Rabisa Ali

Rabisa Report

Quick Submit

Quick Submit

National University of Computer and Emerging Sciences, Islamabad

Document Details

Submission ID
trn:oid:::1:3419709281

Submission Date
Nov 21, 2025, 3:29 PM GMT+5

Download Date
Nov 21, 2025, 3:32 PM GMT+5

File Name
Final_Project_Report_11.docx

File Size
565.8 KB

9 Pages

807 Words

4,398 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

0 AI-generated only 0%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Final Project Report — Programming Fundamentals

University Name: FAST (**National University of Computer & Emerging Sciences Karachi Campus)**

Department: Department of Computer Science

Course: Programming Fundamentals

Project Title: Rewordling Wordle

Submitted By: Rabisa Ali (25K-0876) Syeda Marium Fatima (25K-0970)

Submitted To: Ms.Kinza Mushtaq

Semester: Fall 2025

Date: November 21, 2025

Abstract

Rewordling wordle is a console-based game developed in C language to implement online wordle game efficiently. A player will be given ten consecutive Wordle puzzles. However, a single failed attempt to guess the word will lead to the termination of program. At the end, the program will calculate the average number of guesses, best guess, and a percentile ranking of pre-hard coded player performance data collected from google. This project applies fundamental programming concepts such as loops, arrays, and functions. This takes Wordle beyond a casual guessing game to something much more intense.

1. Introduction

Wordle is an online word puzzle game where a player has six attempts to guess a five-letter word. Feedback is given through colored tiles: green for a correct letter in the right spot, yellow for a correct letter in the wrong spot, and red for a letter not in the word at all. The goal is to guess the hidden word using this feedback within the limited attempts. Although there are numerous Wordle implementations online, most of them offer instant win or loss results without any in-depth analytical insights.

2. Objectives

- To find the hidden word using the feedback within the attempts.
- To find out the average number of tries it took the user to judge the word
- To find out the best guess
- To provide a percentile ranking
- To reinforce programming concepts like loops, arrays, and functions.
- To provide a simple, user-friendly interface.

3. System Design

System Overview

Flow of the program:

Start → Load the words into memory → Display instructions → Ask the user to enter the word → Compare the word with the original word and change the font color accordingly → If the user guessed the word correctly within the set number of tries, ask him if he want to continue. If yes then continue, else end the game → End the game by displaying average number of guesses in which the user was able to guess the word correctly, best guess, and his percentile ranking → Exit.

Algorithm

1. Start the program

2. Load the words into memory

3. Display instructions

4. Ask the user to enter the word

5. Compare the word with the original word and change the font color accordingly

6. If the user guessed the word correctly within the set number of tries, ask him if he want to continue. If yes then continue, else end the game.

7. End the game by displaying average number of guesses in which the user was able to guess the word correctly, best guess, and his percentile ranking.

8. End

Input & Output

Input: Guess of the user.

Output: Display instructions, feedback on each word, best guess, average number of tries and the percentile ranking.

