Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2022

Homework 1 – SWordle

Due: 15/03/2022, Tuesday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism and homework trading will not be tolerated!

Introduction

In this homework, you will implement Sabancı version of the world-wide internet sensation "Wordle". The regular game challenges the users to guess a word, and if they can guess the correct word in six attempts or less, they win. Every time the user guesses a word, the game will give hints about the correctness of the guess. When a user guesses a word, the letters of the guessed word will be colored with one of three colors: green, yellow, and black, and each of these colors has different meanings. For each letter in the guessed word, if the letter falls in the same position in the correct word, it will be colored green. If the letter is in the correct word, but is in the wrong position, it will be colored yellow. Finally, if the letter does not occur at all in the correct word, it will be colored black. Here's an example. Let's assume the word the user needs to guess is "trade". First, user tries the word "ready" as shown in Figure 1.

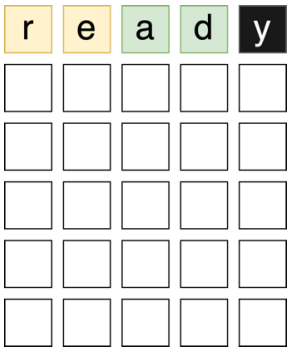

Figure 1: User entered "ready" when the correct word is "trade".

As you can see in Figure 1, the letter “y” is colored black because it does not occur in the word “trade”. The letters “r” and “e” are colored yellow because they occur in “trade”, but in wrong positions. Finally, the letters “a” and “d” are in the correct positions, so they are colored green.

Next, the user tries “grade” (Figure 2). Only one letter is wrong!

Figure 2: user entered “grade” when the correct word is “trade”.

Finally, the user tries “trade” and finds the correct word, it is a win (Figure 3)!

Figure 3: user entered “trade” which is the correct word.

The game that you will build will be different than the regular Wordle, so we rename it as SWordle (Sabancı version of Wordle). Mainly, there are four differences:

1. You will be able to specify the number of attempts that the user can try; it could be more or less than 6 attempts.
2. The words that the user will guess don’t need to be 5 letters long. It can be shorter or longer.
3. There are some extra input checks for the guesses that regular Wordle does not have (see below).
4. Regular Wordle enforces to use a meaningful word as the guess, but in SWordle, we do not have such a rule.

Inputs, Outputs, Matrix Uage and Program Flow

First, the user will be prompted for the name of a txt file. If the file with the given name cannot be opened, user will be prompted again until a file can successfully be opened. Your program should read this txt file. First line contains one integer that specifies the number of attempts the user is allowed to try. Second line of the file contains the word to be guessed. After reading the file, you should display the number of letters in the secret word and the total number of attempts.

You will use a 2D matrix to represent the current state of the user's attempts. The matrix will be made up of cell structs, of which the definition is given below:

```
struct cell
{
    char letter; // letter inside the cell
    char color; // color of the cell
};
```

The `letter` data member will contain the letter in the cell, and the `color` data member will contain the characters '-', 'G', 'Y', or 'B' depending on the color of the cell, no color (representing unused guess), green, yellow, and black, respectively.

This matrix **must be a 2D vector** (i.e. a `vector` of `vector` of `cell` structs). It will have one row for each attempt the user is allowed, and it will have as many columns as the number of letters in the correct word. Initially, all the cells must have the letter and color members as '-' as shown in Figure 4 (for an example case where the correct word has 4 letters and the number of attempts is 3).

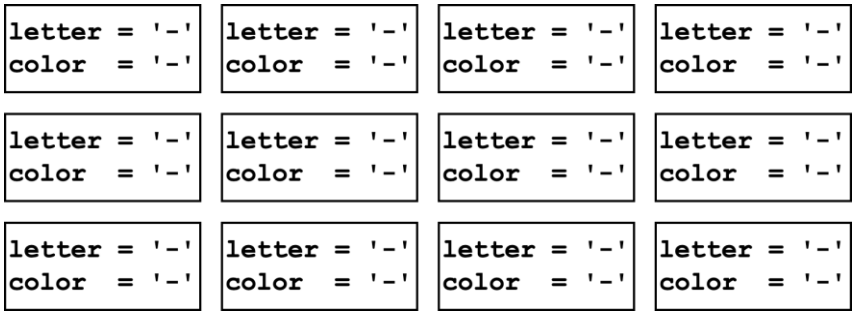

Figure 4: The initial state of an example matrix for a 4-letter word and 3 attempts.

