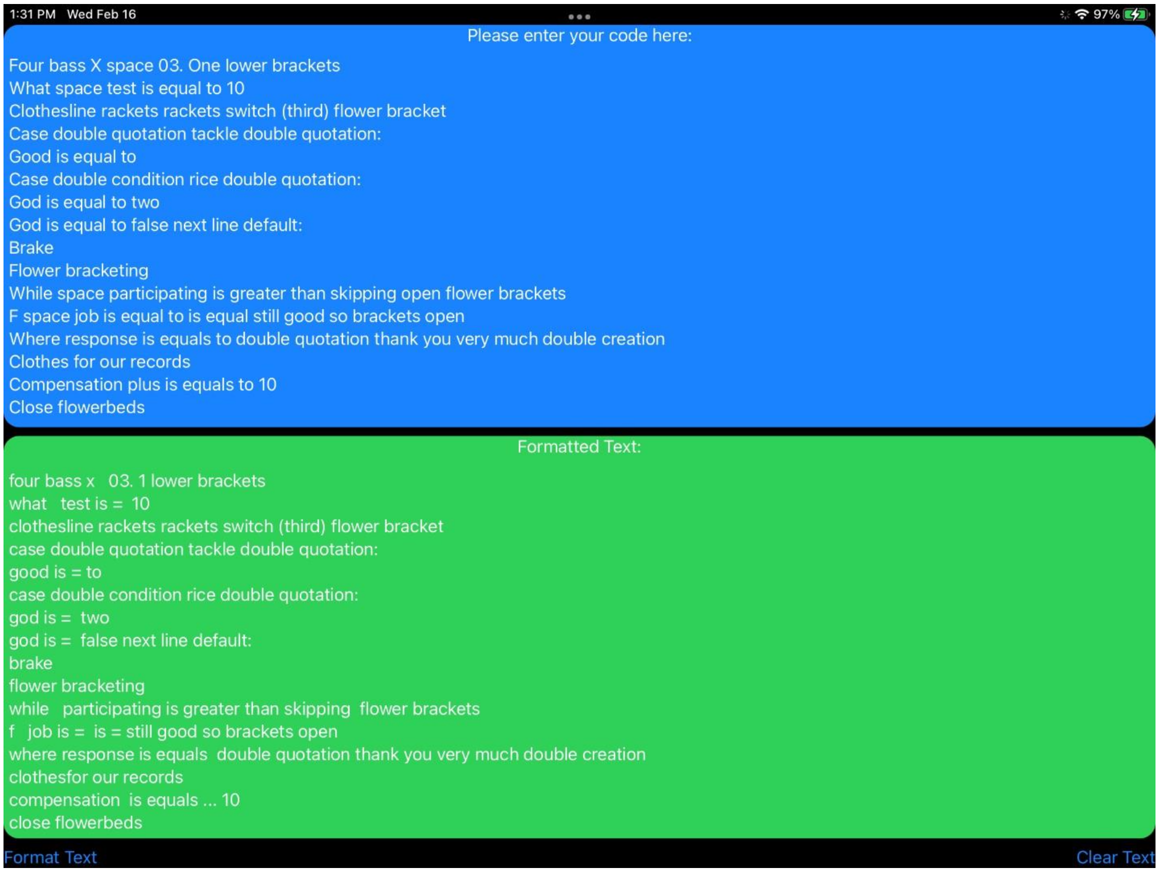
```
4_021 la brisa  motor the next line print bases  courts  hello   world coats clothes clothes go  the next line 4x021 bases  where is =  10 on florida
schools in florida clothes switch bases  food banks  floor floor bracket  guess courts  taco coats clothes: what is an excellent good =  true
excellent case courts  race course close: =  throw guess coats coats clothes what an excellent good =  falls before: it's   the next little
while what is putting brighter than skipping floor bracket  job equals 2 equals  good floor bracket  motor the next 1 where response = 2 quarts
thank you very much coach  with an excellent products  compensation  =  10
```

H.8) Participant 9



```
4_03. One open flat brackets
Print (double condition hello space world double quotation)
Four bass X space 03. One lower brackets
What space test is equal to 10
Clothesline rackets rackets switch (third) flower bracket
Case double quotation tackle double quotation:
Good is equal to
Case double condition rice double quotation:
God is equal to two
God is equal to false next line default:
Brake
Flower bracketing
While space participating is greater than skipping open flower brackets
F space job is equal to is equal still good so brackets open
Where response is equals to double quotation thank you very much double creation
Clothes for our records
```

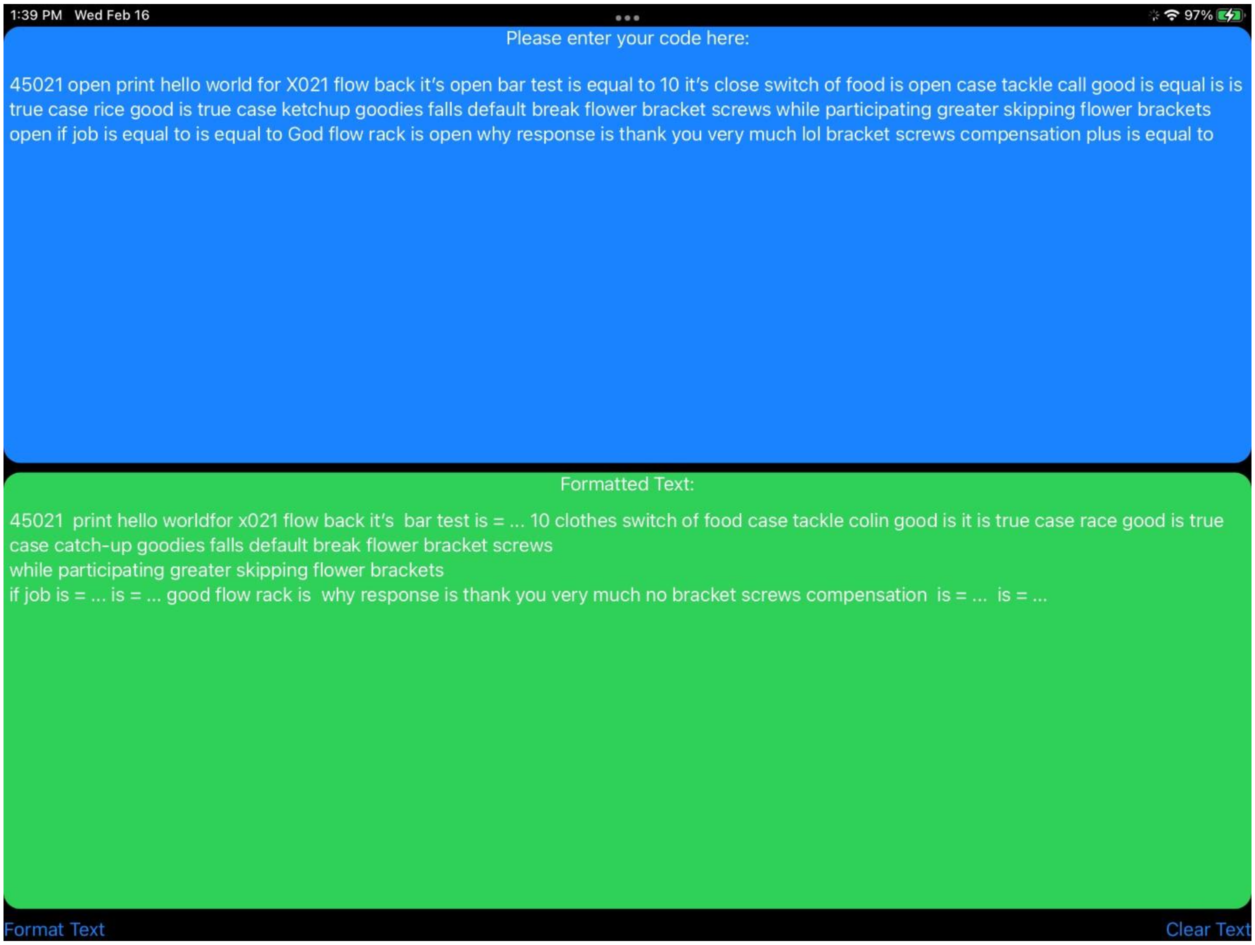
```
4_03. 1 flat brackets
print (double condition hello   world double quotation)
four bass x   03. 1 lower brackets
what   test is =  10
clothesline rackets rackets switch (third) flower bracket
case double quotation tackle double quotation:
good is = to
case double condition rice double quotation:
god is =  two
god is =  false next line default:
brake
flower bracketing
while   participating is greater than skipping  flower brackets
f   job is =  is = still good so brackets open
where response is equals  double quotation thank you very much double creation
clothesfor our records
compensation  is equals ... 10
```



```
Four bass X space 03. One lower brackets
What space test is equal to 10
Clothesline rackets rackets switch (third) flower bracket
Case double quotation tackle double quotation:
Good is equal to
Case double condition rice double quotation:
God is equal to two
God is equal to false next line default:
Brake
Flower bracketing
While space participating is greater than skipping open flower brackets
F space job is equal to is equal still good so brackets open
Where response is equals to double quotation thank you very much double creation
Clothes for our records
Compensation plus is equals to 10
Close flowerbeds
```

```
four bass x   03. 1 lower brackets
what   test is =  10
clothesline rackets rackets switch (third) flower bracket
case double quotation tackle double quotation:
good is = to
case double condition rice double quotation:
god is =  two
god is =  false next line default:
brake
flower bracketing
while   participating is greater than skipping  flower brackets
f   job is =  is = still good so brackets open
where response is equals  double quotation thank you very much double creation
clothesfor our records
compensation  is equals ... 10
close flowerbeds
```

H.9) Participant 10



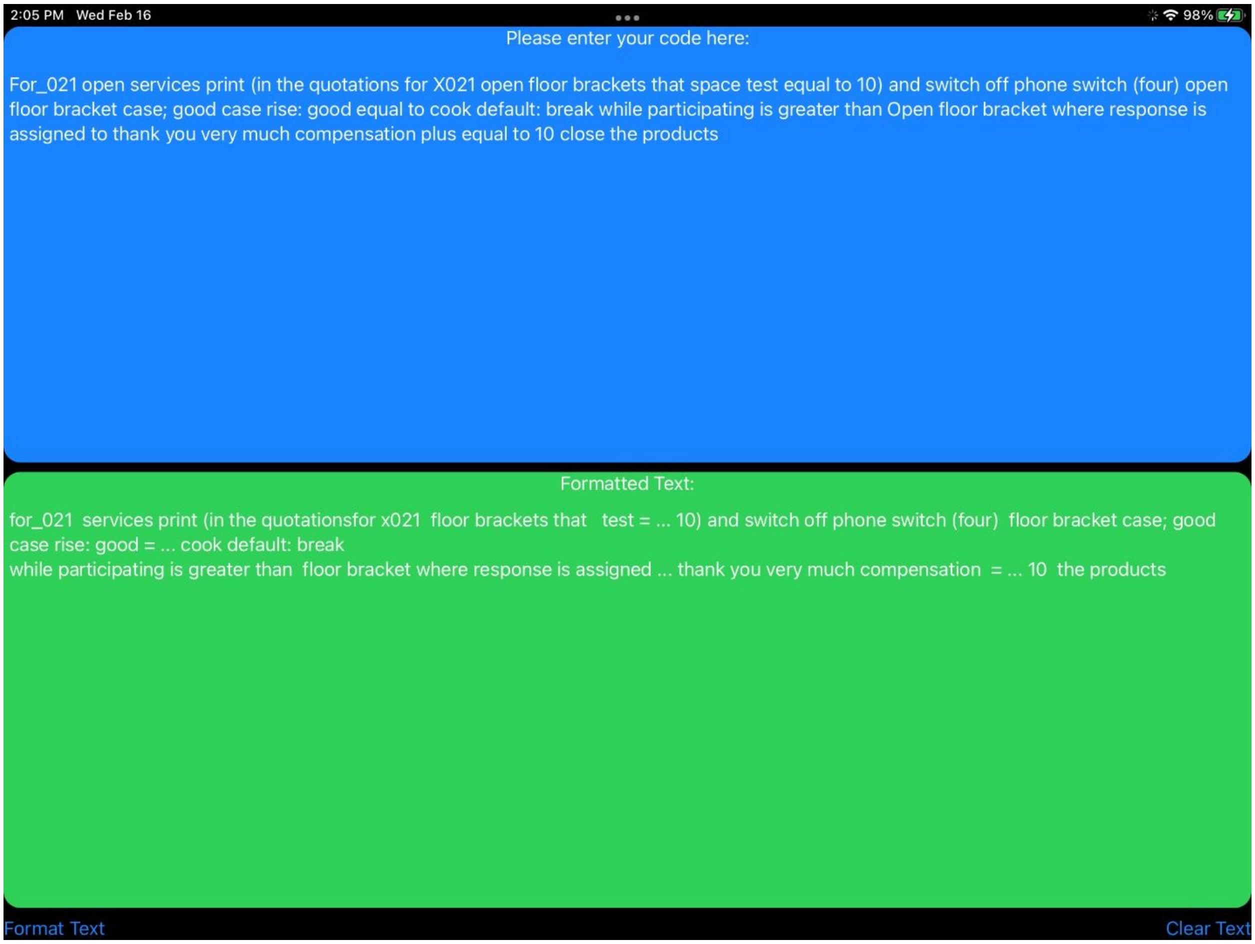
Please enter your code here:

```
45021 open print hello world for X021 flow back it's open bar test is equal to 10 it's close switch of food is open case tackle call good is equal is is true case rice good is true case ketchup goodies falls default break flower bracket screws while participating greater skipping flower brackets open if job is equal to is equal to God flow rack is open why response is thank you very much lol bracket screws compensation plus is equal to
```

Formatted Text:

```
45021  print hello worldfor x021 flow back it's  bar test is = ... 10 clothes switch of food case tackle colin good is it is true case race good is true case catch-up goodies falls default break flower bracket screws
while participating greater skipping flower brackets
if job is = ... is = ... good flow rack is  why response is thank you very much no bracket screws compensation  is = ...  is = ...
```

H.10) Participant 11



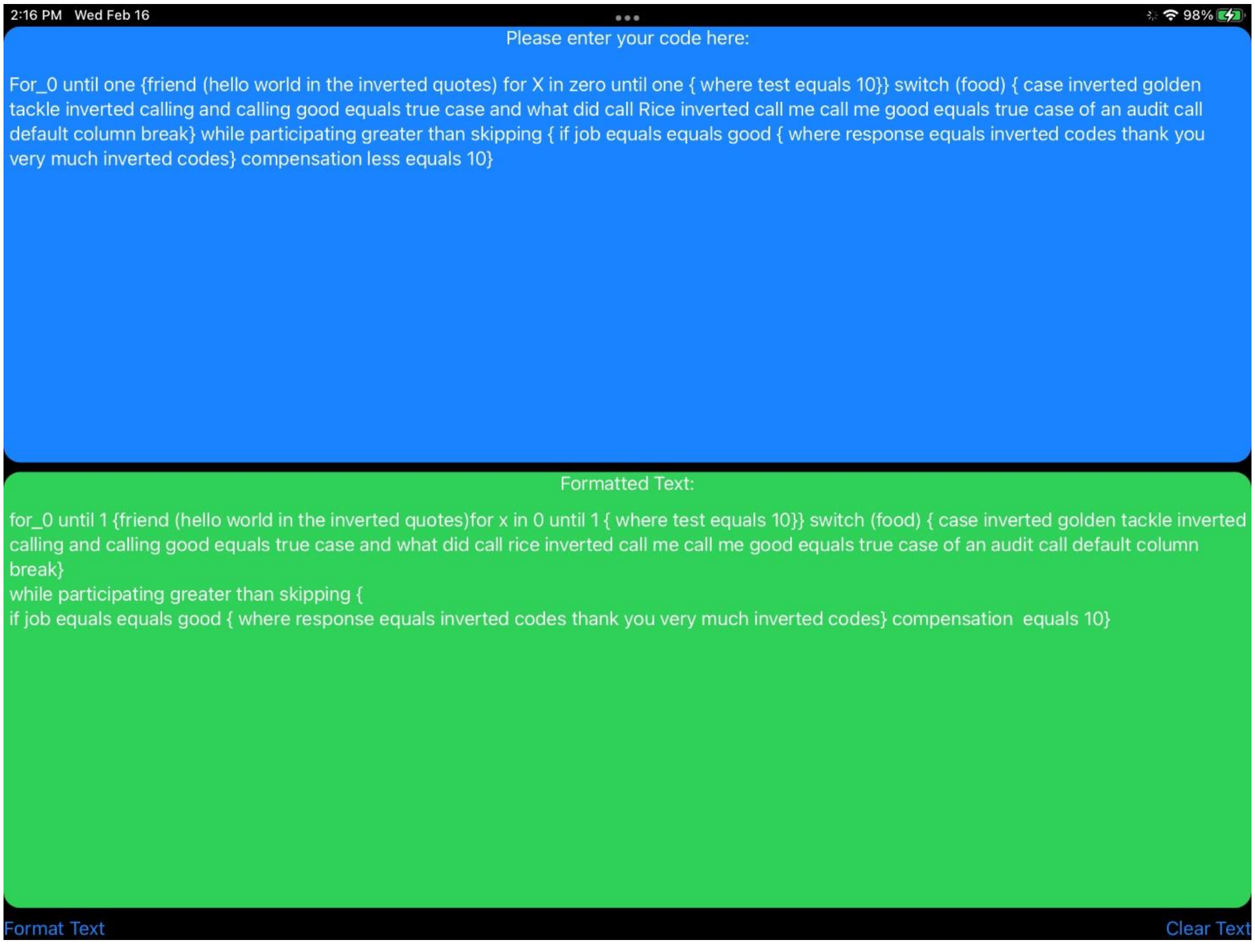
Please enter your code here:

For_021 open services print (in the quotations for X021 open floor brackets that space test equal to 10) and switch off phone switch (four) open floor bracket case; good case rise: good equal to cook default: break while participating is greater than Open floor bracket where response is assigned to thank you very much compensation plus equal to 10 close the products

Formatted Text:

```
for_021  services print (in the quotationsfor x021  floor brackets that   test = ... 10) and switch off phone switch (four)  floor bracket case; good
case rise: good = ... cook default: break
while participating is greater than  floor bracket where response is assigned ... thank you very much compensation  = ... 10  the products
```

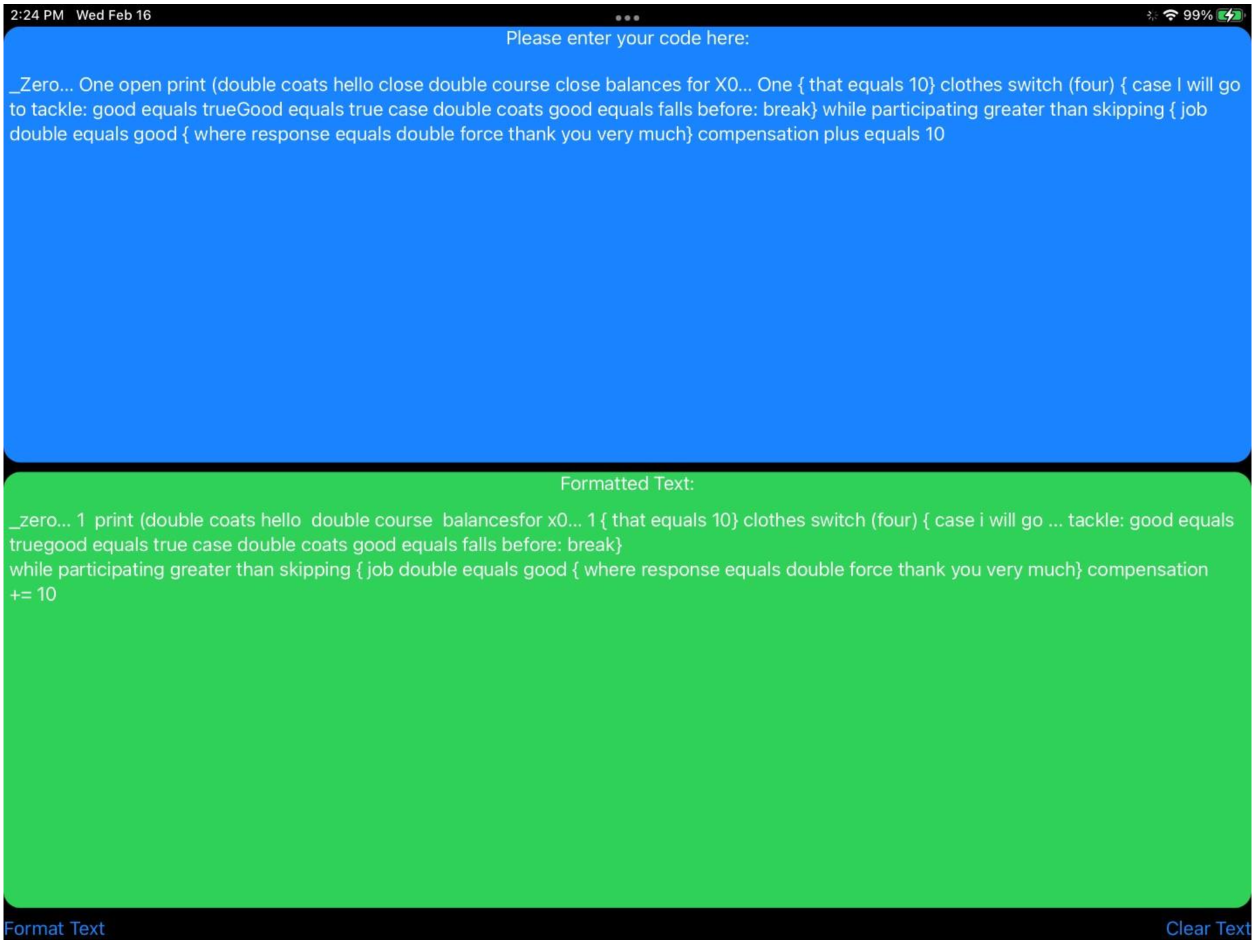
H.11) Participant 12



For_0 until one {friend (hello world in the inverted quotes) for X in zero until one { where test equals 10}} switch (food) { case inverted golden tackle inverted calling and calling good equals true case and what did call Rice inverted call me call me good equals true case of an audit call default column break} while participating greater than skipping { if job equals equals good { where response equals inverted codes thank you very much inverted codes} compensation less equals 10}

for_0 until 1 {friend (hello world in the inverted quotes)for x in 0 until 1 { where test equals 10}} switch (food) { case inverted golden tackle inverted calling and calling good equals true case and what did call rice inverted call me call me good equals true case of an audit call default column break}
while participating greater than skipping {
if job equals equals good { where response equals inverted codes thank you very much inverted codes} compensation equals 10}

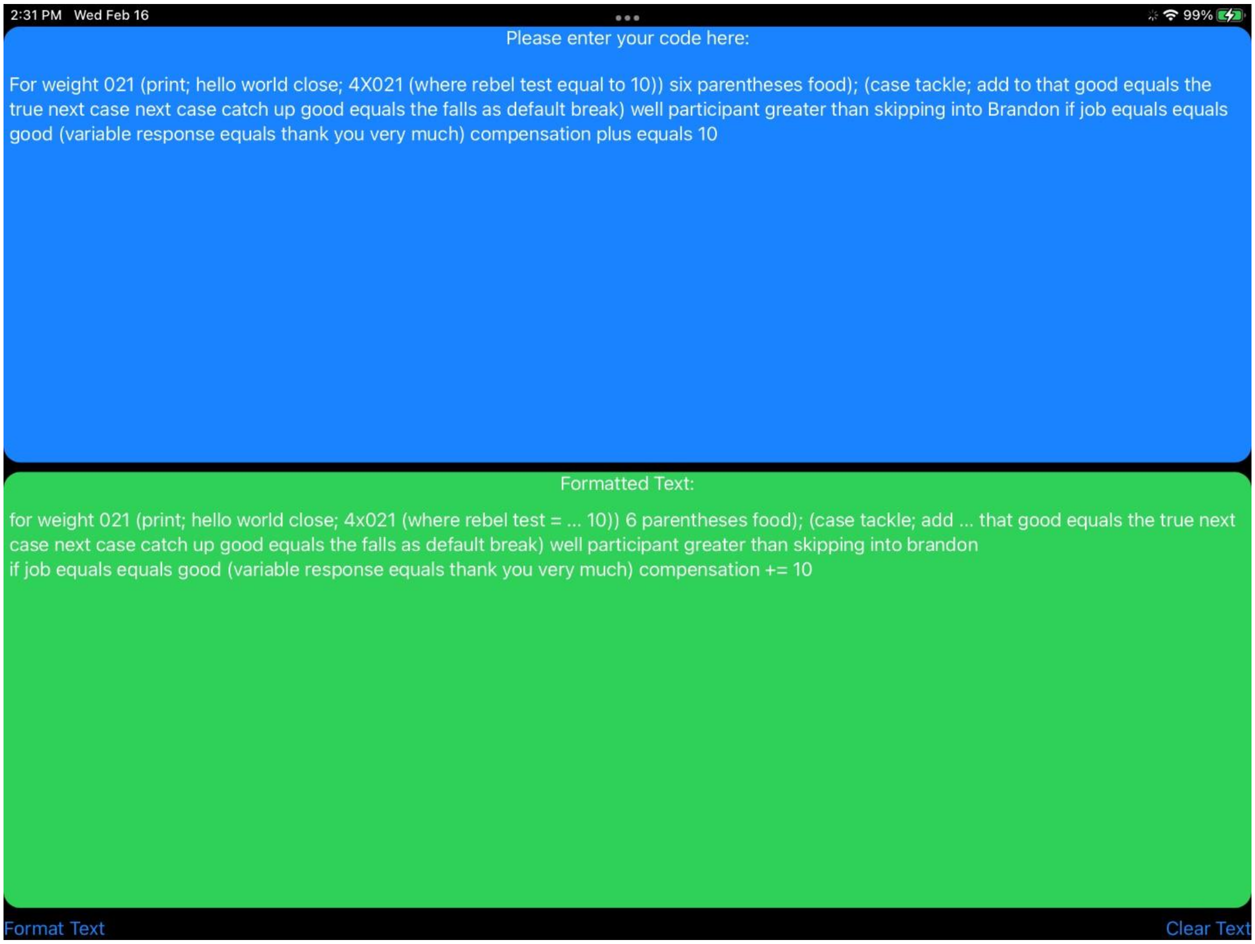
H.12) Participant 13



```
_Zero... One open print (double coats hello close double course close balances for X0... One { that equals 10} clothes switch (four) { case I will go to tackle: good equals trueGood equals true case double coats good equals falls before: break} while participating greater than skipping { job double equals good { where response equals double force thank you very much} compensation plus equals 10
```

```
_zero... 1  print (double coats hello  double course  balancesfor x0... 1 { that equals 10} clothes switch (four) { case i will go ... tackle: good equals truegood equals true case double coats good equals falls before: break}
while participating greater than skipping { job double equals good { where response equals double force thank you very much} compensation += 10
```

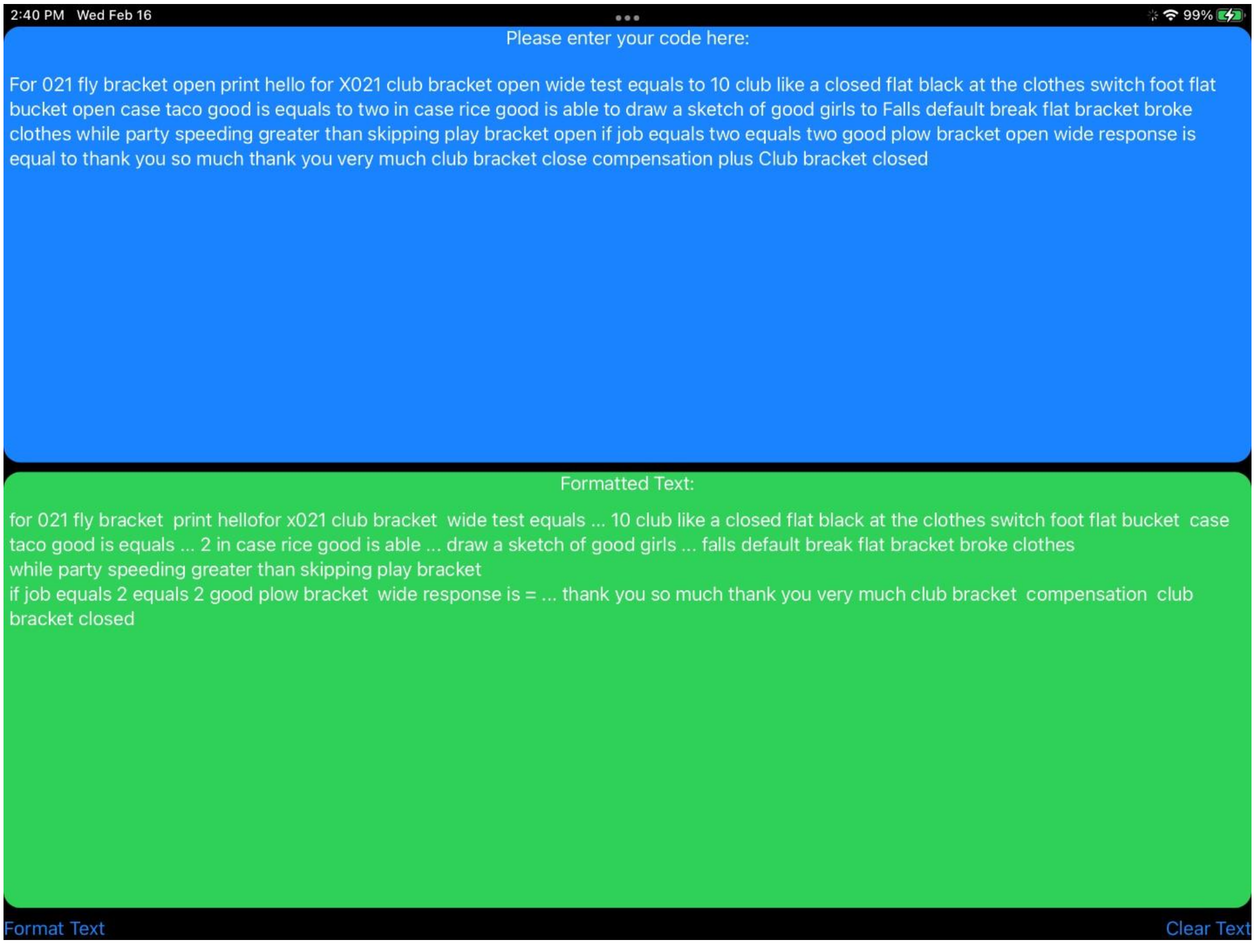
H.13) Participant 14



For weight 021 (print; hello world close; 4X021 (where rebel test equal to 10)) six parentheses food); (case tackle; add to that good equals the true next case next case catch up good equals the falls as default break) well participant greater than skipping into Brandon if job equals equals good (variable response equals thank you very much) compensation plus equals 10

```
for weight 021 (print; hello world close; 4x021 (where rebel test = ... 10)) 6 parentheses food); (case tackle; add ... that good equals the true next case next case catch up good equals the falls as default break) well participant greater than skipping into brandon
if job equals equals good (variable response equals thank you very much) compensation += 10
```