4. Implementation

Language: C

Compiler/IDE: Code::Blocks / Dev C++ / GCC

Key Features

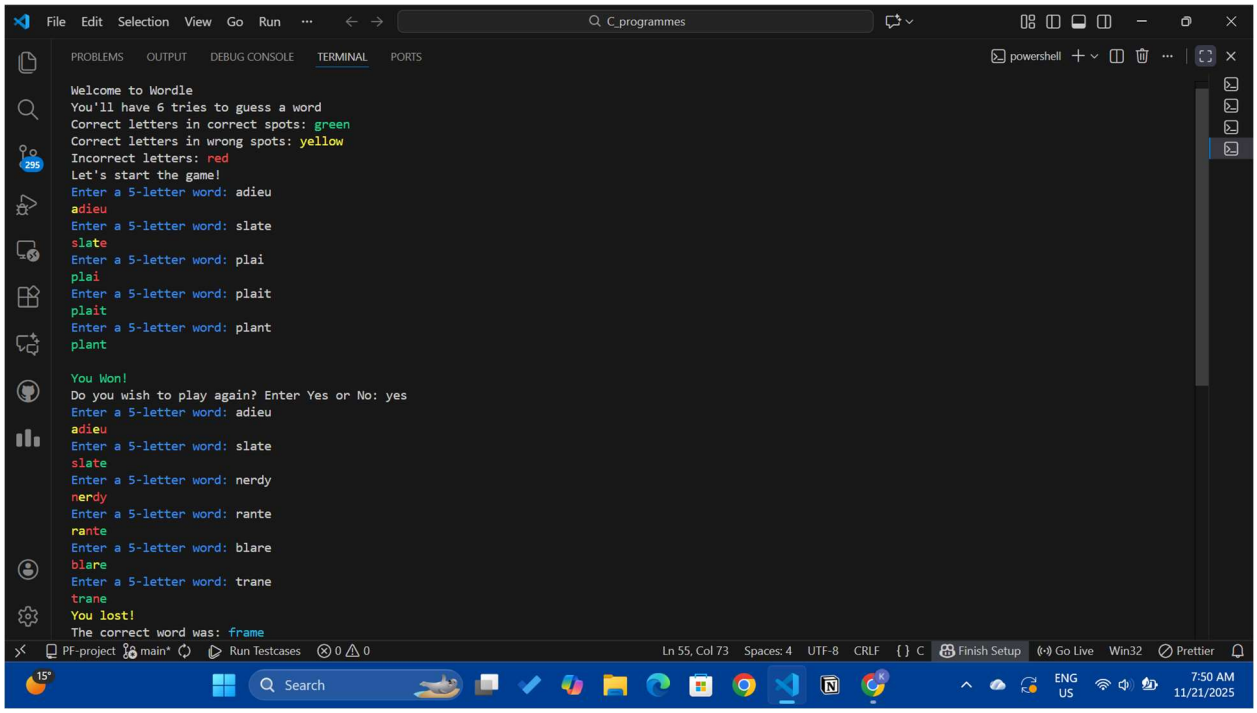
- Sequential gameplay of up to 10 word puzzles in a single session
- Immediate session termination upon first failed attempt
- Per-letter feedback system highlighting correct, misplaced, and incorrect letters
- Tracking and averaging of total guesses across solved puzzles
- Hardcoded percentile evaluation based on statistical benchmarks

Code Snippet

```
int processGuess(char *answer, char *guess) {
  int correct = 0;
  for (int i = 0; i < 5; i++) {
    if (guess[i] == answer[i]) {
      printf("\033[92m%c\033[0m", guess[i]);
      correct++;
    } else {
      int found = 0;
      for (int j = 0; j < 5; j++) {
        if (guess[i] == answer[j] && i != j) {
          found = 1;
          break;
        }
      }
      if (found)
        printf("\033[93m%c\033[0m", guess[i]);
      else
```

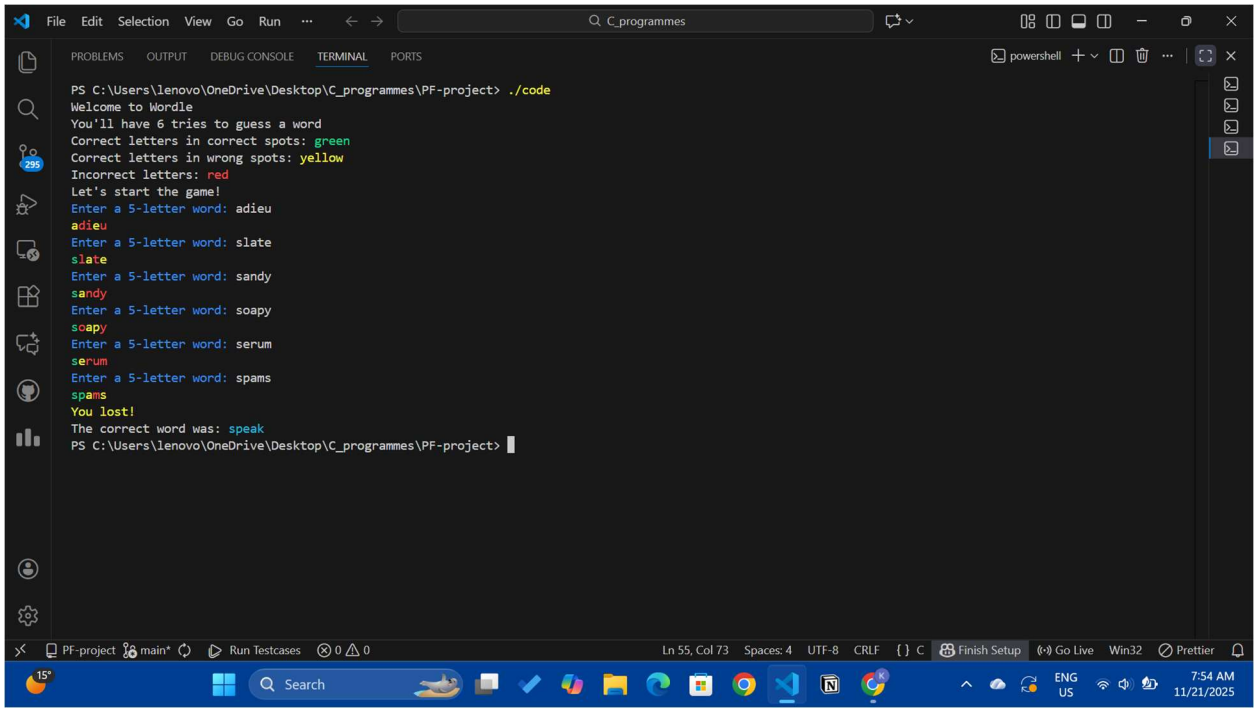
```
            printf("\033[91m%c\033[0m", guess[i]);
        }
    }
    printf("\n");
    return correct;
}

int game(char* answer, char* guess) {
    int guesses = 0;
    while (guesses < 6) {
        printf("\033[94mEnter a 5-letter word: \033[0m");
        scanf("%5s", guess);
        guesses++;
        if (processGuess(answer, guess) == 5) {
            return guesses;
        }
    }
    return 7;
}
```

