Programming Development Lab:

Final Project Report:

1. Project Title: Typing Speed Test Application

2. Team members name and registration number:

Name: Ayush Tiwari

Registration number: 221080072

3. Problem Statement:

The goal of this project is to develop a typing speed test application using Python and Tkinter. The application should present users with a randomly selected sentence that they must type as quickly and accurately as possible. The program will calculate and display the user's typing speed, accuracy, and words per minute (WPM) upon completion of the test.

4. Motivation:

Typing speed is a crucial skill in today's digital age, especially for programmers and professionals who spend a significant amount of time working on computers. This typing speed test application aims to provide users with a fun and interactive way to assess and improve their typing skills. The motivation behind this project is to create a user-friendly tool that helps individuals enhance their productivity by enhancing their typing speed and accuracy.

5. Methodology:

Libraries Used:

Tkinter: Used for creating the graphical user interface (GUI) of the typing speed test application.

Time: Utilized for tracking the time taken by the user to complete the typing test.

Random: Employed to randomly select sentences for the typing test.

Application Flow:

1. **Initialization:**

   - The application initializes variables and sets up the GUI with labels, buttons, and text fields.

2. **Start Game:**

   - Clicking the "Start" button initiates the typing test.
   - A random sentence is selected and displayed on the GUI.
   - The timer starts when the user begins typing.

3. **Typing Input:**

   - As the user types, the application continuously checks the input against the correct sentence.
   - The application tracks when the user completes typing the sentence.

4. Calculation:

- Upon completion, the application calculates the total time taken, accuracy, and words per minute (WPM).
- The results are displayed on the GUI.

5. Reset Game:

- The "Reset" button allows the user to reset the test & start again.

6. Pseudo code with Output:

```
// Import the modules needed for the program
import tkinter, time, and random modules

// Define a list of sentences to use for the typing test
sentences = ["The quick brown fox jumps over the lazy dog."]

// Define a function that returns a random sentence from the list
function get_random_sentence
  return a random element from sentences
end function

// Define a function that returns the current time in seconds
function get_current_time
  return the time in seconds
end function
```

```
// Define a function that calculates the accuracy of the user's input by comparing it with the given word
function get_accuracy with parameters input_text and word
  // Split the input text and the word into words
  input_words = input_text split by spaces
  word_words = word split by spaces
  // Initialize a variable to store the number of correct words
  correct_words = 0
  // Loop through the minimum length of the input words and the word words
  for i from 0 to the minimum of the length of input_words and the length of word_words
    // If the words at the same index are equal, increment the correct words
    if input_words[i] is equal to word_words[i]
      correct_words = correct_words + 1
    end if
  end for
  // Return the percentage of correct words
  return (correct_words divided by the length of word_words) times 100
end function

// Define a function that calculates the words per minute (WPM) of the user's input by dividing the number of characters by 5 and dividing by the total time in minutes
function get_wpm with parameters input_text and total_time
  return (the length of input_text divided by 5) divided by (total_time divided by 60)
end function
```

// Define a class that represents the game logic and the GUI elements
class Game with parameter master
  // Define the constructor method that takes a master window as an argument
  method __init__ with parameter master
    // Assign the master window to an instance attribute
    self.master = window
    // Initialize some instance attributes to store the game state and data
    self.reset_flag = True // A flag to indicate whether the game is reset or not
    self.end = False // A flag to indicate whether the game is over or not

    // Set the title and the size of the master window
    self.master.title = "Typing Speed Test Application"
    self.master.geometry = "750x500"

    // Create a label widget for the heading and place it on the master window
    self.heading = a new label with text "Typing Speed Test Application", font ("Arial", 25), and foreground color 'black'
    self.heading.place at x = 150 and y = 10

    // Create a button widget for starting the game and place it on the master window
    self.start_button = a new button with text "Start", command self.start_game, width 15, and background color "light blue"
    self.start_button.place at x = 125 and y = 75

    // Create a button widget for resetting the game and place it on the master window

self.reset_button = a new button with text "Reset", command self.reset_game, width 15, and background color "light blue"
self.reset_button.place at x = 500 and y = 75

// Create a label widget for showing the results and place it on the master window
self.score_label = a new label with text 'Time:0 Accuracy:0 % WPM:0 ', font ("Arial", 20), and foreground color 'black'
self.score_label.place at x = 150 and y = 125

// Create a label widget for showing the instruction and place it on the master window
self.instruction_label = a new label with text "Instruction: Type the above sentence as fast and as accurately as you can:"
self.instruction_label.place at x = 75 and y = 300