Once the matrix is created, your program should start asking the user for guesses. The user will input the words using the keyboard. Every time the user inputs a word, your code will do some input checks (which will be discussed in the next section). If the user enters a guess that fails any of the input checks, you must ask the user for another input for the same attempt. You will keep asking the user for input until it passes the input checks. Once the input passes the checks, you will add the word to the matrix, determine the success of the guess (i.e. the colors), print out the status of the matrix, then check if the game is over.

To demonstrate how you will fill the matrix, let's look at an example. Let's assume that in this example, the user is allowed three attempts, and that the correct word is "play". Figure 5 shows what the matrix will look like after the user tries the word "pale".

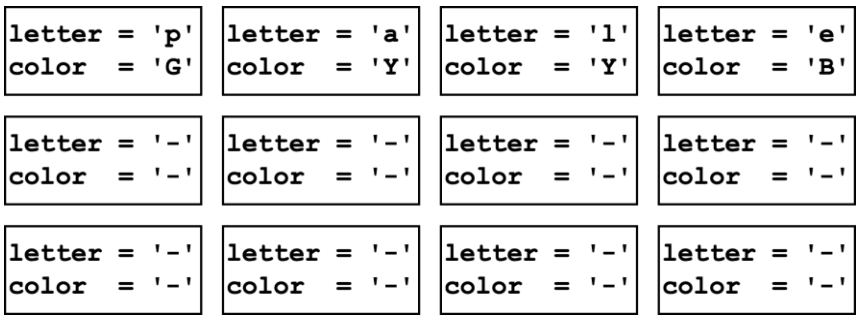

Figure 5: The letters and colors of the matrix after the user enters "pale" when the correct word is "play".

The letter 'p' is green since it's in the correct position. The letters 'a' and 'l' are yellow since they occur in the correct word but in different positions. The letter 'e' is black since it does not occur in the correct word at all.

After every attempt by the user, your program must print out the current state of the matrix. For a sample format of printing, please check the sample runs.

Your code will keep asking the user for words until either the user guesses the correct word (winning case) or he/she runs out of attempts (losing case). Once one of these two outcomes happens, the program will end execution.

Input Checks

Before the input checks, let us give the assumptions first. You may assume that there will not be any empty line in the text file and there are exactly two lines in it. Moreover, the number of allowed attempts is assumed to be given as a positive integer. Also, the word in the text file is assumed to contain only lowercase letters (no numbers, special characters, uppercase letters, etc.). Moreover, we also assume that the word in the text file does not contain any repeated letters. All of these assumptions are guaranteed and you do not need to check them.

However, there are some input checks to be done by your program. These are for the user guesses that you take as input. Specifically, when the user enters a word as a guess, you must check the following conditions in this exact order:

1. The number of letters in the user's entered word matches that of the correct word.
2. The characters of the word are all lowercase letters.
3. There are no repeated letters in the word.
4. The word has not been tried previously.
5. If, a particular letter was green in one of the user's previous attempts, then the same letter must be in the newly entered word and in the same position. In other words, once you correctly guess a letter, you have to use that letter all the time correctly for the upcoming trials. For example, in the example shown in Figure 5, if the user tries to enter the word "liar" as the second guess, then this check fails. This is because the first letter of "liar" is not "p".

If any of these input checks fail, you will ask the user for another input for the same attempt. You will keep asking the user for inputs until he/she enters a word that passes all checks. Please note that these checks will be done sequentially and if any of them fails, an appropriate error message will be printed (see sample runs) and others will not be checked for that input. In other words, if there are two or more input check

problems of an entry, only one of them (the one encountered first in the above list) will be printed as error. For example, if an input has more letters than the correct word and has repeated letters, you must only print an error message stating that the word has too many letters. As another example, if an input has correct amount of characters, but has repeated letters and non-lowercase characters, the error message regarding the non-lowercase letters will be displayed only.