H.14) Participant 15



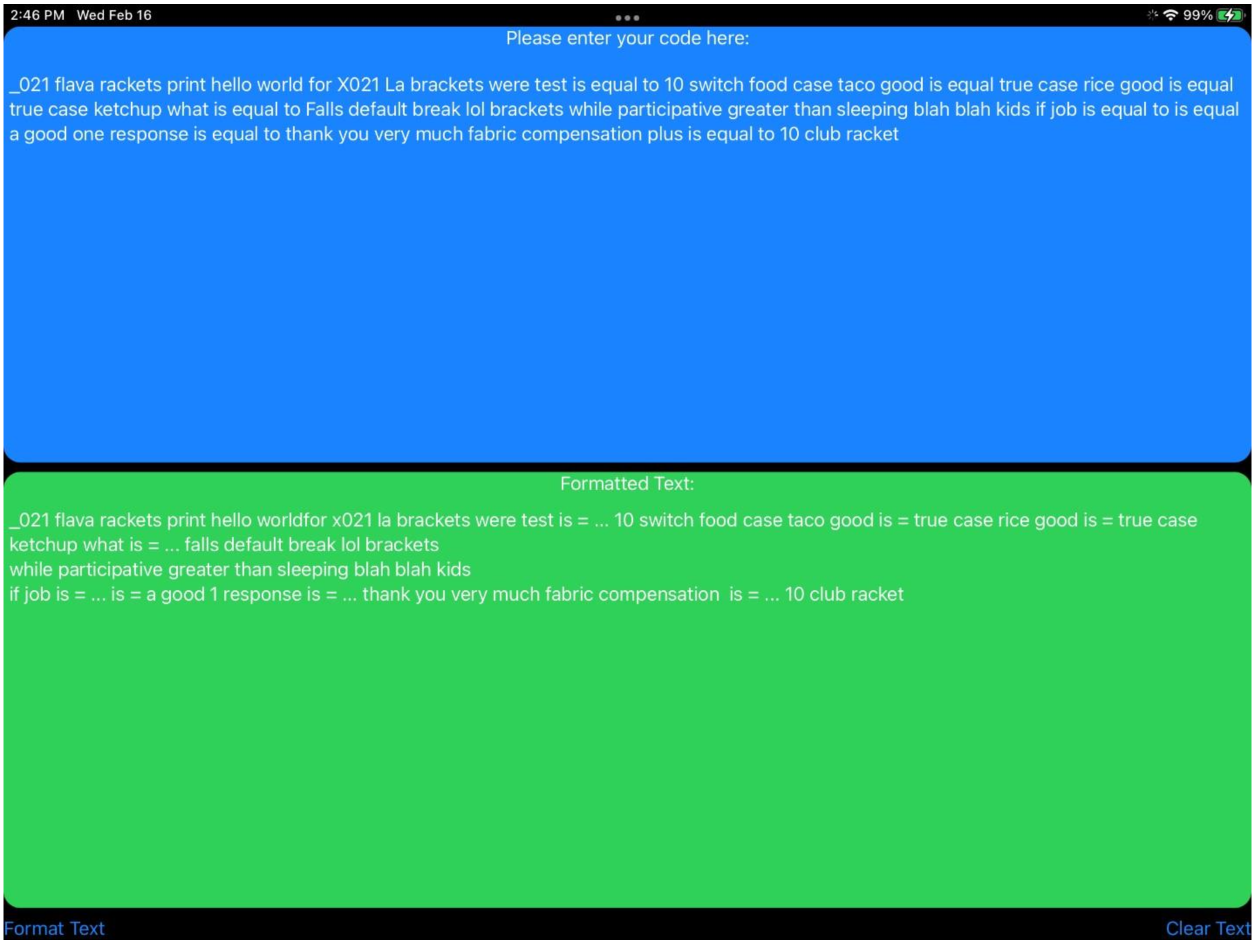
Please enter your code here:

For 021 fly bracket open print hello for X021 club bracket open wide test equals to 10 club like a closed flat black at the clothes switch foot flat bucket open case taco good is equals to two in case rice good is able to draw a sketch of good girls to Falls default break flat bracket broke clothes while party speeding greater than skipping play bracket open if job equals two equals two good plow bracket open wide response is equal to thank you so much thank you very much club bracket close compensation plus Club bracket closed

Formatted Text:

```
for 021 fly bracket  print hellofor x021 club bracket  wide test equals ... 10 club like a closed flat black at the clothes switch foot flat bucket  case taco good is equals ... 2 in case rice good is able ... draw a sketch of good girls ... falls default break flat bracket broke clothes
while party speeding greater than skipping play bracket
if job equals 2 equals 2 good plow bracket  wide response is = ... thank you so much thank you very much club bracket  compensation  club bracket closed
```

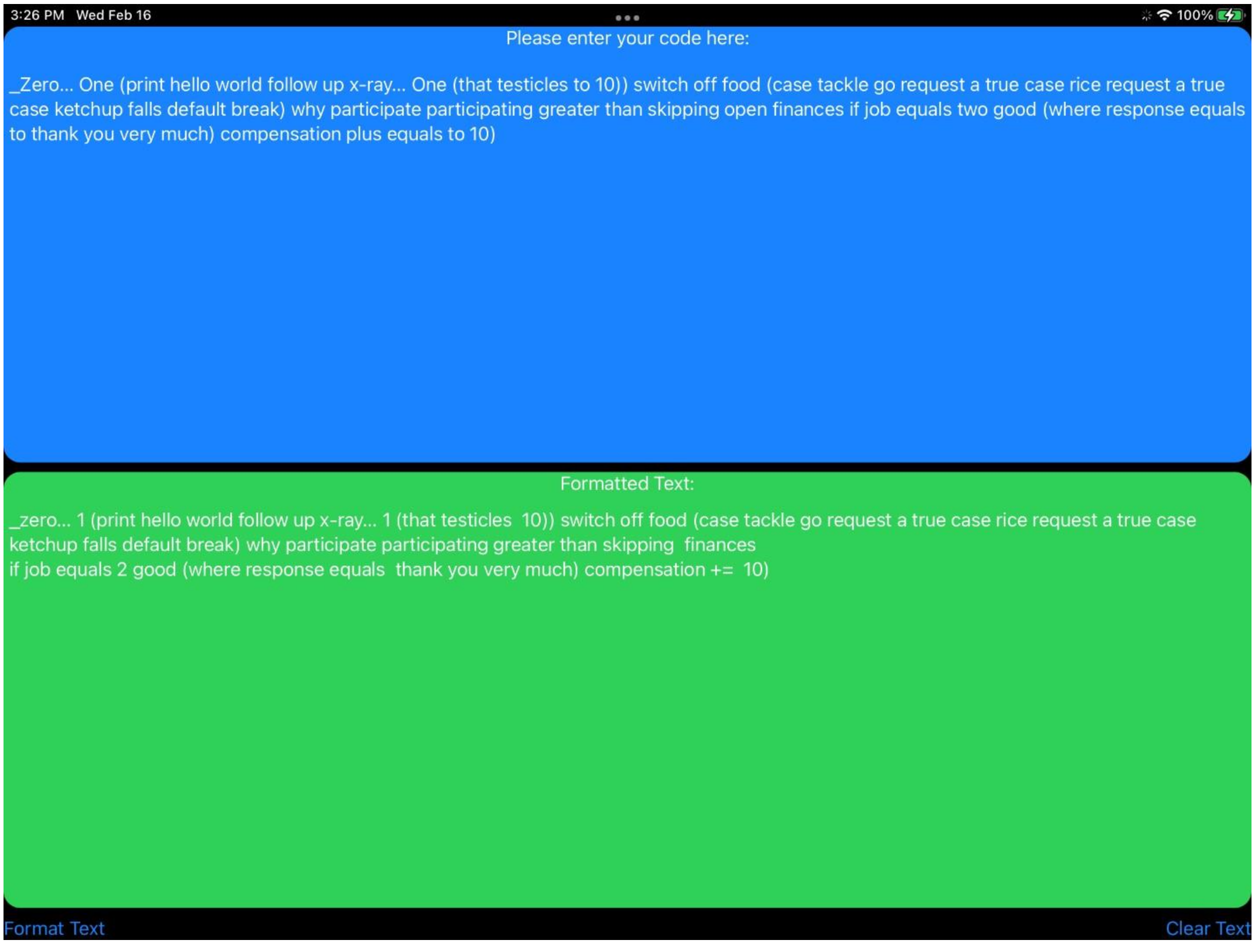
H.15) Participant 16



_021 flava rackets print hello world for X021 La brackets were test is equal to 10 switch food case taco good is equal true case rice good is equal true case ketchup what is equal to Falls default break lol brackets while participative greater than sleeping blah blah kids if job is equal to is equal a good one response is equal to thank you very much fabric compensation plus is equal to 10 club racket

```
_021 flava rackets print hello worldfor x021 la brackets were test is = ... 10 switch food case taco good is = true case rice good is = true case ketchup what is = ... falls default break lol brackets
while participative greater than sleeping blah blah kids
if job is = ... is = a good 1 response is = ... thank you very much fabric compensation  is = ... 10 club racket
```

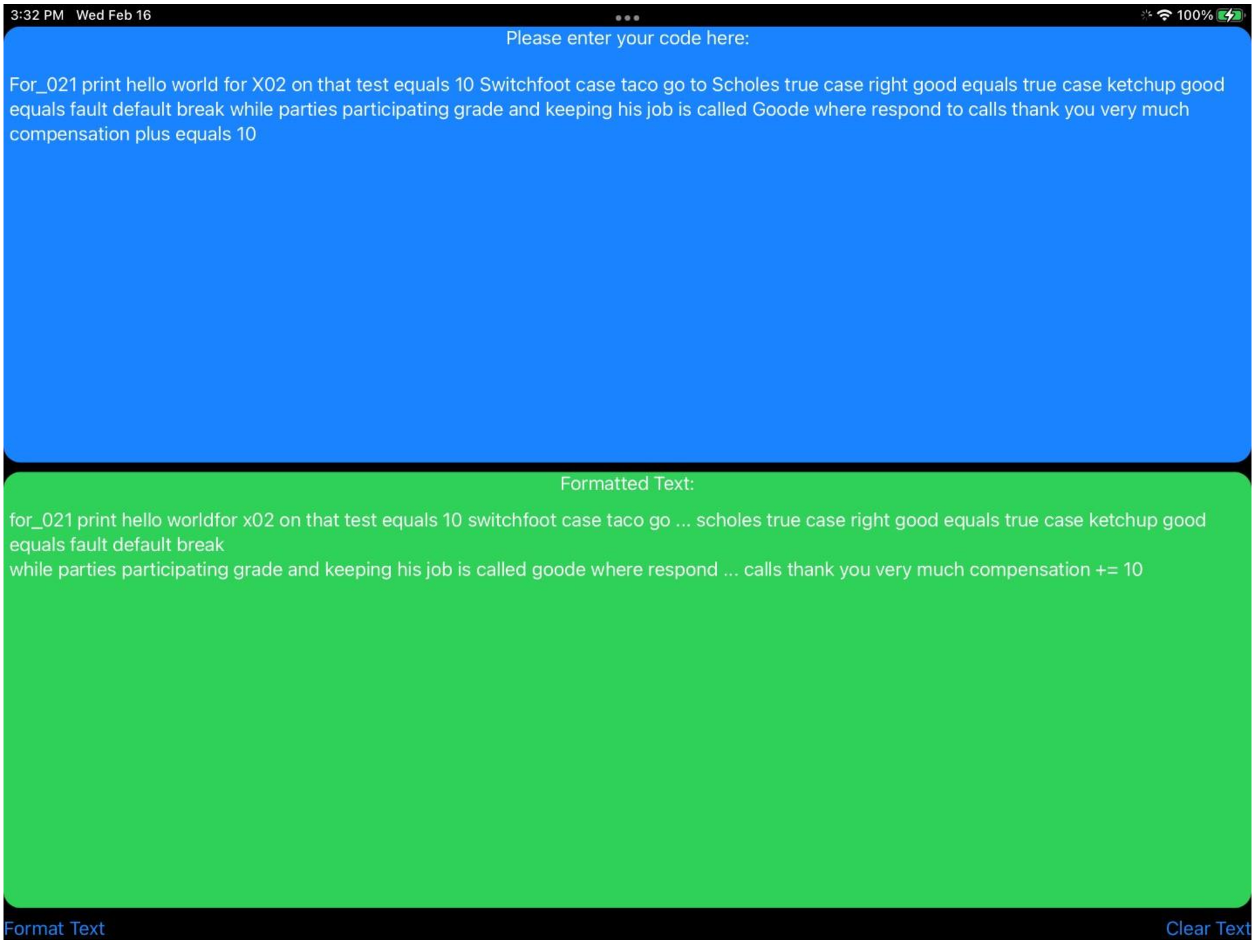
H.16) Participant 18



```
_Zero... One (print hello world follow up x-ray... One (that testicles to 10)) switch off food (case tackle go request a true case rice request a true case ketchup falls default break) why participate participating greater than skipping open finances if job equals two good (where response equals to thank you very much) compensation plus equals to 10)
```

```
_zero... 1 (print hello world follow up x-ray... 1 (that testicles  10)) switch off food (case tackle go request a true case rice request a true case ketchup falls default break) why participate participating greater than skipping  finances
if job equals 2 good (where response equals  thank you very much) compensation +=  10)
```

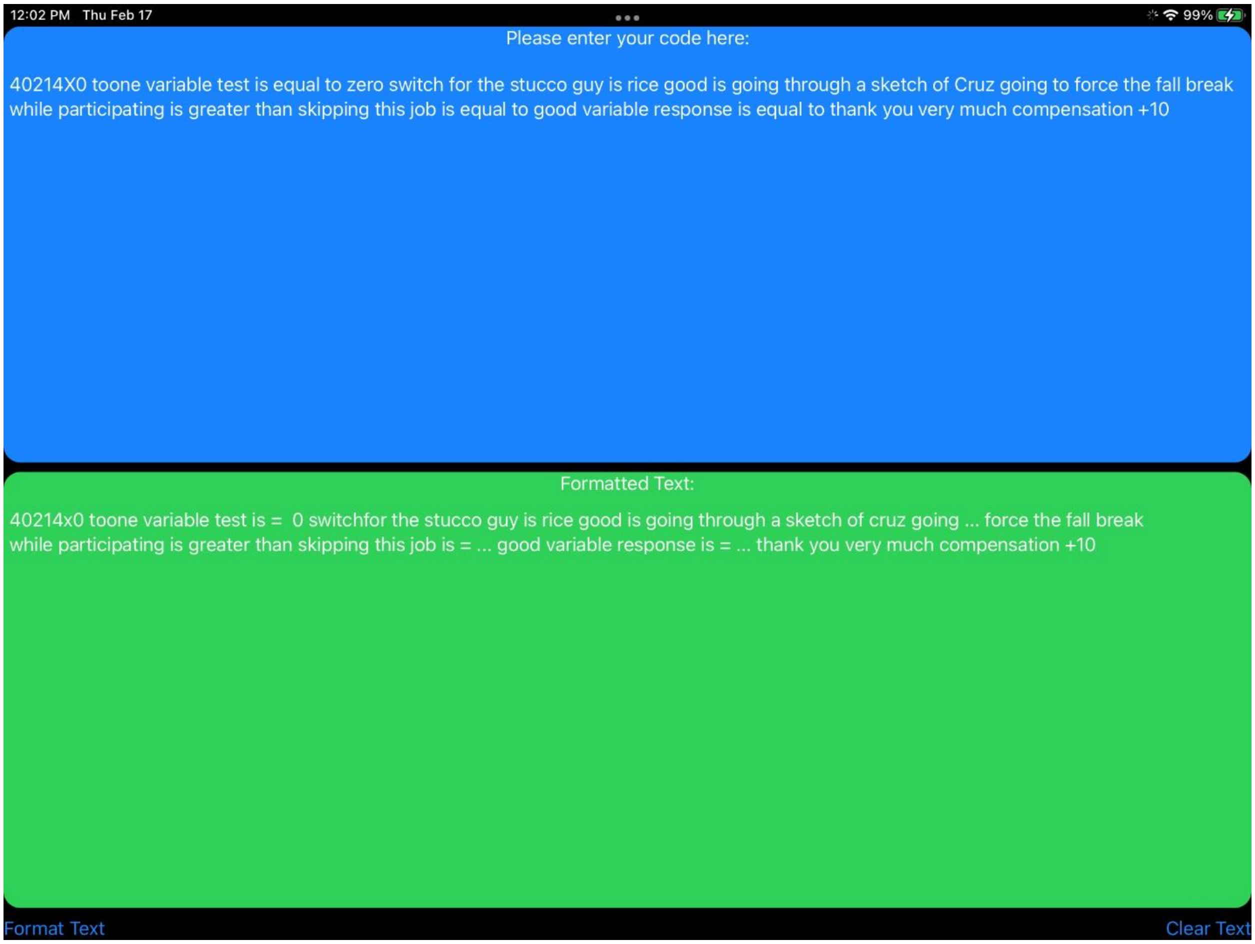
H.17) Participant 19



For_021 print hello world for X02 on that test equals 10 Switchfoot case taco go to Scholes true case right good equals true case ketchup good equals fault default break while parties participating grade and keeping his job is called Goode where respond to calls thank you very much compensation plus equals 10

```
for_021 print hello worldfor x02 on that test equals 10 switchfoot case taco go ... scholes true case right good equals true case ketchup good equals fault default break
while parties participating grade and keeping his job is called goode where respond ... calls thank you very much compensation += 10
```

