Word Guessing, Sub-sequences & Spell Checking

Assignment 1 – Milestone 2

2800ICT Object Oriented Programming

Bennett taylor

S5095512

Q1 - Word guessing game:

Problem Statement:

The goal of this project is to create a program that never loses the Word guessing game. The word guessing game works in a similar way to the game hangman, where one player chooses a secret word while the other player tries to guess it. The playerA with the secret word writes dashes equal to the secret word length. The other playerB starts guessing the word one letter at a time until the secret word is found, or they run out of guesses. Each time playerB guesses a correct letter, playerA must reveal each instance of that letter in the word. PlayerB wins when all the letters in the secret word have been revealed. PlayerA wins when playerB has run out of guesses. In this program the user is playerB and the computer is playerA. However, the computer really wants to win the game to prove it is smarter than humans. Rather than committing to a secret word, the computer cheats and compiles a list of every word in the English language that equals the established word length. When the user guesses a letter, the computer checks whether there are more words that include the guessed letter or more words that do not include the guessed letter. The computer will then reveal a letter or leave it blank based on this choice.

User Requirements:

- User will be able to undo their last guess at any time by typing “undo” into the terminal.
- User will be able to type “undo” as many times as they like, allowing them to go back to the start if they want to.
- The user will be able to keep guessing until they run out of guesses
- User will be able to ask the computer for a hint at any time by typing “hint” into the terminal. The hint given to the user will be as ambiguous as possible.
- The user will be displayed the hidden word in the form of dashes.
- When the user guesses a correct letter, its placement in the hidden word will be shown from until the game finishes.
- After each guess the user will be displayed how many guesses they have left and how many words are left.

- The user will be able to choose the length of the word and how many guesses they get. This will be done on the command line when compiling the program. Eg. ./word_guess 6 6
- The user will be displayed the hidden word at the end of the game if they did not guess the correct word.

Derived Requirements:

Instructions:

1. Unpack s5095512_Milestone1.zip
2. Open terminal of your choosing
3. Locate file directory in terminal Eg. H:/Uni/OOP
4. Compile the program by typing “g++ Q1.cpp -o Q1” in terminal
5. Run the compiled program with the word length and the amount of guesses you would like. Eg. ./Q1 6 6

Documentation:

Int compare(string s1, string s2, char letter)

This function check two strings to see if they have the same letter positioning. For example, if s1 = code, s2 = hole, letter = e, the program would return true since e is located in the same position in both words. This function is used to help build word categories.

Vector<string> remove_letter_words(vector<string> words, vector<string> letter words)

This function removes one list of words from another list of words. Used to divide the word list into two categories: one containing words with a certain letter and one not containing words of the same letter.

Vector<string> create_letter_list(vector<string> words, char letter)

This function creates a list of words containing a certain letter from the words given in the string vector words.

Vector<string> get_biggest_category(vector<string> words, char letter)

This function splits a word list into categories based on the letter given to it. It does this by creating a 2D string vector where each row contains a different word category. The function returns the biggest row contained within the 2D string vector.

String update(string word, string output_word, char letter)

This function updates the hidden word and reveals any letters the user might have guessed right. It does this by taking the first word from the biggest word category and checking where the letter is positioned. The function then writes the letter to the same position in the hidden word (output_word).

String downgrade(string output_word, char letter)

This function replaces a given letter in the hidden word with a “_”. This function is called when the user types “undo” into the terminal. For example, if the user guessed the correct letter “e” then typed “undo”. The function replace the letter “e” with “_”, technically making it hidden again.

Int found(string final_word)

This function checks whether there are any dashes left in the hidden word.

Vector<string> get_words (int word_length)

This function reads the file "Dictionary.txt" line by line and returns a list of words matching the word length that the user defined.

String create_dashs(int word_length)

This function creates a string of dashes equal the word length the user has defined then returns it.

Void get_hint_from_file(char letter)

This function searches the file "Riddles.txt" for a hint to display to the user about a given letter. The file "Riddles.txt" contains one riddle for each letter of the alphabet. All the riddles where written by me to make the game more fun/challenging while still providing "the most ambiguous hint possible".

Void hint(vector<string> words, vector<char> prev_letters)

This function provides a hint to the user based on the most common letter or least common letter found in the word list. The function first creates a list for the most common and least common letter. It then check which of the two is bigger and calls get_hint_from_file(). The idea behind this method is to still give the user the worst possible odds. If the program only gave hints of what the next letter will be, it would be bettering the odds for the user. For example, say the most common letter in the word list is "e" with 500 words out of 1000. The least common letter is "w" with 50 words out of 1000. The program will always try to make it harder for the user so it will say the hidden word does not contain "w". The user will then still have 950 words to guess out of compared to 500 words. The function get_hint_from_file() makes it harder again for the user because they have to guess that "w" is the answer of the riddle.

Int word_guessing(int word_length, int guesses)

This function is the center of the whole program. The function first creates a word list from dictionary.txt by calling get_words(). It then enters a loop which runs until the user's guesses equal 0. The function then prompts the user for an input. The only valid inputs are letters, "hint" and "undo". Any other input by the user will result in them having to try again. Once the user enters a valid letter input, the function splits the word list into words containing that letter and words that don't contain that letter. It splits the letter containing list even further by finding the biggest word category. The function then checks which of the two list is bigger. If the list containing the letter is bigger, it reveals that letter in the hidden word and continues with letter list becoming the new word list. If the list without the letter is bigger, then the function will not reveal anything about the hidden word and continue with the non letter list becoming the new word list.

Testing:

Results:

Q2 - Word Sub-Sequences/Spell checker:

Problem Statement:

The goal of this project is to create a program that can perform two actions of one file. Those actions are:

1. Read the file and list the top 10 occurrences of sub-sequences of 2, 3, 4, and 5 letters with no consecutive repeating letters.
2. Allow the user to enter a word and if it is not in the dictionary then display a list of words that are "close" to this word

The file being used for these two actions is "Dictionary.txt"

User Requirements:

- User will be able to run the program within the terminal.
- User will be able to give command line arguments while executing the program.
- The user will be able to type any word of any length when asked for a word to spell check.
- The user will be displayed the top 10 occurrences of sub-sequences in the terminal.

Documentation:

Vector<string> read_file()

This function reads the file "Dictionary.txt", adds its contents to a list called words and returns that list.

Int check_lettering(string user_input, string word)

This function checks how much two different words differ in lettering. For example, if the two words being check are "choker" and "choked". The function would return 1 because there is only one letter different in the two words.

Vector<string> read_file()

This function reads the file "Dictionary.txt", adds its contents to a list called words and returns that list.

Vector<vector<string>> add_to_close_words(vector<vector<string>> close_words, string file_word, string user_input)

This function is called when a new word is read from Dictionary.txt. This function takes that new word and inserts it in a 2D string vector ranked by letter difference.