// Create a text widget for showing the word to be typed and place it on the master window
self.sentence = a new text with height 3, width 60, and font ("Courier", 14)
self.sentence.place at x = 50 and y = 225

// Create a label widget for showing the timer and place it on the master window
self.timer_label = a new label with text "00:00:00", font ("Arial", 20), and foreground color 'red'
self.timer_label.place at x = 300 and y = 75

// Create a text widget for getting the user's input and place it on the master window
self.input = a new text with height 3, width 60
self.input.place at x = 50 and y = 325

```
    // Bind the key release event of the input widget to a function that
checks the user's input
    self.input.bind the event "<KeyRelease>" to the function self.check
  end method

  // Define a method that starts the game
  method start_game
    // If the game is reset
    if self.reset_flag is True
      // Set the reset flag to False
      self.reset_flag = False
      // Set the active flag to True
      self.active = True
      // Delete the contents of the sentence and input widgets
      self.sentence.delete from 1.0 to the end
      self.input.delete from 1.0 to the end
      // Get a random sentence and assign it to the word attribute
      self.word = get_random_sentence
      // Insert the word into the sentence widget
      self.sentence.insert at 1.0 the value self.word
      // Get the current time and assign it to the time start attribute
      self.time_start = get_current_time
      // Call the update timer method
      self.update_timer
    end if
  end method

  // Define a method that resets the game
  method reset_game
    // If the game is not reset
    if self.reset_flag is False
      // Set the reset flag to True
```

```
    self.reset_flag = True
    // Set the active flag to False
    self.active = False
    // Reset the game state and data attributes
    self.input_text = ''
    self.word = ''
    self.time_start = 0
    self.total_time = 0
    self.results = 'Time:0      Accuracy:0 %      WPM:0 '
    self.wpm = 0
    self.end = False
    // Delete the contents of the sentence and input widgets
    self.sentence.delete from 1.0 to the end
    self.input.delete from 1.0 to the end
    // Update the text of the score label
    self.score_label.config with text self.results
    // Update the text of the timer label
    self.timer_label.config with text "00:00:00"
  end if
 end method

 // Define a method that checks the user's input
 method check with parameter event
  // If the game is active and not over
  if self.active is True and self.end is False
   // Get the user's input from the input widget and strip any whitespace
   self.input_text = self.input.get from 1.0 to the end and strip whitespace
   // Set the end flag to True if the input text is equal to the word in length
```

```
    self.end = the length of self.input_text is equal to the length of self.word
    // Calculate the results of the game
    self.calculate
  end if
 end method

 // Define a method that calculates the results of the game
 method calculate
  // If the game is over
  if self.end is True
    // Get the current time and subtract the start time to get the total time
    self.total_time = get_current_time minus self.time_start
    // Calculate the accuracy of the user's input
    self.accuracy = get_accuracy with parameters self.input_text and self.word
    // Calculate the WPM of the user's input
    self.wpm = get_wpm with parameters self.input_text and self.total_time
    // Format the results as a string
    self.results = "Time:" + round self.total_time + " Accuracy:" + round self.accuracy + " % WPM:" + round self.wpm
    // Update the text of the score label
    self.score_label.config with text self.results
    // Delete the contents of the sentence widget
    self.sentence.delete from 1.0 to the end
    // Insert a message into the sentence widget
    self.sentence.insert at 1.0 the value "You have completed the test."
    // Set the active flag to False
    self.active = False
  end if
```