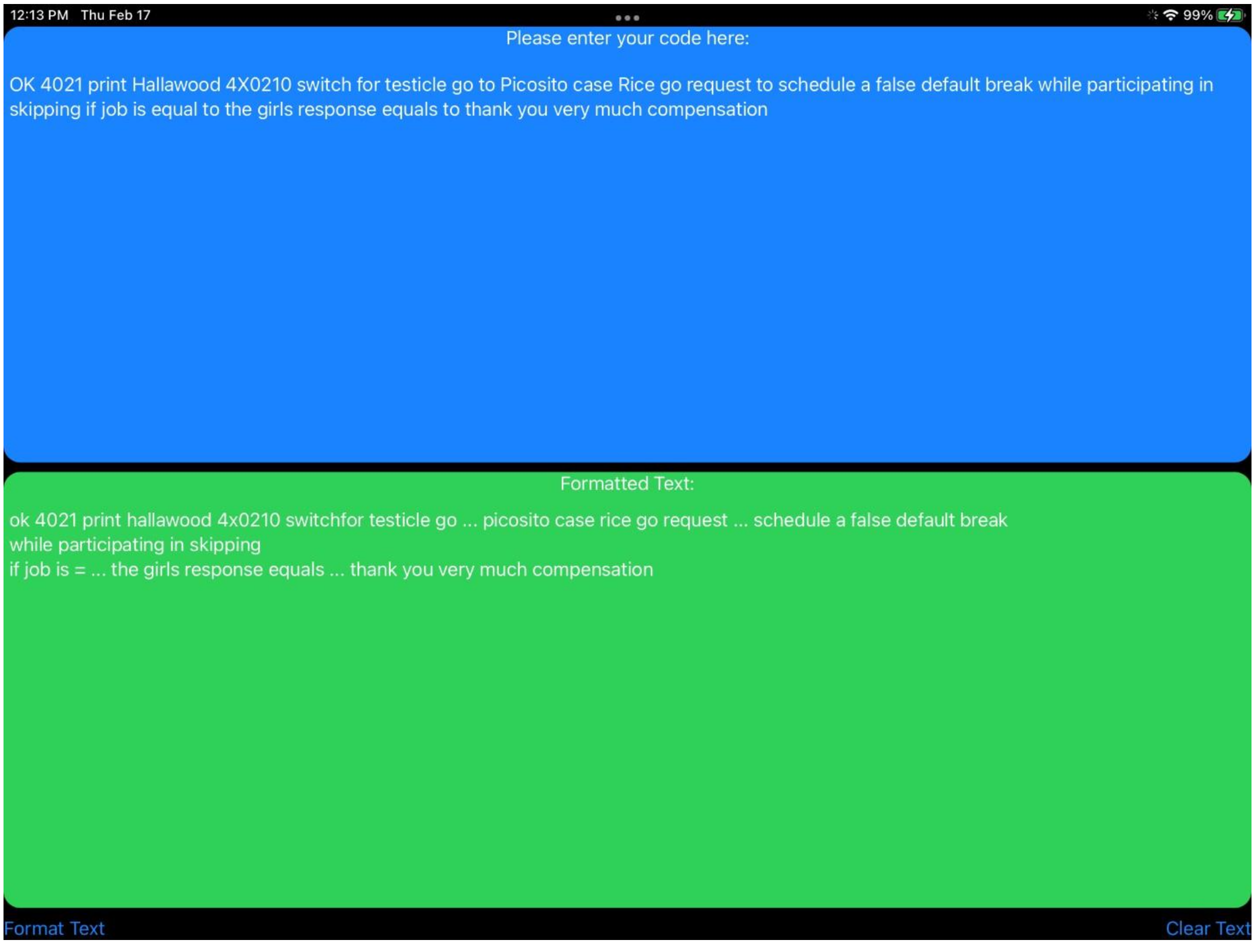
H.18) Participant 20



40214X0 toone variable test is equal to zero switch for the stucco guy is rice good is going through a sketch of Cruz going to force the fall break while participating is greater than skipping this job is equal to good variable response is equal to thank you very much compensation +10

40214x0 toone variable test is = 0 switchfor the stucco guy is rice good is going through a sketch of cruz going ... force the fall break
while participating is greater than skipping this job is = ... good variable response is = ... thank you very much compensation +10

H.19) Participant 21



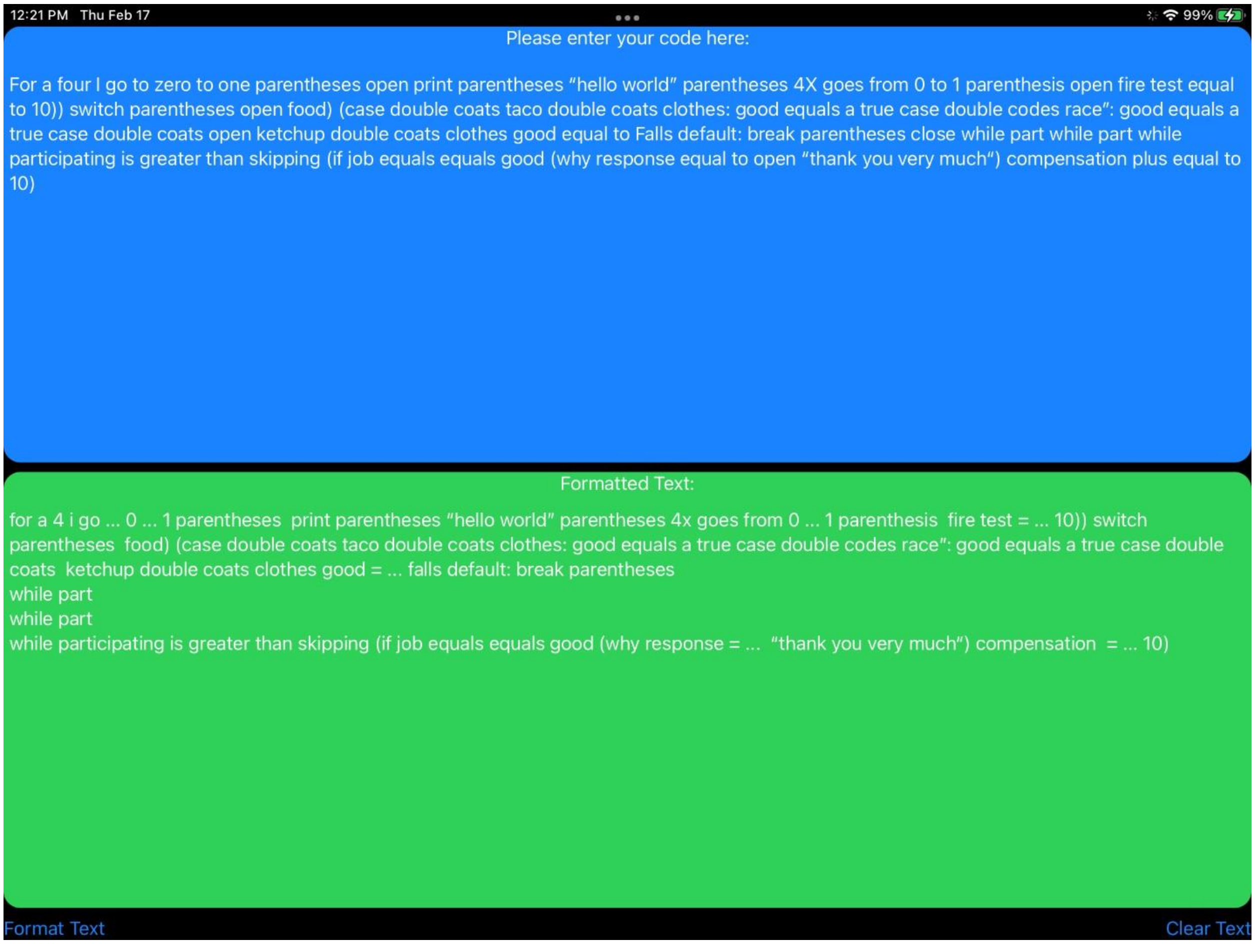
Please enter your code here:

OK 4021 print Hallawood 4X0210 switch for testicle go to Picosito case Rice go request to schedule a false default break while participating in skipping if job is equal to the girls response equals to thank you very much compensation

Formatted Text:

```
ok 4021 print hallawood 4x0210 switchfor testicle go ... picosito case rice go request ... schedule a false default break
while participating in skipping
if job is = ... the girls response equals ... thank you very much compensation
```

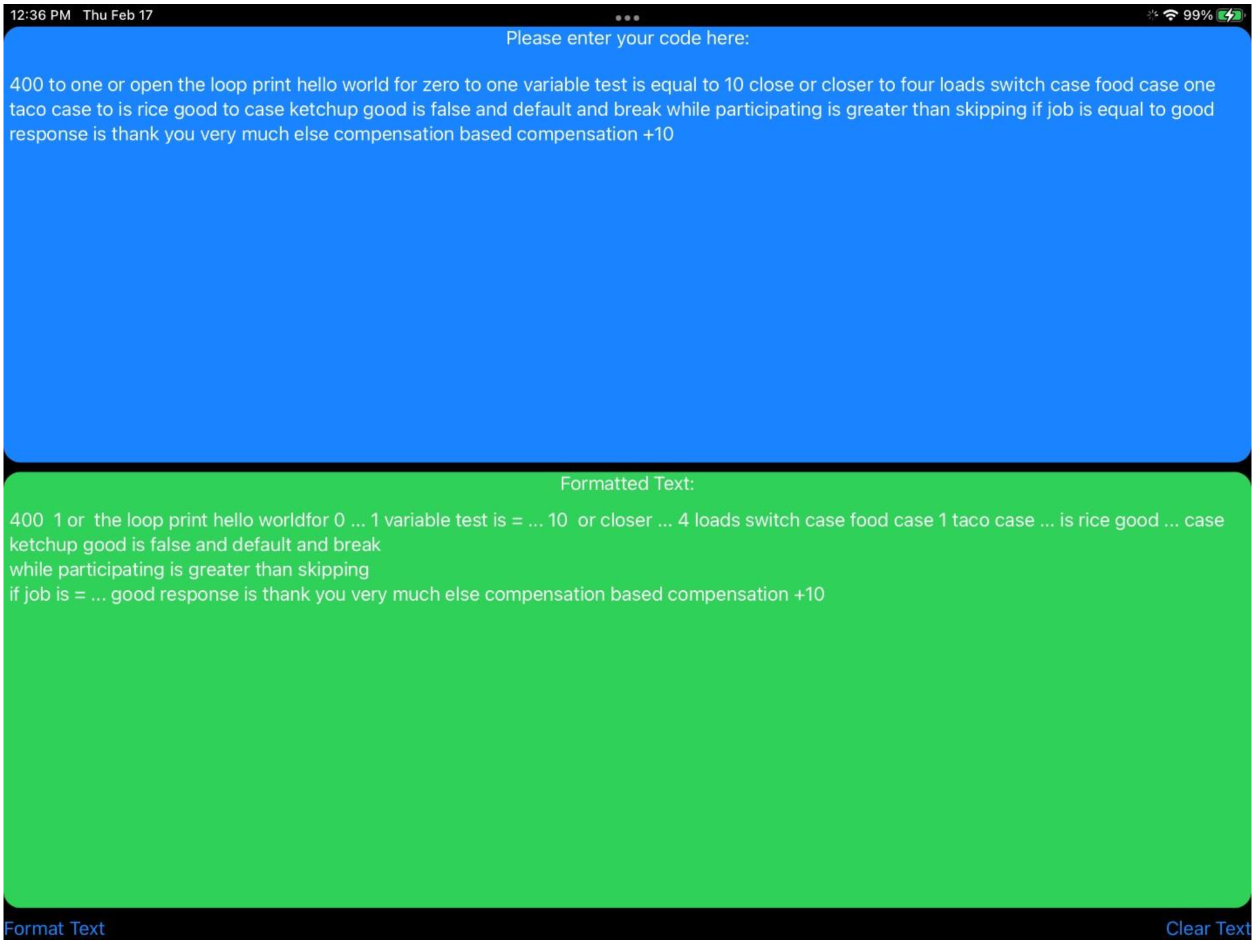
H.20) Participant 22



For a four I go to zero to one parentheses open print parentheses "hello world" parentheses 4X goes from 0 to 1 parenthesis open fire test equal to 10)) switch parentheses open food) (case double coats taco double coats clothes: good equals a true case double codes race": good equals a true case double coats open ketchup double coats clothes good equal to Falls default: break parentheses close while part while part while participating is greater than skipping (if job equals equals good (why response equal to open "thank you very much") compensation plus equal to 10)

```
for a 4 i go ... 0 ... 1 parentheses  print parentheses "hello world" parentheses 4x goes from 0 ... 1 parenthesis  fire test = ... 10)) switch parentheses  food) (case double coats taco double coats clothes: good equals a true case double codes race": good equals a true case double coats  ketchup double coats clothes good = ... falls default: break parentheses
while part
while part
while participating is greater than skipping (if job equals equals good (why response = ...  "thank you very much") compensation  = ... 10)
```

H.21) Participant 23



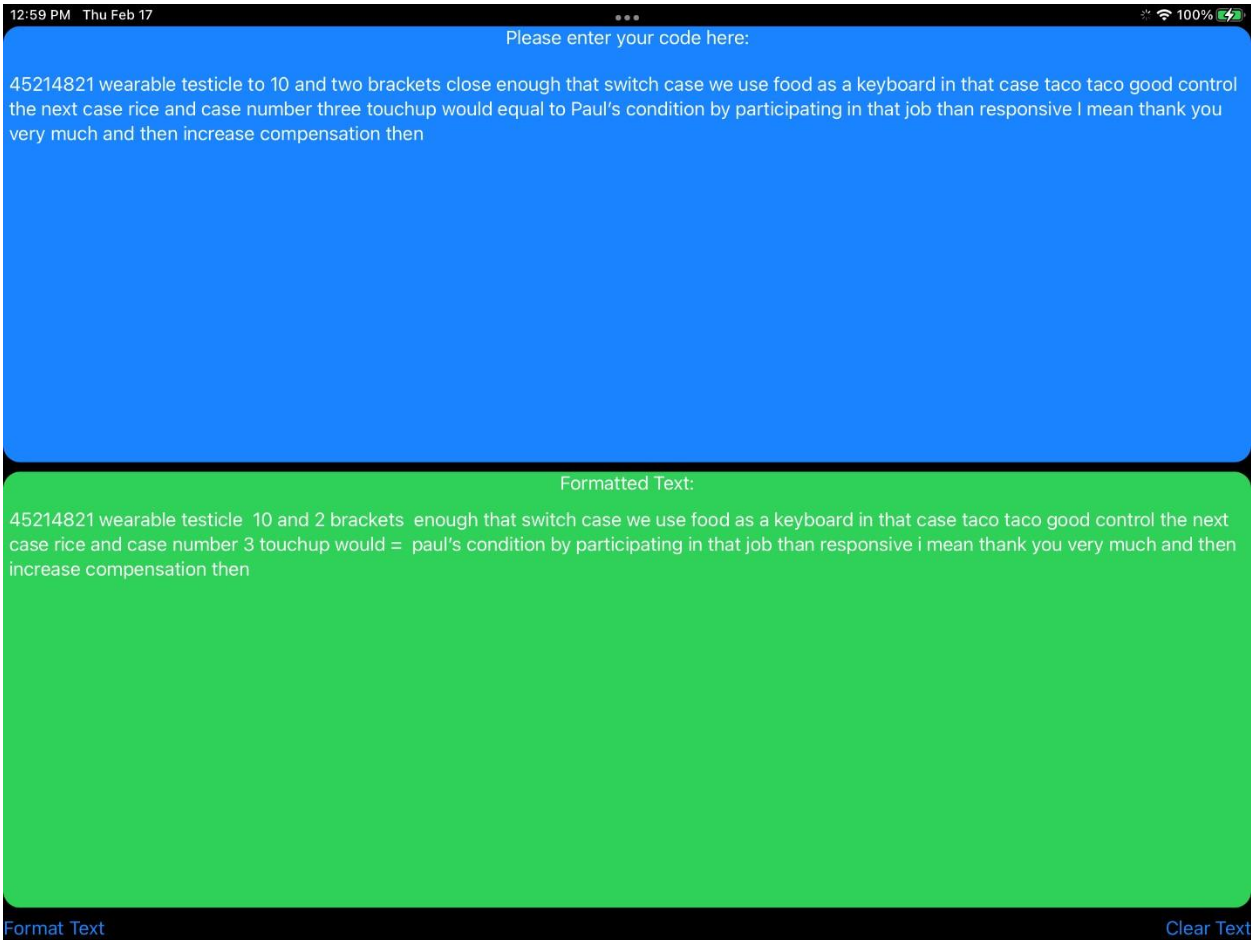
Please enter your code here:

400 to one or open the loop print hello world for zero to one variable test is equal to 10 close or closer to four loads switch case food case one taco case to is rice good to case ketchup good is false and default and break while participating is greater than skipping if job is equal to good response is thank you very much else compensation based compensation +10

Formatted Text:

```
400  1 or  the loop print hello worldfor 0 ... 1 variable test is = ... 10  or closer ... 4 loads switch case food case 1 taco case ... is rice good ... case ketchup good is false and default and break
while participating is greater than skipping
if job is = ... good response is thank you very much else compensation based compensation +10
```

H.22) Participant 24



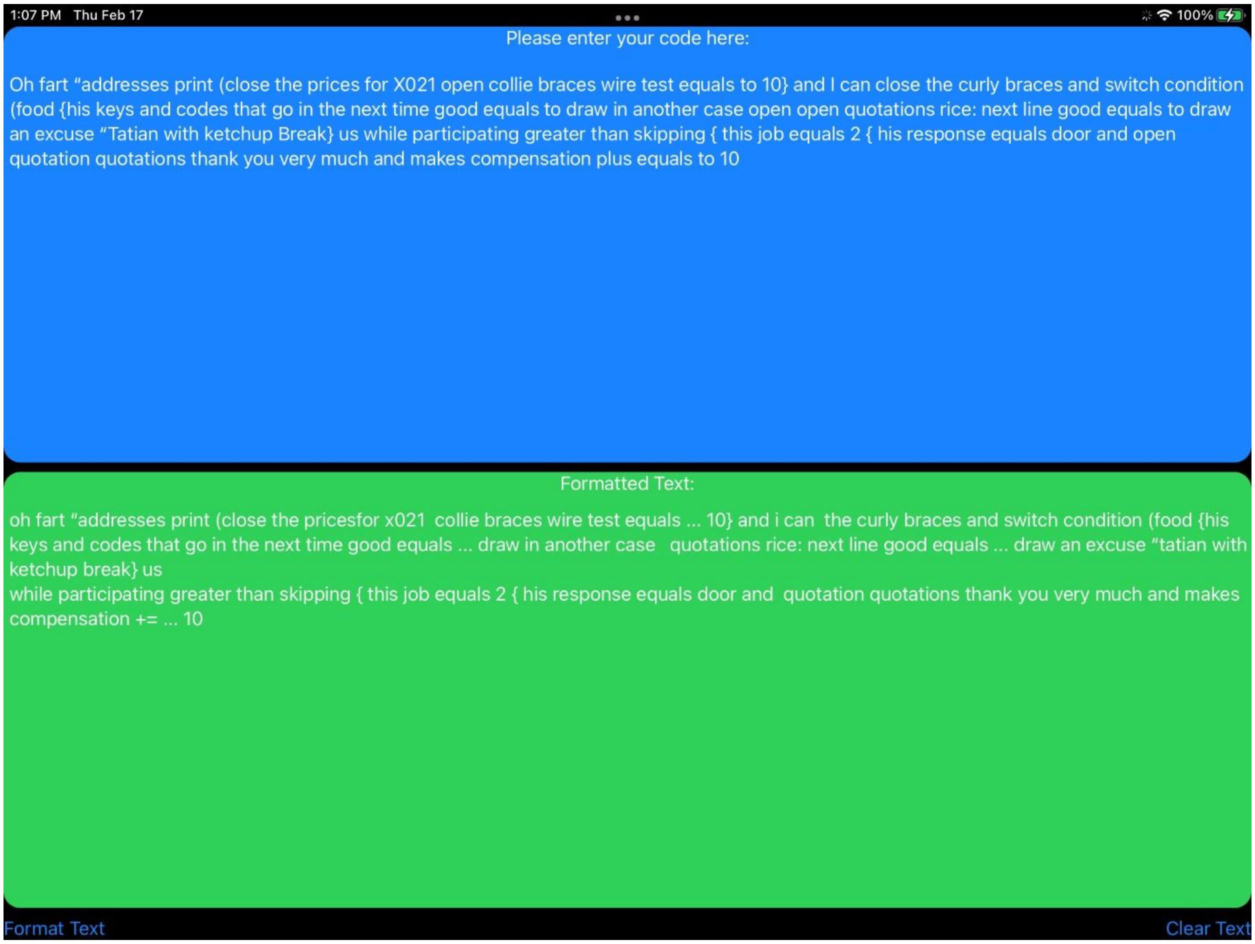
Please enter your code here:

45214821 wearable testicle to 10 and two brackets close enough that switch case we use food as a keyboard in that case taco taco good control the next case rice and case number three touchup would equal to Paul’s condition by participating in that job than responsive I mean thank you very much and then increase compensation then

Formatted Text:

45214821 wearable testicle 10 and 2 brackets enough that switch case we use food as a keyboard in that case taco taco good control the next case rice and case number 3 touchup would = paul’s condition by participating in that job than responsive i mean thank you very much and then increase compensation then

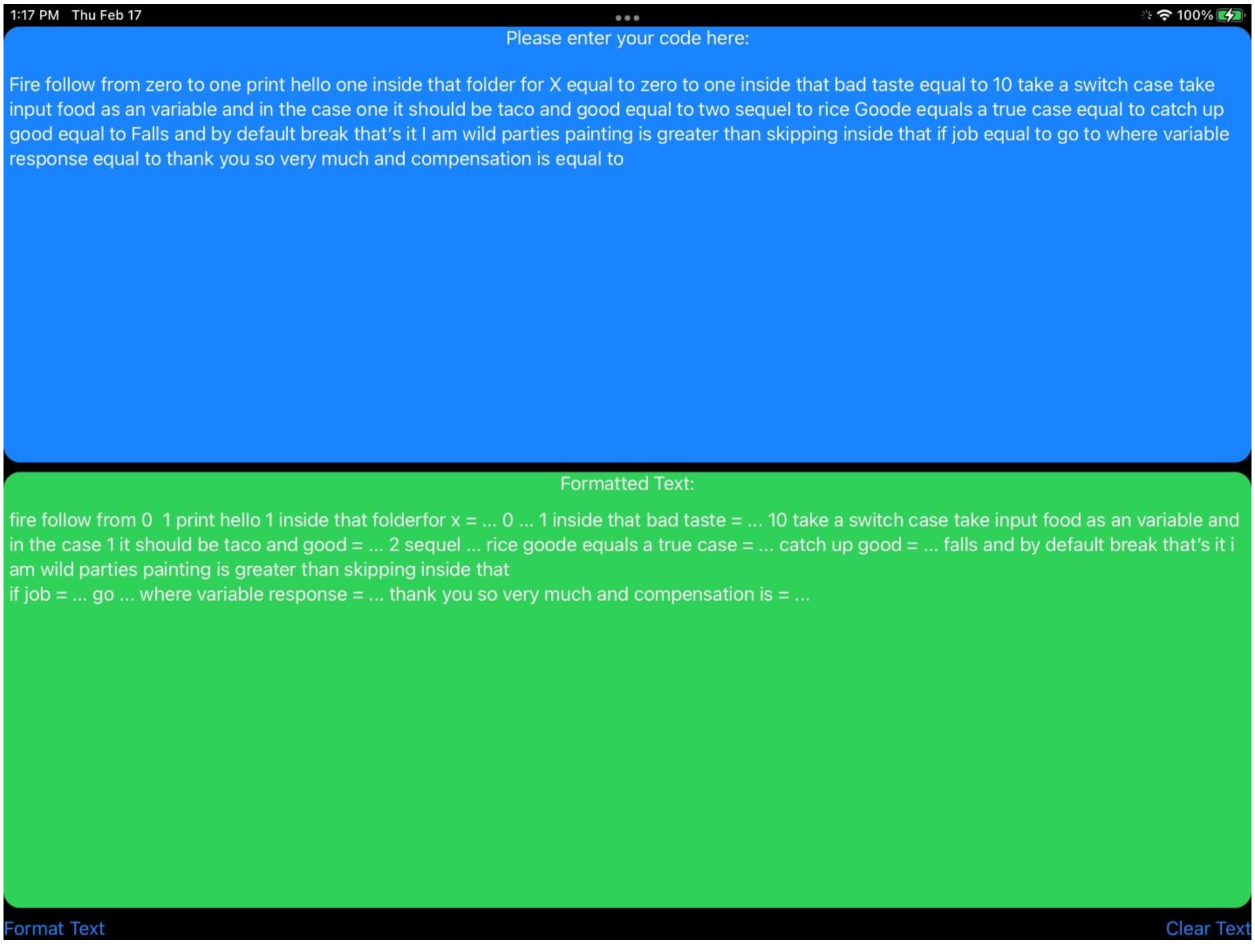
H.23) Participant 25



```
Oh fart “addresses print (close the prices for X021 open collie braces wire test equals to 10} and I can close the curly braces and switch condition (food {his keys and codes that go in the next time good equals to draw in another case open open quotations rice: next line good equals to draw an excuse “Tatian with ketchup Break} us while participating greater than skipping { this job equals 2 { his response equals door and open quotation quotations thank you very much and makes compensation plus equals to 10
```

```
oh fart “addresses print (close the pricesfor x021  collie braces wire test equals ... 10} and i can  the curly braces and switch condition (food {his keys and codes that go in the next time good equals ... draw in another case   quotations rice: next line good equals ... draw an excuse “tatian with ketchup break} us
while participating greater than skipping { this job equals 2 { his response equals door and  quotation quotations thank you very much and makes compensation += ... 10
```

H.24) Participant 26



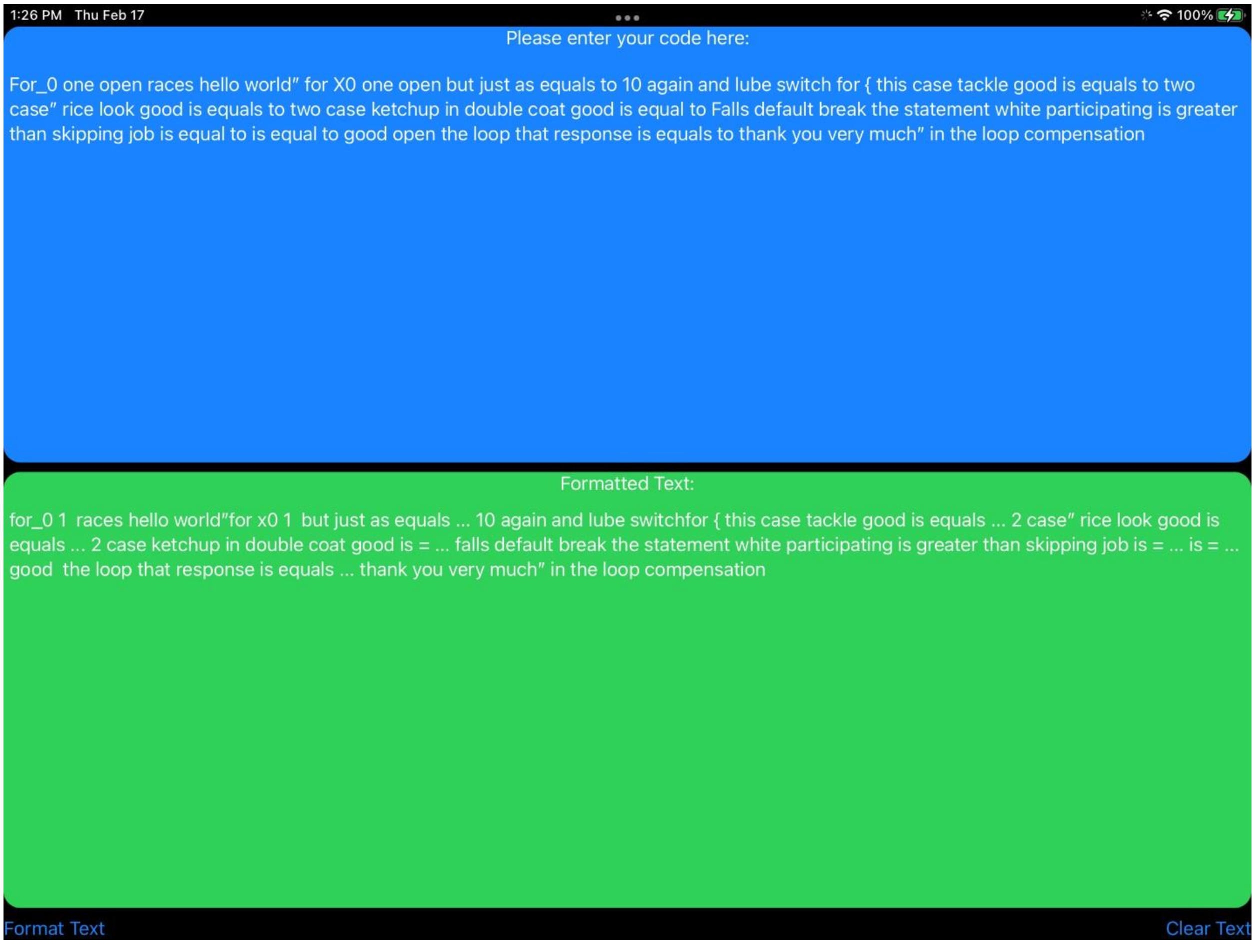
Please enter your code here:

Fire follow from zero to one print hello one inside that folder for X equal to zero to one inside that bad taste equal to 10 take a switch case take input food as an variable and in the case one it should be taco and good equal to two sequel to rice Goode equals a true case equal to catch up good equal to Falls and by default break that’s it I am wild parties painting is greater than skipping inside that if job equal to go to where variable response equal to thank you so very much and compensation is equal to

Formatted Text:

```
fire follow from 0  1 print hello 1 inside that folderfor x = ... 0 ... 1 inside that bad taste = ... 10 take a switch case take input food as an variable and in the case 1 it should be taco and good = ... 2 sequel ... rice goode equals a true case = ... catch up good = ... falls and by default break that’s it i am wild parties painting is greater than skipping inside that
if job = ... go ... where variable response = ... thank you so very much and compensation is = ...
```

