Initialise global constant list of the 20 different sentences that the player might type called *POSSIBLE\_SENTENCES*

Loop forever

Initialise empty list called *accuracies*

Initialise empty list called *playerWPMs*

Display “Welcome to Joshua’s Typing Test: ”

Loop forever

try

Display “How many rounds do you want the game to be (must be between 3 and ” + length of *POSSIBLE\_SENTENCES* + “): ”

Prompt for input and save as *numberOfRounds*

If the user didn’t input anything for *numberOfRounds* then

Set *numberOfRounds* to the length of *POSSIBLE\_SENTENCES*

If *numberOfRounds* is between 3 and the length of *POSSIBLE\_SENTENCES* then

Break

Throw division by zero error

except

Display “Please enter an integer between 3 and ” + length of *POSSIBLE\_SENTENCES*

Display “Strict Mode means that your answers are case sensitive”

Loop forever

Prompt user if they want to use strict mode (y/n) and save as *strictModeStr*

Set *strictModeStr* to all lowercase

If *strictModeStr* is “y” or “yes” or “” then

Set global *strict* as True

Break

If *strictModeStr* is “n” or “N” then

Set global *strict* as False

Break

Display “Please enter a ‘y’ or ‘n’ for yes or no respectively”

Display “Your test will last ” + *numberOfRounds* + “ Rounds”

If *strict* then

Display “And your input is case sensitive”

Else Display “And your input is not case sensitive”

Display “If you want to finish early then just enter ‘x’ ”

Initialise empty list *sentencesToUse*

Set *unusedTestSentences* to value copy of (not a pointer) *POSSIBLE\_SENTENCES*

Loop *numberOfRounds* times

Append random sentence out of *unusedTestSentences* to *sentencesToUse*

Remove last item in *sentencesToUse* from *unusedTestSentences*

Display “Press enter to begin”

Prompt the user to press enter

Loop using int *i* to enumerate *numberOfRounds* times

Display “Round” + (*i* + 1) “/” + *numberOfRounds* + “. Your sentence is: ”

Display the *i*th object of *sentencesToUse*

Save the current timestamp as *tempTime*

Prompt user for input and save input as *playerInput*

Subtract *tempTime* from the current timestamp and then save that as *tempTime*

If *playerInput* is ‘x’ then

break

Call **calculate\_accuracy** pass the *i*th object of *sentencesToUse*, *playerInput*, and *strict*

Concatenate the return to the end of *accuracies*

Call **calculate\_wpm** pass the *i*th object of *sentencesToUse*, *tempTime*, the *i*th object of *accuracies*

Concatenate the return to the end of *playerWPMs*

Display “Accuracy: ” + the last object of *accuracies* as an integer

Display “WPM: ” + the last object of *playerWPMs* as an integer

Display “Press enter to get the next question”

Prompt the user to press enter and save input at *inp*

If *inp* is ‘x’ then break

Display “You have completed the test”

Display “Results: ”

If length *accuracies* of is 0 then

Display “You don’t have results.”

Display “You skipped the whole test”

Else

Set *maxAccuracy* the max value of *accuracies*

Set *averageAccuracy* to the sum of all the objects in *accuracies* divided by the length of *accuracies*

Display “Max Accuracy: ” + *maxAccuracy*

Display “Average Accuracy: ” + *averageAccuracy*

Set *maxPlayerWPM* to the max value of *playerWPMs*

Set *averagePlayerWPM* to the sum of *playerWPMs* divided by the length of *playerWPMs*

Display “Max WPM: ” + *maxPlayerWPM*

Display “Average WPM: ” + *averagePlayerWPM*

Display “Breakdown: ”

Display “Round \tAccuracy \tWPM”

Display “----- \t-------- \t---”

Loop using int *i* to enumerate *numberOfRounds* times

Display (*i* + 1)+ “\t” + the *i*th object of *accuracies* + “\t” + the *i*th object of *playerWPMs*

Display “Would you like to play again (y/n)”

Prompt the user to input a “y” or “n” save input as retry

Set retry to all lowercase

If retry is “n” or “no” then

Display “Goodbye then”

Display “Press Enter to close the program

Prompt the user to press Enter

Break

**Functions**

Function **calculate\_accuracy** parameters are *target\_sentence*, and *user\_input* and *strict*

If *strict* then

Set *target\_sentence* to all lowercase

Set *user\_input* to all lowercase

//TODO: when tutor tells us how robust this system needs to be

Return *accuracy*

Function **calculate\_wpm** parameters are *user\_input*, *time\_taken*, *accuracy*

Initialise list *user\_words* to *user\_input* that has been split into objects of a list every space

Initialise int *numberOfWords* as the length of *user\_words*

Initialise wpm as *numberOfWords* / (*time\_taken* / 60) \* accuracy / 100

Return *wpm* rounded to nearest int