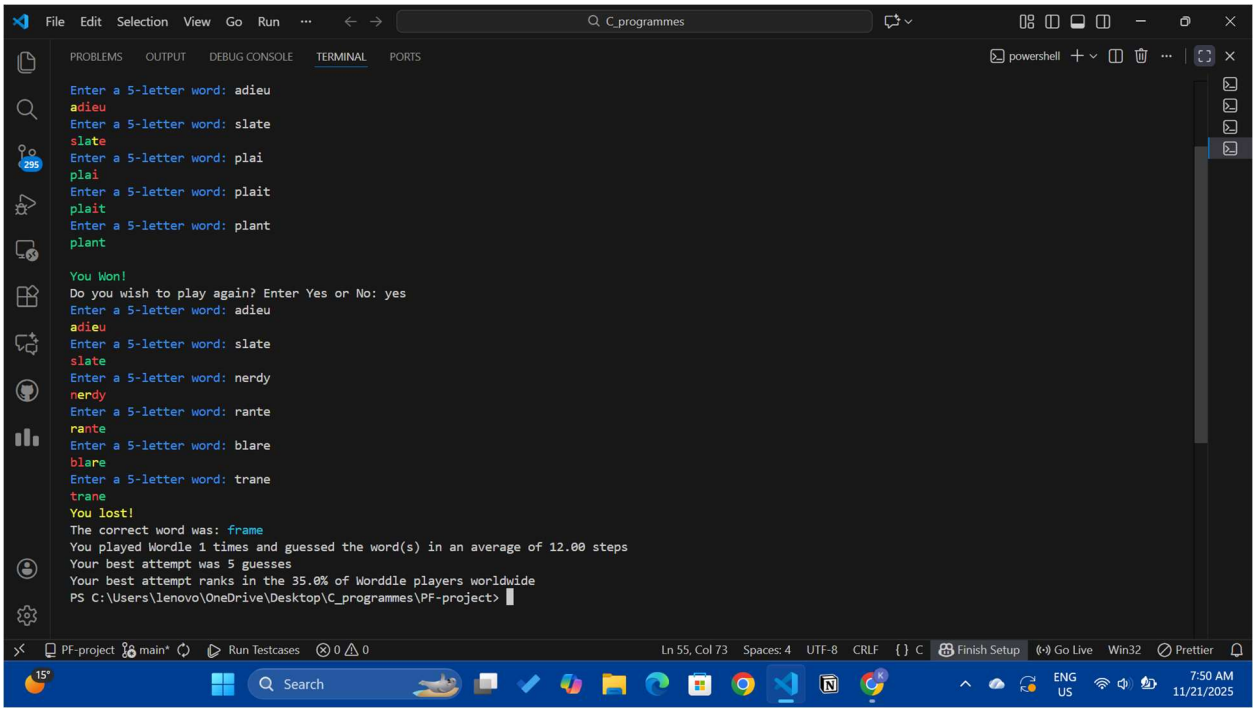
Sample Output:

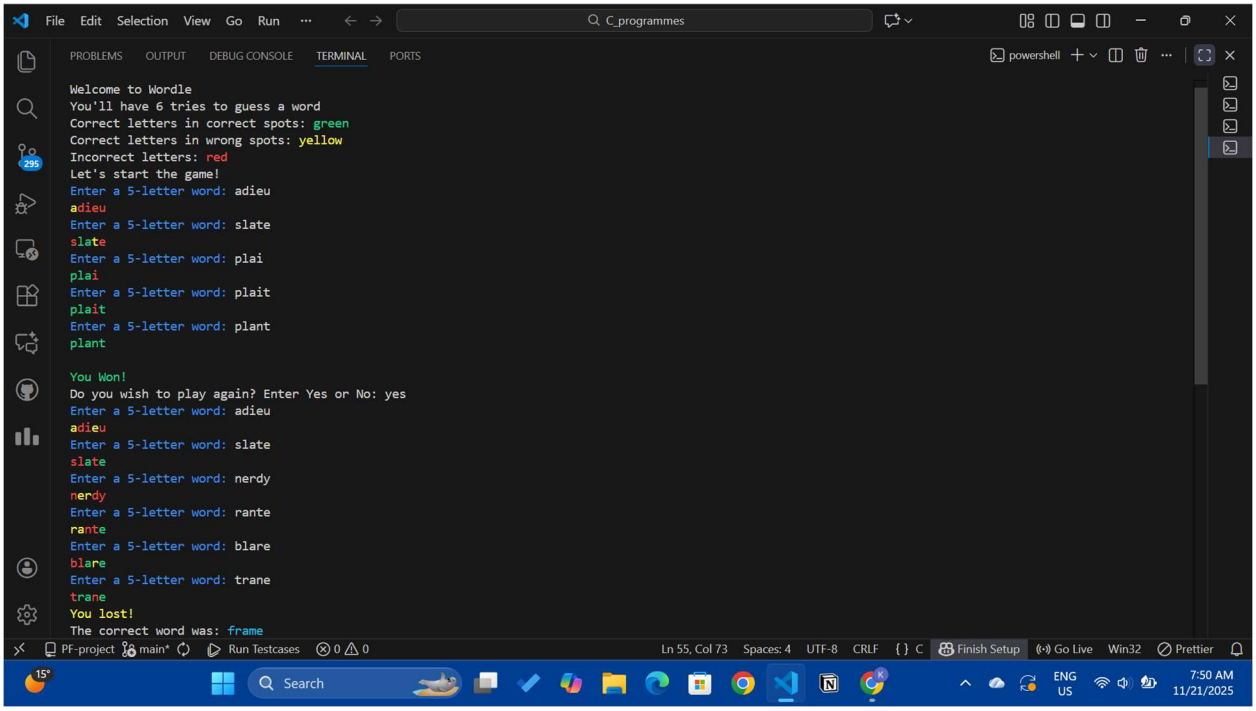


5. Testing and Results

```
Welcome to Wordle
You'll have 6 tries to guess a word
Correct letters in correct spots: green
Correct letters in wrong spots: yellow
Incorrect letters: red
Let's start the game!
Enter a 5-letter word: adieu
adieu
Enter a 5-letter word: slake
slake
Enter a 5-letter word: snake
snake
Enter a 5-letter word: soapy
soapy
Enter a 5-letter word: surey
surey
Enter a 5-letter word: seate
seate
You lost!
The correct word was: shaft
PS C:\Users\lenovo\OneDrive\Desktop\C_programmes\PF-project>
```







The program performed successfully for all test cases. It handled both correct and incorrect guesses efficiently, produced accurate best guess and percentile. Execution speed was near-instant, and the program required minimal system resources.

6. Conclusion, Limitations & References

Conclusion

The Rewordling Wordle successfully demonstrates the application of basic programming principles. It takes Wordle beyond a casual guessing game to measurable performance evaluation. The project strengthened understanding of arrays, loops, conditional statements, and functions.

Limitations

- Data of average number of guesses and best guess is lost when the program closes (no file handling yet).
- No graphical interface, purely console-based.

Future Enhancements

- Add file handling to store records permanently.
- Implement a graphical or web interface.

References

- Let Us C by Yashavant P. Kanetkar
- https://www.geeksforgeeks.org/c-programming-language/
- https://www.sportskeeda.com
- https://www.nytimes.com › wordle-bot-year-in-review