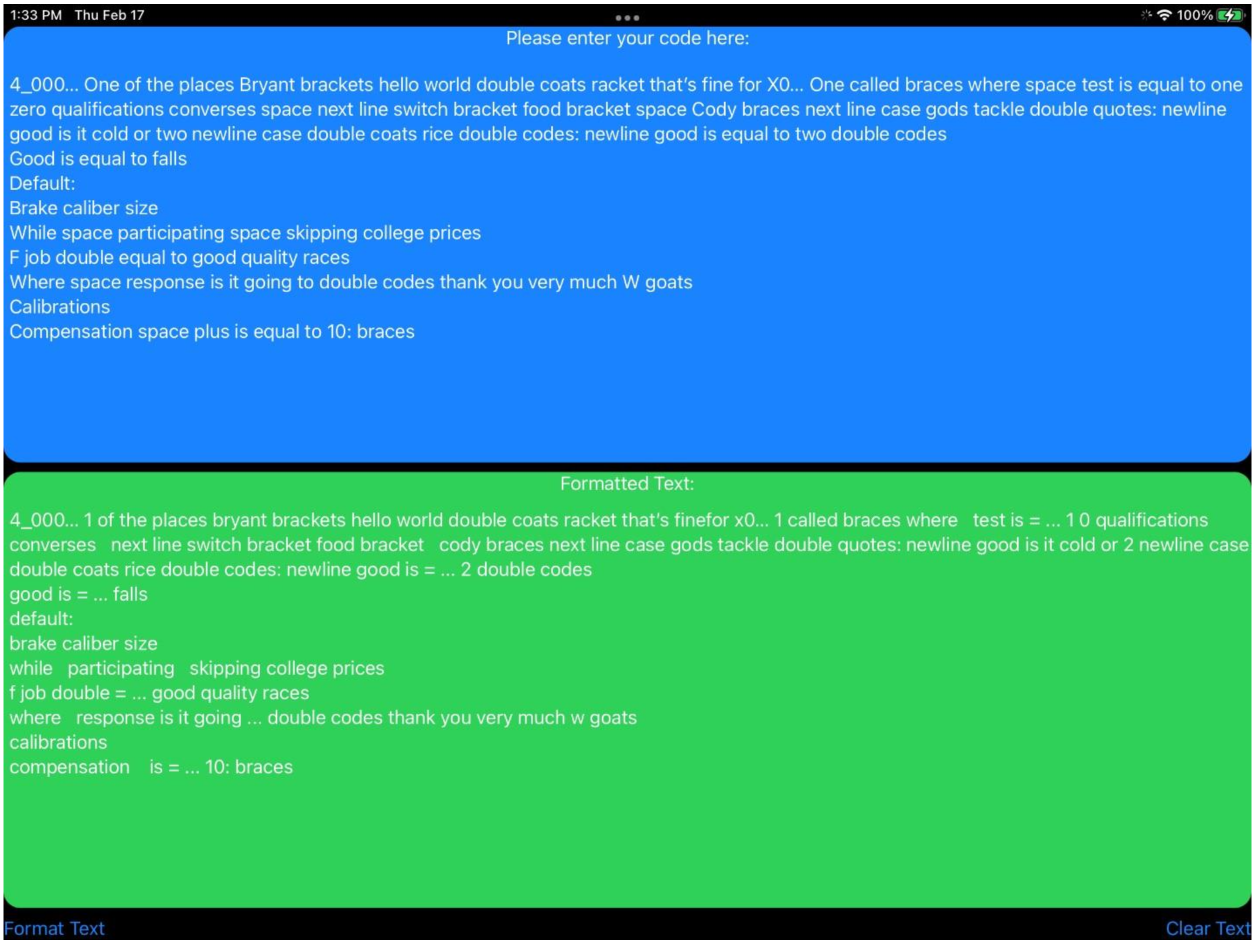
H.25) Participant 27



```
For_0 one open races hello world" for X0 one open but just as equals to 10 again and lube switch for { this case tackle good is equals to two case" rice look good is equals to two case ketchup in double coat good is equal to Falls default break the statement white participating is greater than skipping job is equal to is equal to good open the loop that response is equals to thank you very much" in the loop compensation
```

```
for_0 1  races hello world"for x0 1  but just as equals ... 10 again and lube switchfor { this case tackle good is equals ... 2 case" rice look good is equals ... 2 case ketchup in double coat good is = ... falls default break the statement white participating is greater than skipping job is = ... is = ... good  the loop that response is equals ... thank you very much" in the loop compensation
```

H.26) Participant 28



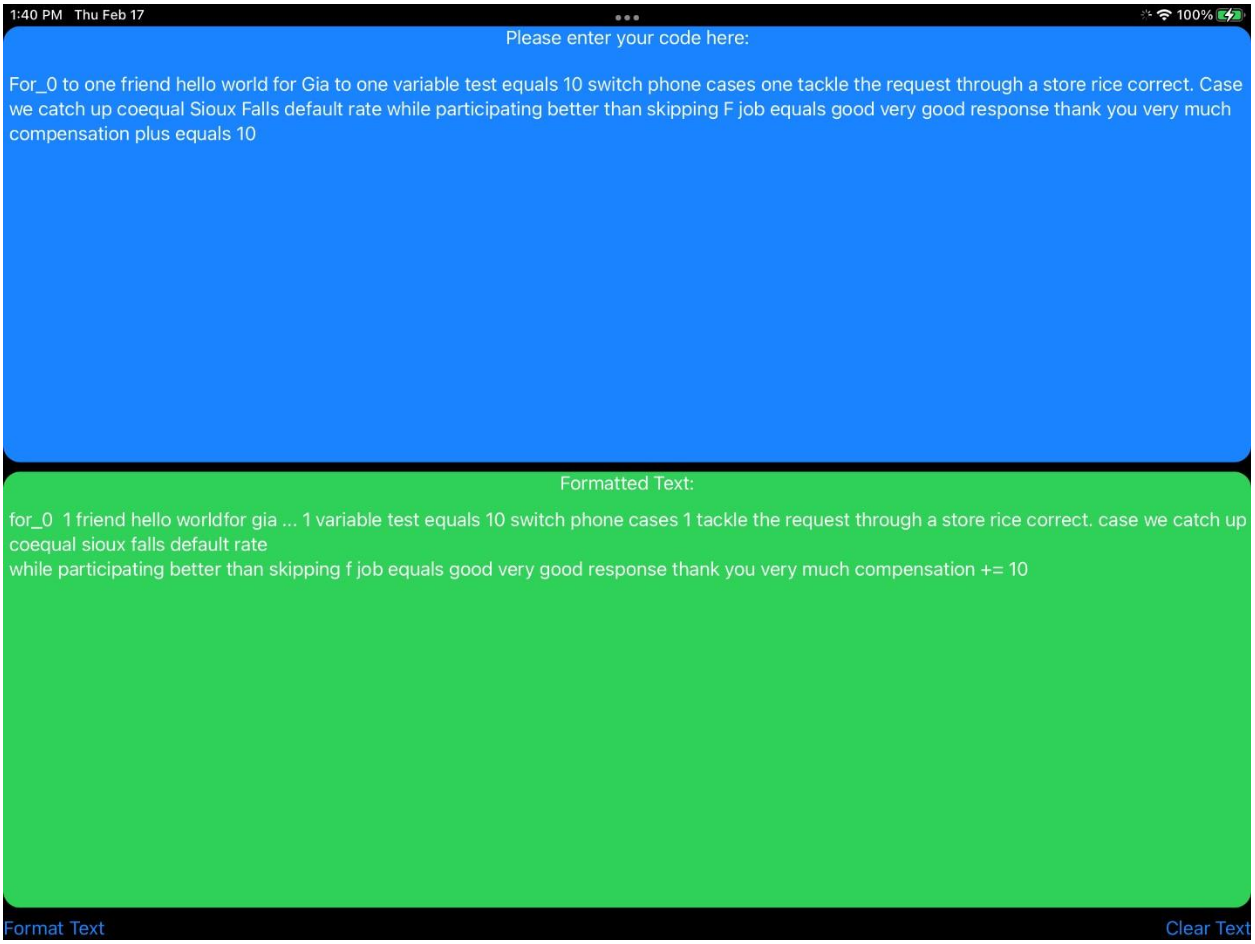
Please enter your code here:

```
4_000... One of the places Bryant brackets hello world double coats racket that's fine for X0... One called braces where space test is equal to one zero qualifications converses space next line switch bracket food bracket space Cody braces next line case gods tackle double quotes: newline good is it cold or two newline case double coats rice double codes: newline good is equal to two double codes
Good is equal to falls
Default:
Brake caliber size
While space participating space skipping college prices
F job double equal to good quality races
Where space response is it going to double codes thank you very much W goats
Calibrations
Compensation space plus is equal to 10: braces
```

Formatted Text:

```
4_000... 1 of the places bryant brackets hello world double coats racket that's finefor x0... 1 called braces where   test is = ... 1 0 qualifications converses   next line switch bracket food bracket   cody braces next line case gods tackle double quotes: newline good is it cold or 2 newline case double coats rice double codes: newline good is = ... 2 double codes
good is = ... falls
default:
brake caliber size
while   participating   skipping college prices
f job double = ... good quality races
where   response is it going ... double codes thank you very much w goats
calibrations
compensation    is = ... 10: braces
```

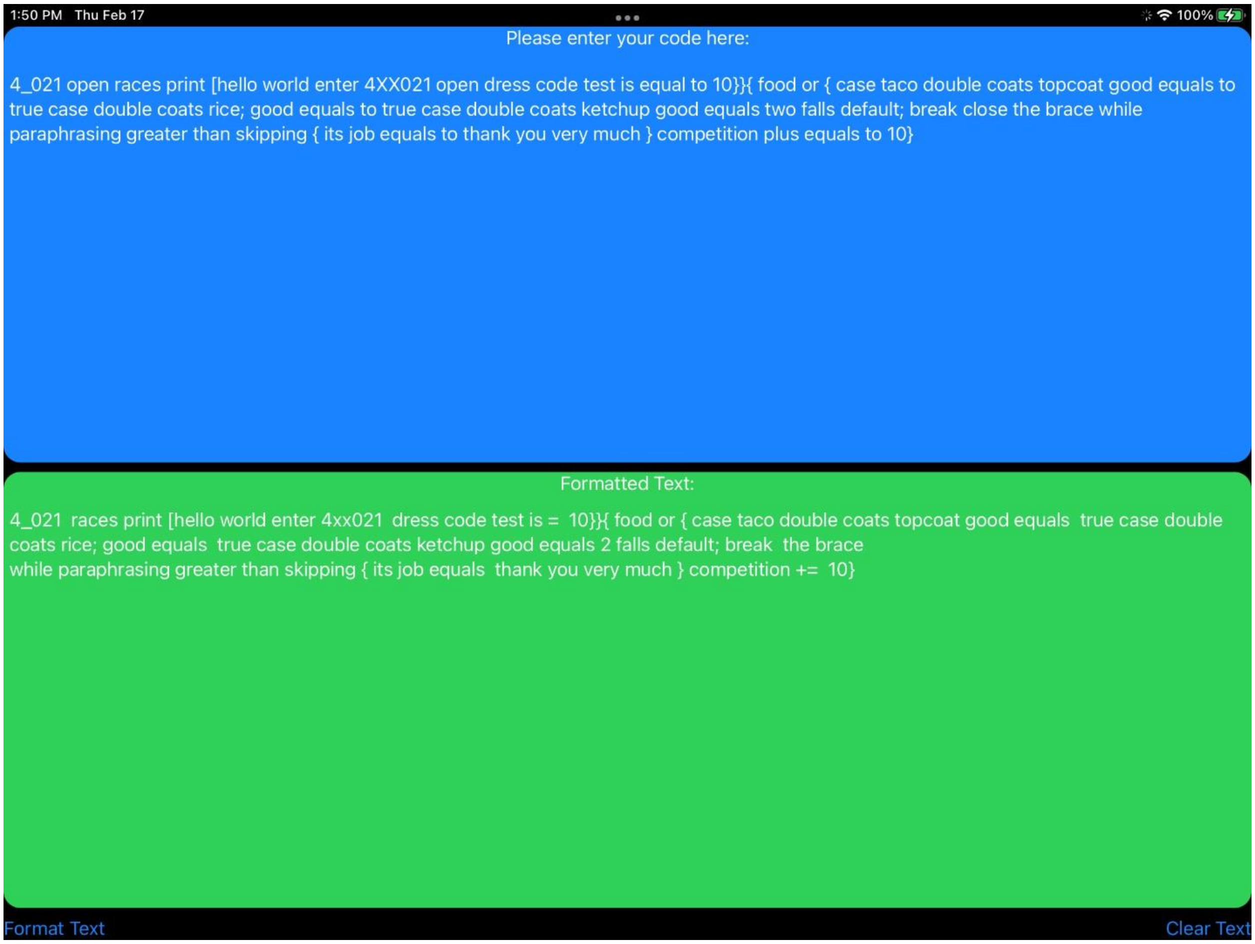
H.27) Participant 29



For_0 to one friend hello world for Gia to one variable test equals 10 switch phone cases one tackle the request through a store rice correct. Case we catch up coequal Sioux Falls default rate while participating better than skipping F job equals good very good response thank you very much compensation plus equals 10

```
for_0  1 friend hello worldfor gia ... 1 variable test equals 10 switch phone cases 1 tackle the request through a store rice correct. case we catch up coequal sioux falls default rate
while participating better than skipping f job equals good very good response thank you very much compensation += 10
```

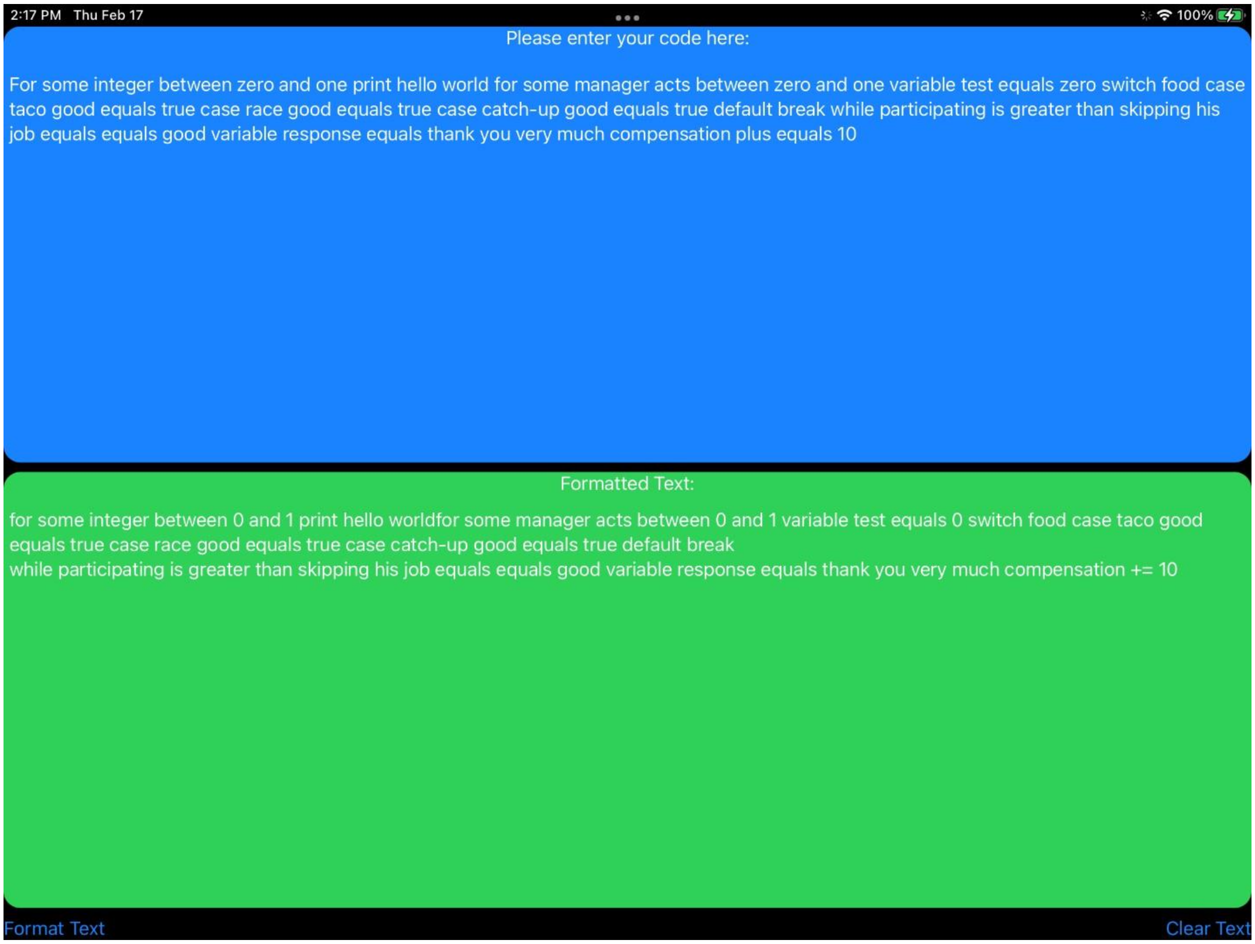
H.28) Participant 30



4_021 open races print [hello world enter 4XX021 open dress code test is equal to 10}}{ food or { case taco double coats topcoat good equals to true case double coats rice; good equals to true case double coats ketchup good equals two falls default; break close the brace while paraphrasing greater than skipping { its job equals to thank you very much } competition plus equals to 10}

```
4_021  races print [hello world enter 4xx021  dress code test is =  10}}{ food or { case taco double coats topcoat good equals  true case double coats rice; good equals  true case double coats ketchup good equals 2 falls default; break  the brace
while paraphrasing greater than skipping { its job equals  thank you very much } competition +=  10}
```