Sample Runs

Some sample runs are given below, but these are not comprehensive; therefore, you must consider **all possible cases** to get full mark.

You do not need to give the output exactly as in the sample runs, provided that your output is understandable and clear.

Sample Run 1:

Contents of the file "word1.txt":

```
5
trade
```

Console output (input is in ***bold italics***):

```
Welcome to SWordle!
Please enter the file name: word1.txt
The word that you will guess has 5 letters and you have 5 attempts.
What's your guess? ready
|| r , Y || e , Y || a , G || d , G || y , B ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
What's your guess? grade
|| r , Y || e , Y || a , G || d , G || y , B ||
|| g , B || r , G || a , G || d , G || e , G ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
What's your guess? trade
|| r , Y || e , Y || a , G || d , G || y , B ||
|| g , B || r , G || a , G || d , G || e , G ||
|| t , G || r , G || a , G || d , G || e , G ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
You win!
```

Sample Run 2:

Contents of the file “word2.txt”:

```
3
saw
```

Console output (input is in ***bold italics***):

```
Welcome to SWordle!
Please enter the file name: word2.txt
The word that you will guess has 3 letters and you have 3 attempts.
What's your guess? long
The word is too long!
What's your guess? toolong
The word is too long!
What's your guess? sh
The word is too short!
What's your guess? new
|| n , B || e , B || w , G ||
|| - , - || - , - || - , - ||
|| - , - || - , - || - , - ||
What's your guess? que
You did not use the green letter in your word!
What's your guess? wow
Your input has a duplicate letter!
What's your guess? woo
Your input has a duplicate letter!
What's your guess? won
You did not use the green letter in your word!
What's your guess? new
You've already tried this word!
What's your guess? now
|| n , B || e , B || w , G ||
|| n , B || o , B || w , G ||
|| - , - || - , - || - , - ||
What's your guess? saw
|| n , B || e , B || w , G ||
|| n , B || o , B || w , G ||
|| s , G || a , G || w , G ||
You win!
```

Sample Run 3:

Contents of the file “word3.txt”:

```
1
knit
```

Console output (input is in ***bold italics***):

```
Welcome to SWordle!
Please enter the file name: word3.txt
The word that you will guess has 4 letters and you have 1 attempts.
What's your guess? ball
Your input has a duplicate letter!
What's your guess? grit
|| g , B || r , B || i , G || t , G ||
You lose!
```

Sample Run 4:

Contents of the file “word4.txt”:

```
4
ice
```

Console output (input is in ***bold italics***):

```
Welcome to SWordle!
Please enter the file name: word4.txt
The word that you will guess has 3 letters and you have 4 attempts.
What's your guess? lie
|| l , B || i , Y || e , G ||
|| - , - || - , - || - , - ||
|| - , - || - , - || - , - ||
|| - , - || - , - || - , - ||
What's your guess? bye
|| l , B || i , Y || e , G ||
|| b , B || y , B || e , G ||
|| - , - || - , - || - , - ||
|| - , - || - , - || - , - ||
What's your guess? ion
You did not use the green letter in your word!
What's your guess? lie
You've already tried this word!
What's your guess? die
|| l , B || i , Y || e , G ||
|| b , B || y , B || e , G ||
|| d , B || i , Y || e , G ||
|| - , - || - , - || - , - ||
What's your guess? tie
|| l , B || i , Y || e , G ||
|| b , B || y , B || e , G ||
|| d , B || i , Y || e , G ||
|| t , B || i , Y || e , G ||
You lose!
```

Sample Run 5:

Contents of the file “word5.txt”:

```
5
knife
```

Console output (input is in ***bold italics***):

```
Welcome to SWordle!
Please enter the file name: word5.txt
The word that you will guess has 5 letters and you have 5 attempts.
What's your guess? 51ler
Your input has illegal letters!
What's your guess? ready
|| r , B || e , Y || a , B || d , B || y , B ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
What's your guess? KNIFE
Your input has illegal letters!
What's your guess? KniFE
Your input has illegal letters!
What's your guess? knife
|| r , B || e , Y || a , B || d , B || y , B ||
|| k , G || n , G || i , G || f , G || e , G ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - || - , - ||
You win!
```