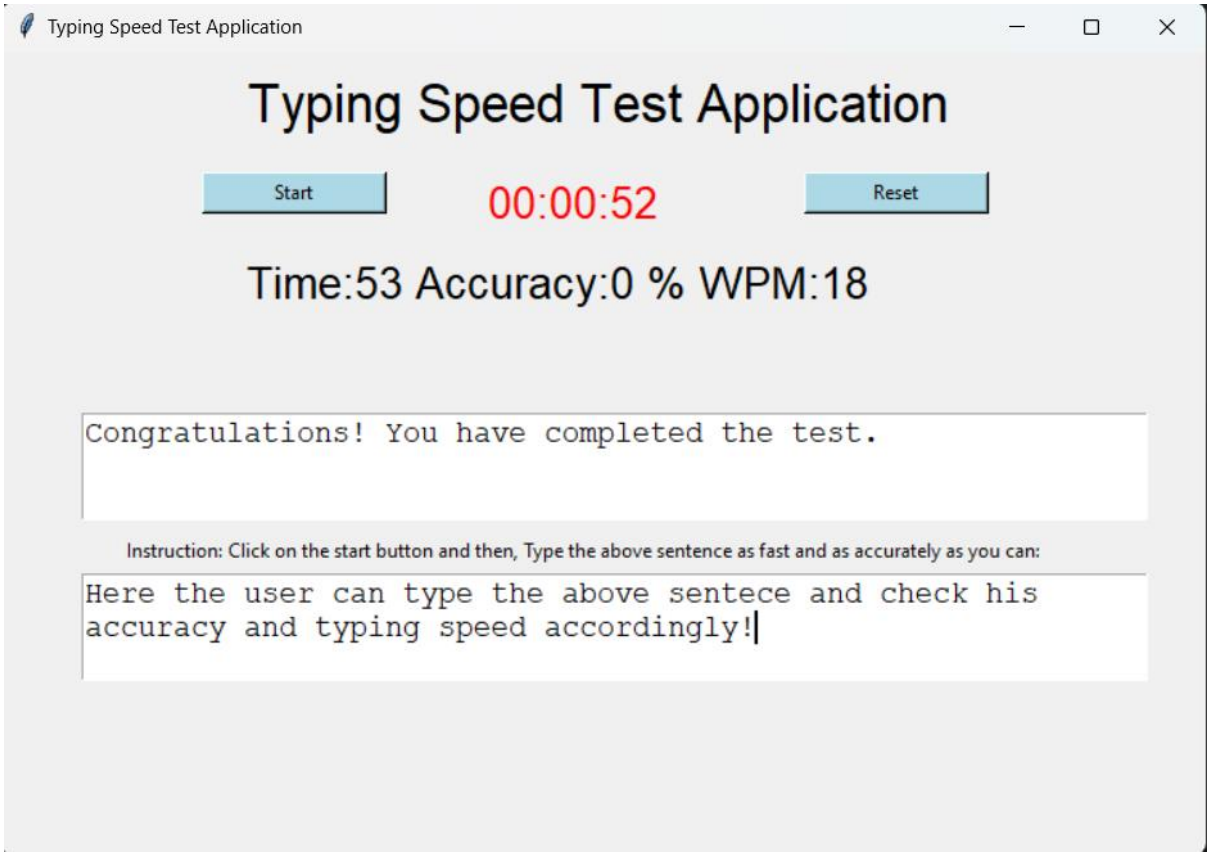
end method
end class

// Create a Tkinter window object
window = a new Tkinter window
// Create a game object with the window as the master
game = a new Game with parameter window
// Start the main loop of the window
window.mainloop

```
# Import necessary modules
import tkinter as tk
import time
import random

# Define a list of sentences for the typing test
sentences = [...]

# Define function to get a random sentence
function get_random_sentence():
    return random.choice(sentences)

# Define function to get current time in seconds
function get_current_time():
    return time.time()

# Define function to calculate accuracy of user's input
function get_accuracy(input_text, word):
    ...

# Define function to calculate words per minute (WPM)
function get_wpm(input_text, total_time):
    ...

# Define the Game class
class Game:
    # Constructor
    function __init__(self, master):
        ...

    # Method to start the game
    function start_game():
        ...
```

```
    # Method to reset the game
    function reset_game():
        ...

    # Method to check user's input
    function check(event):
        ...

    # Method to calculate game results
    function calculate():
        ...

    # Method to update the timer
    function update_timer():
        ...

# Create Tkinter window object
window = tk.Tk()
# Create a game object with the window as the master
game = Game(window)
# Start the main loop of the window
window.mainloop()
```

Output:



7. Discussion:

The Typing Speed Test application appropriately addresses the problem statement by providing users with an interactive tool to evaluate and enhance their typing skills. The GUI is designed to be user-friendly, with clear instructions and visual feedback. The application incorporates random sentence selection, time tracking, and accuracy calculation to provide comprehensive results to the user.

The use of Tkinter simplifies the creation of the graphical interface, making it accessible to users of all levels. The incorporation of features such as WPM and accuracy percentage adds depth to assessment, offering a more holistic view of the user's typing abilities.

The project's motivation stems from the recognition of the importance of typing skills in various professional fields. As technology continues to play a central role in our daily lives, improving typing speed and accuracy remains a relevant and valuable pursuit. The Typing Speed Test application aligns with this motivation by offering a practical and engaging solution for users to gauge and enhance their typing proficiency.

Therefore, concluding that the Typing Speed Test application serves as a useful tool for individuals seeking to refine their typing skills. Its straightforward design, coupled with accurate metrics and instant feedback, makes it an effective solution for both beginners and experienced typists. The project contributes to the broader goal of promoting efficiency and productivity in the digital age.