H.29) Participant 31



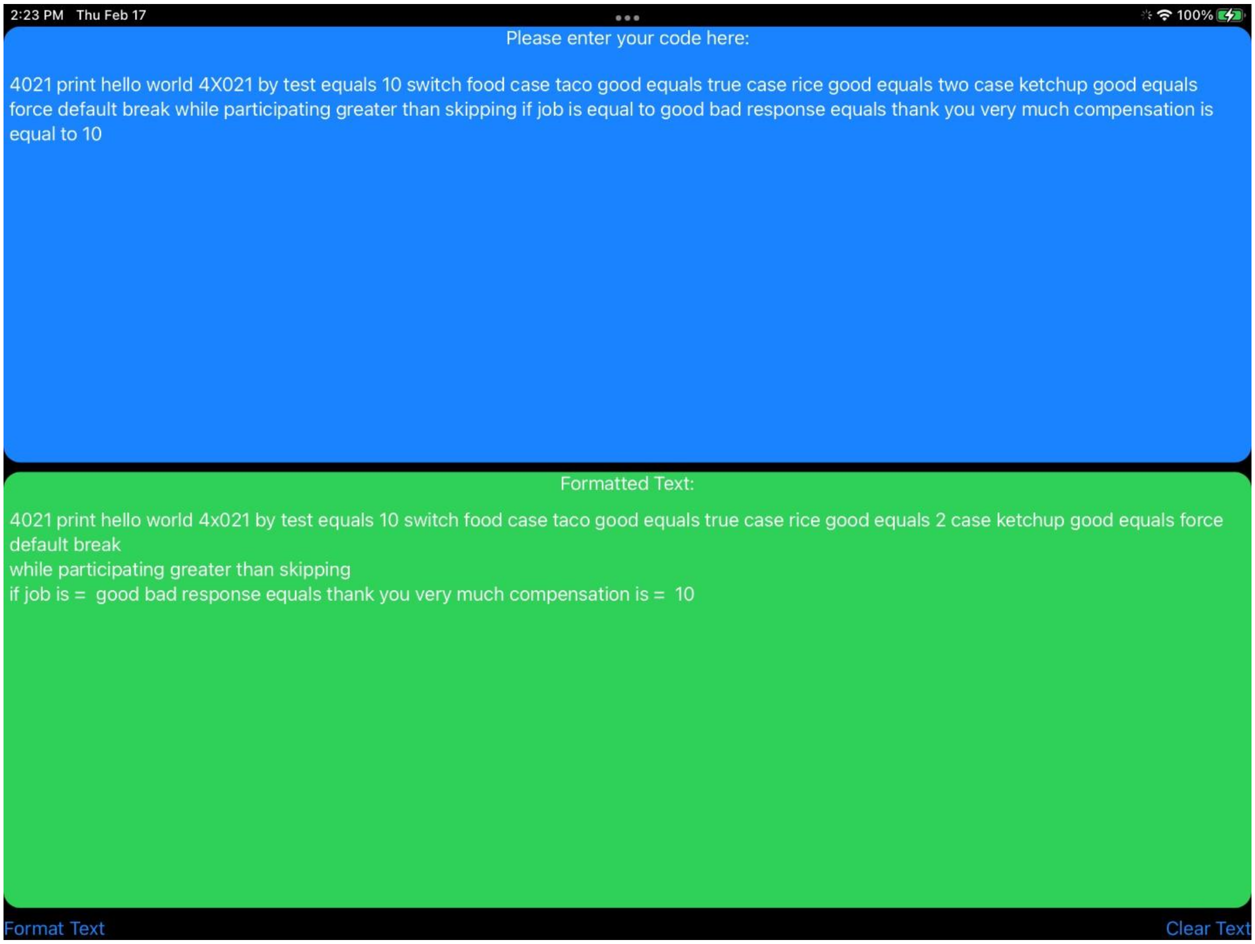
Please enter your code here:

For some integer between zero and one print hello world for some manager acts between zero and one variable test equals zero switch food case taco good equals true case race good equals true case catch-up good equals true default break while participating is greater than skipping his job equals equals good variable response equals thank you very much compensation plus equals 10

Formatted Text:

```
for some integer between 0 and 1 print hello worldfor some manager acts between 0 and 1 variable test equals 0 switch food case taco good equals true case race good equals true case catch-up good equals true default break
while participating is greater than skipping his job equals equals good variable response equals thank you very much compensation += 10
```

H.30) Participant 32



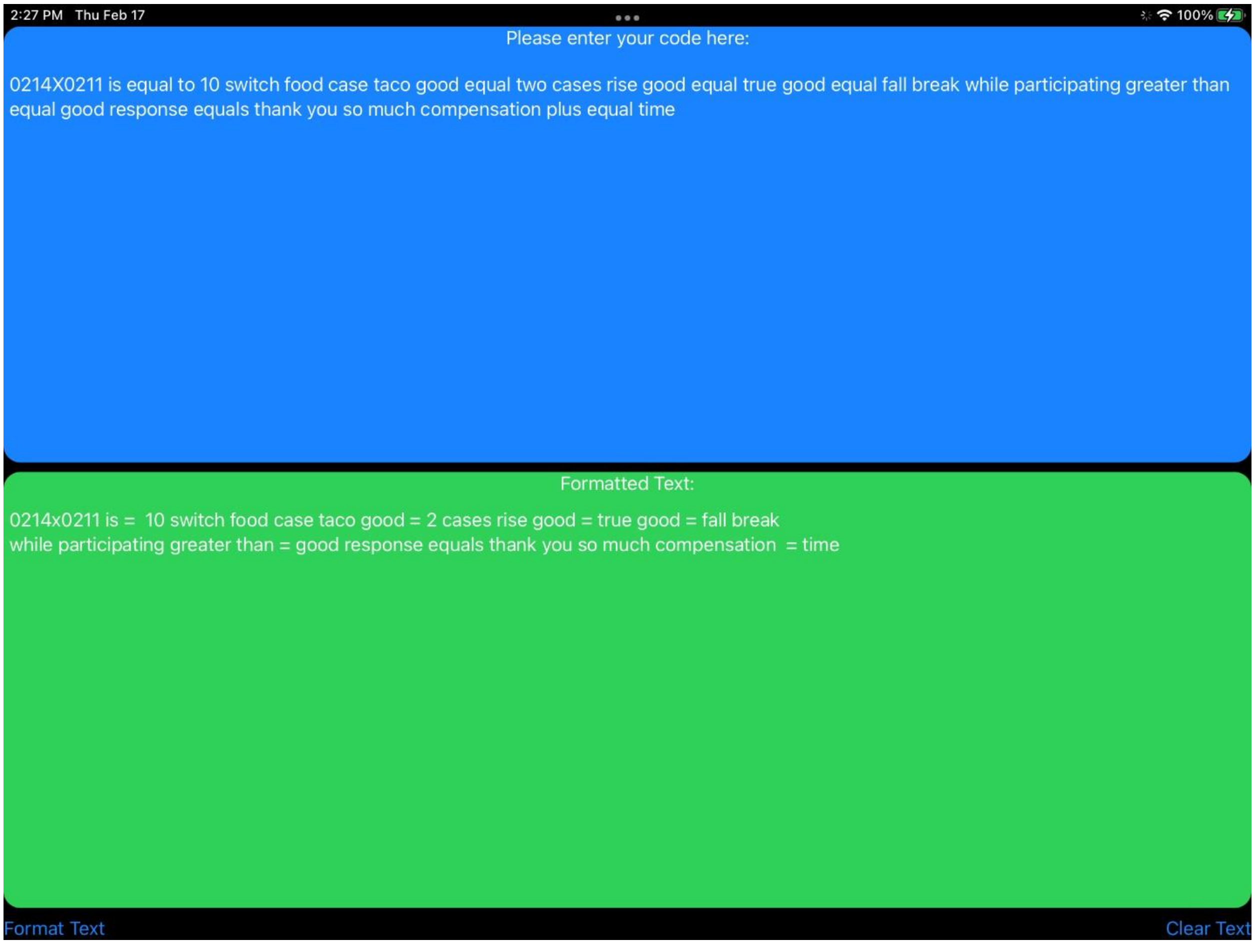
Please enter your code here:

4021 print hello world 4X021 by test equals 10 switch food case taco good equals true case rice good equals two case ketchup good equals force default break while participating greater than skipping if job is equal to good bad response equals thank you very much compensation is equal to 10

Formatted Text:

```
4021 print hello world 4x021 by test equals 10 switch food case taco good equals true case rice good equals 2 case ketchup good equals force default break
while participating greater than skipping
if job is =  good bad response equals thank you very much compensation is =  10
```

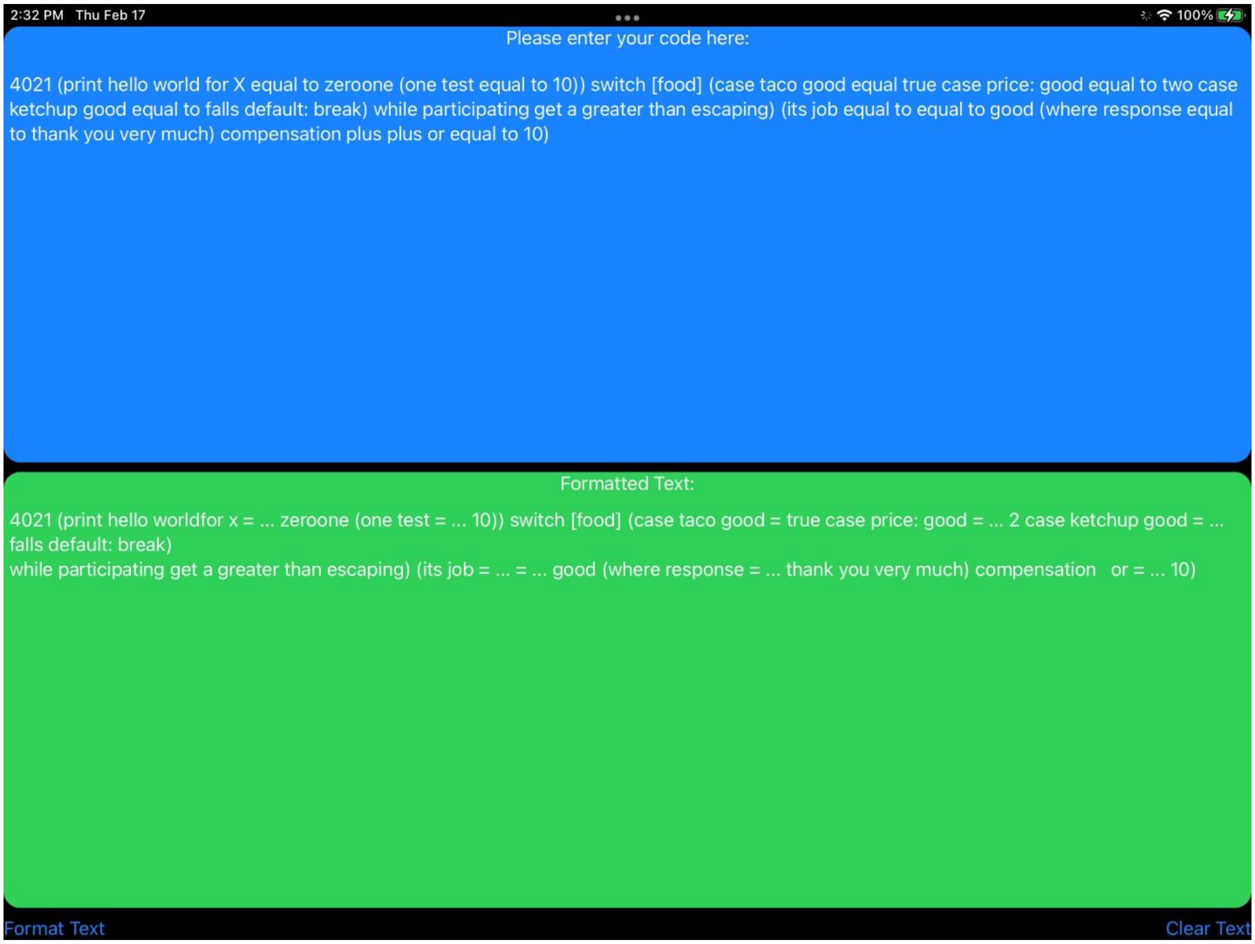
H.31) Participant 33



```
0214X0211 is equal to 10 switch food case taco good equal two cases rise good equal true good equal fall break while participating greater than equal good response equals thank you so much compensation plus equal time
```

```
0214x0211 is =  10 switch food case taco good = 2 cases rise good = true good = fall break
while participating greater than = good response equals thank you so much compensation  = time
```