Sample Run 6:

Contents of the file “word6.txt”:

```
3
h
```

Console output (input is in ***bold italics***):

```
Welcome to SWordle!
Please enter the file name: wrong_file.txt
Couldn't find the file!
Please enter the file name: not_word6.txt
Couldn't find the file!
Please enter the file name: word6.txt
The word that you will guess has 1 letters and you have 3 attempts.
What's your guess? a
|| a , B ||
|| - , - ||
|| - , - ||
What's your guess? b
|| a , B ||
|| b , B ||
|| - , - ||
What's your guess? c
|| a , B ||
|| b , B ||
|| c , B ||
You lose!
```

Sample Run 7:

Contents of the file “word7.txt”:

```
4
when
```

Console output (input is in ***bold italics***):

```
Welcome to SWordle!
Please enter the file name: word9.txt
Couldn't find the file!
Please enter the file name: word7.txt
The word that you will guess has 4 letters and you have 4 attempts.
What's your guess? wedr
|| w , G || e , Y || d , B || r , B ||
|| - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - ||
What's your guess? wtec
|| w , G || e , Y || d , B || r , B ||
|| w , G || t , B || e , G || c , B ||
|| - , - || - , - || - , - || - , - ||
|| - , - || - , - || - , - || - , - ||
What's your guess? wheq
|| w , G || e , Y || d , B || r , B ||
|| w , G || t , B || e , G || c , B ||
|| w , G || h , G || e , G || q , B ||
|| - , - || - , - || - , - || - , - ||
What's your guess? when
|| w , G || e , Y || d , B || r , B ||
|| w , G || t , B || e , G || c , B ||
|| w , G || h , G || e , G || q , B ||
|| w , G || h , G || e , G || n , G ||
You win!
```

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homework, we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**. Of course, your program should work in *Debug* mode as well.

Please do not use any non-ASCII characters (Turkish or other) in your code.

You are allowed to use sample codes shared with the class by the instructor and TAs. However, you cannot start with an existing .cpp or .h file directly and update it; you have start with an empty file. Only the necessary parts of the shared code files can be used and these parts must be clearly marked in your homework by putting comments like the following. Even if you take a piece of code and update it slightly, you have to put a similar marking (by adding "`and updated`" to the comments below.

```
/* Begin: code taken from lab1.cpp */
```

...

```
/* End: code taken from lab1.cpp */
```

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio C++. We recommend using 2012 version; however, if you use another version provided by Sabancı University software repository, it is OK as long as you specify which version you use at the beginning of your program. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course). Using other platforms (Xcode, VSCode, etc.) is risky and may cause some incompatibility problems.

Submissions guidelines are below. Some parts of the grading process might be automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Name your cpp file that contains your main program using the following convention:

“SUCourseUserName_YourLastname_YourNames_HWnumber.cpp”

Your SUCourse user name is your SUNet user name which is used for checking sabanciuniv e-mails (not the numeric one). Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw1.cpp

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case, add informative phrases after the hw number. However, do not add any other character or phrase to

the file names. Sometimes, you may want to use some user defined libraries (such as strutils of Tapestry); in such cases, you have to provide the necessary .cpp and .h files of them as well. If you use standard C++ libraries, you do not need to provide extra files for them.

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case, use the same filename format but add informative phrases after the hw number (e.g. Cago_Ozbugsizkodyazaroglu_Caglayan_hw1_myfuncs.cpp or Cago_Ozbugsizkodyazaroglu_Caglayan_hw1_myfuncs.h). However, do not add any other character or phrase to the file names. Sometimes, you may want to use some user defined libraries (such as strutils of Tapestry); in such cases, you have to provide the necessary .cpp and .h files of them as well by using the same naming convention mentioned above. If you use standard C++ libraries, you do not need to provide extra files for them.

You will receive zero if your compressed zip file does not expand or it does not contain the correct files. The naming convention of the zip file is the same. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourNames_HWnumber.zip

For example, zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

Hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Albert Levi, Amro Alabsi Aljundi