H.32) Participant 34



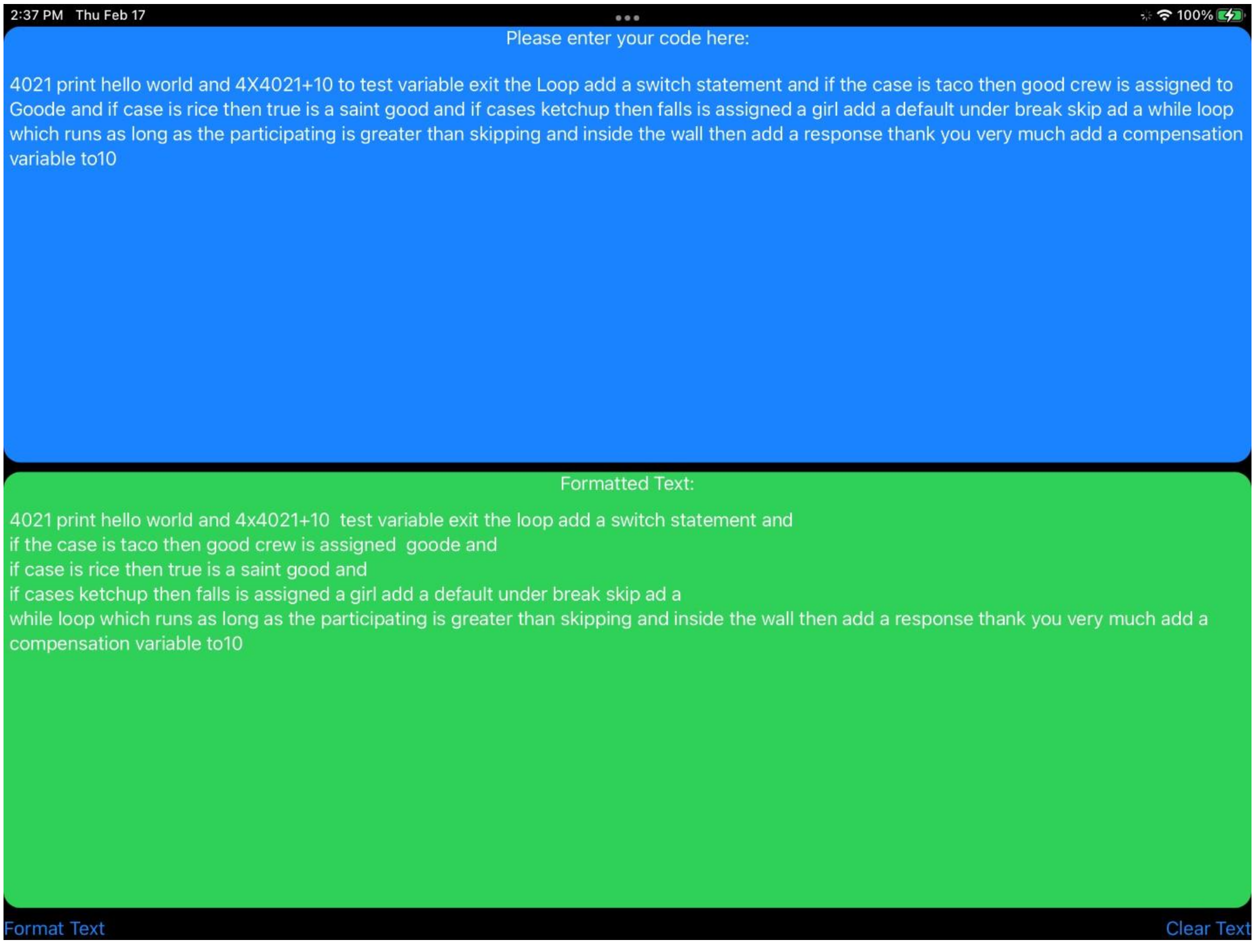
Please enter your code here:

```
4021 (print hello world for X equal to zeroone (one test equal to 10)) switch [food] (case taco good equal true case price: good equal to two case ketchup good equal to falls default: break) while participating get a greater than escaping) (its job equal to equal to good (where response equal to thank you very much) compensation plus plus or equal to 10)
```

Formatted Text:

```
4021 (print hello worldfor x = ... zeroone (one test = ... 10)) switch [food] (case taco good = true case price: good = ... 2 case ketchup good = ... falls default: break)
while participating get a greater than escaping) (its job = ... = ... good (where response = ... thank you very much) compensation   or = ... 10)
```

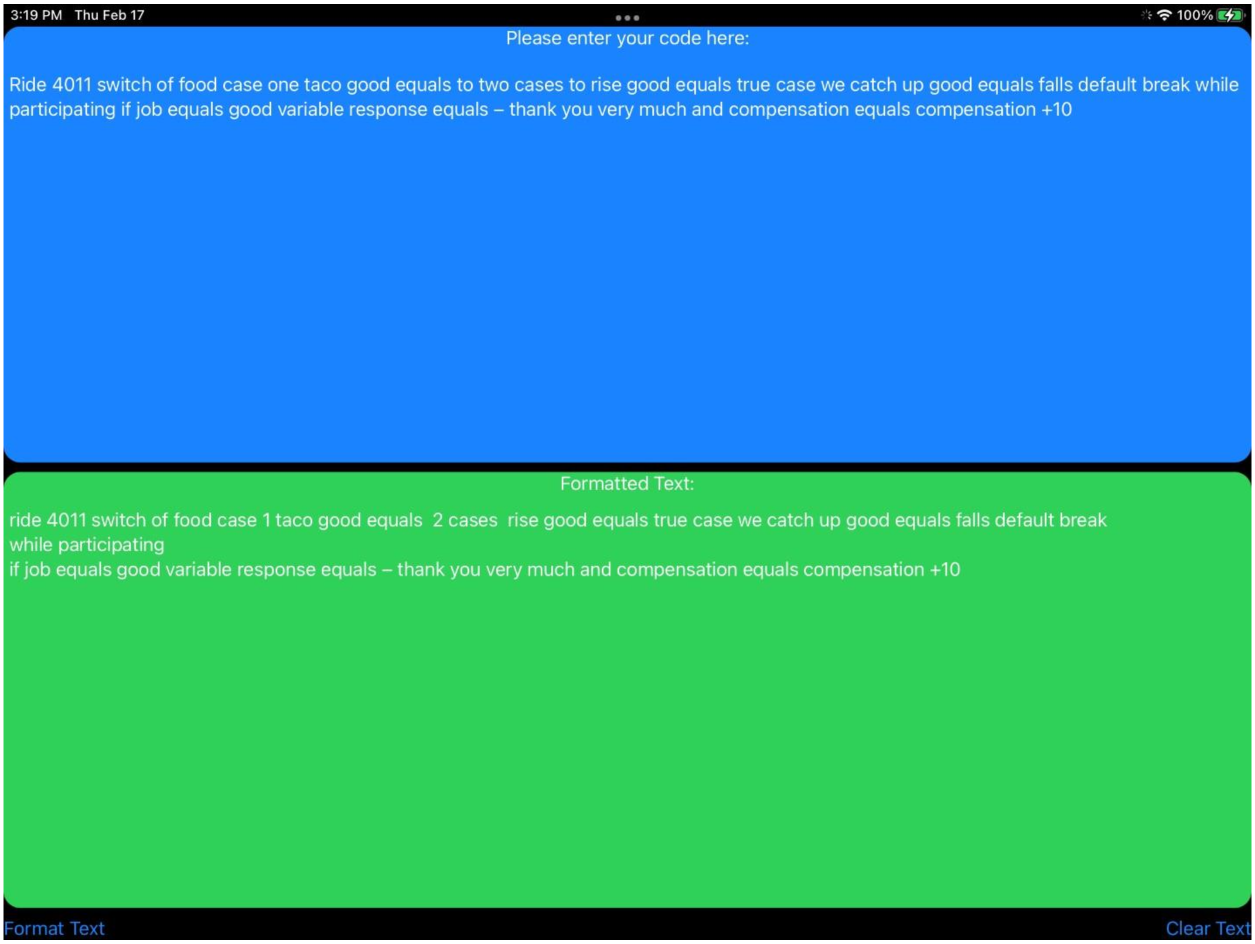
H.33) Participant 35



4021 print hello world and 4X4021+10 to test variable exit the Loop add a switch statement and if the case is taco then good crew is assigned to Goode and if case is rice then true is a saint good and if cases ketchup then falls is assigned a girl add a default under break skip ad a while loop which runs as long as the participating is greater than skipping and inside the wall then add a response thank you very much add a compensation variable to10

```
4021 print hello world and 4x4021+10  test variable exit the loop add a switch statement and
if the case is taco then good crew is assigned  goode and
if case is rice then true is a saint good and
if cases ketchup then falls is assigned a girl add a default under break skip ad a
while loop which runs as long as the participating is greater than skipping and inside the wall then add a response thank you very much add a
compensation variable to10
```

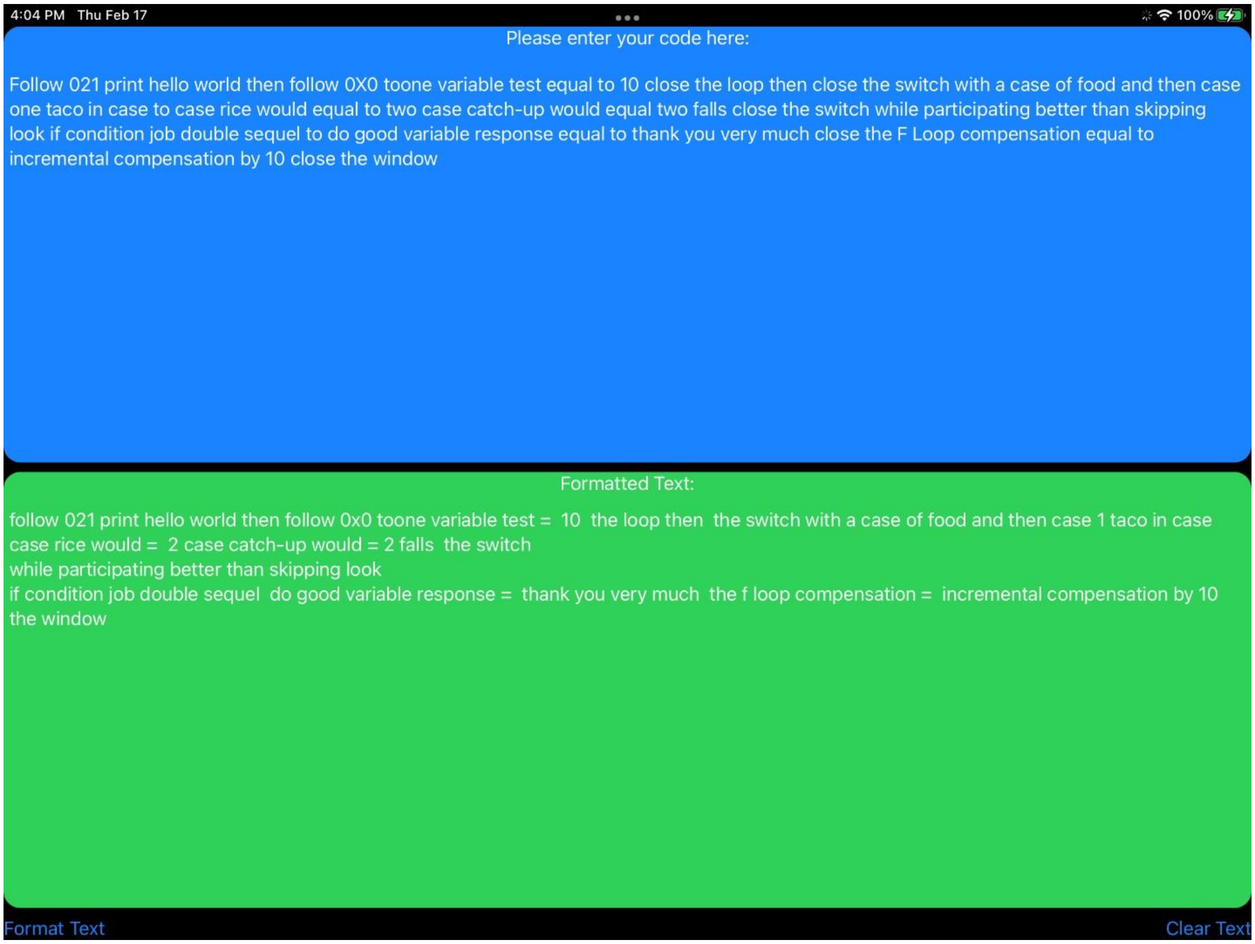
H.34) Participant 36



Ride 4011 switch of food case one taco good equals to two cases to rise good equals true case we catch up good equals falls default break while participating if job equals good variable response equals – thank you very much and compensation equals compensation +10

```
ride 4011 switch of food case 1 taco good equals  2 cases  rise good equals true case we catch up good equals falls default break
while participating
if job equals good variable response equals – thank you very much and compensation equals compensation +10
```

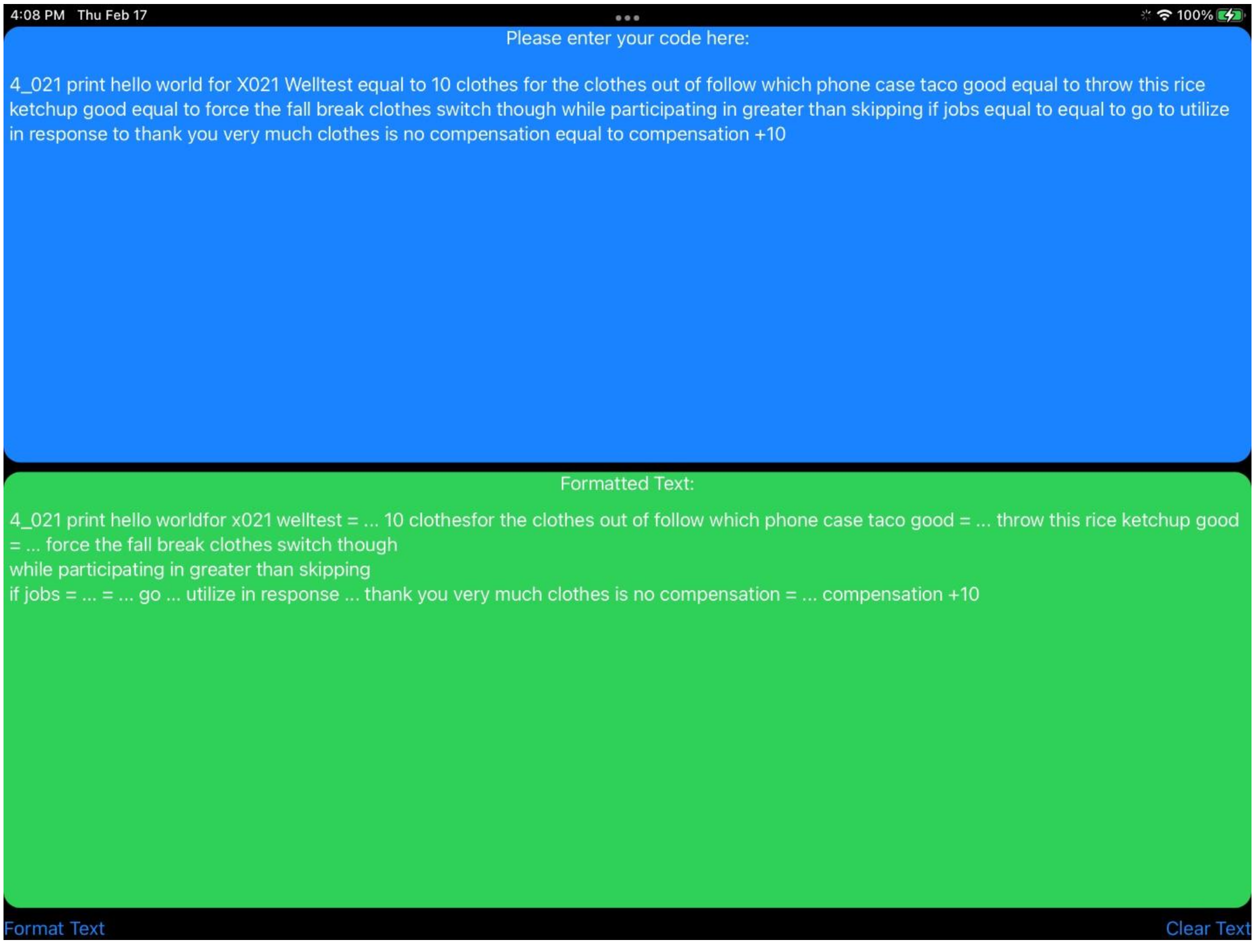
H.35) Participant 37



Follow 021 print hello world then follow 0X0 toone variable test equal to 10 close the loop then close the switch with a case of food and then case one taco in case to case rice would equal to two case catch-up would equal two falls close the switch while participating better than skipping look if condition job double sequel to do good variable response equal to thank you very much close the F Loop compensation equal to incremental compensation by 10 close the window

```
follow 021 print hello world then follow 0x0 toone variable test =  10  the loop then  the switch with a case of food and then case 1 taco in case
case rice would =  2 case catch-up would = 2 falls  the switch
while participating better than skipping look
if condition job double sequel  do good variable response =  thank you very much  the f loop compensation =  incremental compensation by 10
the window
```

H.36) Participant 38



Please enter your code here:

4_021 print hello world for X021 Welltest equal to 10 clothes for the clothes out of follow which phone case taco good equal to throw this rice ketchup good equal to force the fall break clothes switch though while participating in greater than skipping if jobs equal to equal to go to utilize in response to thank you very much clothes is no compensation equal to compensation +10

Formatted Text:

```
4_021 print hello worldfor x021 welltest = ... 10 clothesfor the clothes out of follow which phone case taco good = ... throw this rice ketchup good = ... force the fall break clothes switch though
while participating in greater than skipping
if jobs = ... = ... go ... utilize in response ... thank you very much clothes is no compensation = ... compensation +10
```