COMP 1602 – Computer Programming II
Assignment #2
Date Due: March 17, 2024 at 11:55 pm

Overview

This assignment requires you to write a program to simulate the Bonus Round of the television game show, "Wheel of Fortune". The player must choose one of three categories. A puzzle is then selected from that category and presented to the player. If the puzzle contains 'R', 'S', 'T', 'L', 'N', or 'E', these letters are revealed. The player must then choose three consonants and a vowel. If the player entered the Bonus Round with a wildcard, an additional consonant must be chosen. The consonants and vowel chosen by the player are inserted in the puzzle if they are present and the player must then guess the puzzle.

In the television show, the player is given ten seconds to guess the puzzle. In this assignment, there is no time constraint, and the player is simply given three tries to guess the puzzle. If the player correctly guesses the puzzle, he/she wins a prize (typically, a car, $40,000.00, or the jackpot of $100,000.00).

Struct Declaration

Your program must use the following struct to store the data on each puzzle. You may use other structs if you wish.

```
struct Puzzle {
	string category;
	char puzzle [80];
};
```

Description

1. Reading Categories

When your program starts, it must read the categories of puzzles from the text file, `Categories.txt`, and store them in an array of type ***string***. A category can consist of one or more words. There are no more than 50 categories of puzzles. Here are four lines from the file:

```
In the Kitchen
Living Thing
Movie Title
Occupation
```

2. Reading Puzzles

After reading the puzzle categories, your program must read the puzzles from the text file, `WOF-Puzzles.txt`, and store them in an array of *Puzzle* structs. The data for one puzzle must be read from **two** lines in the data file. The first line is the category of the puzzle, and the second line is the actual puzzle. (See Question 7 from Lab #6.)

Because of the need to maintain the secrecy of the puzzles stored in the file, the second line has been encrypted using the *Shift-5* algorithm. The Shift-5 algorithm is described on Page 3. Puzzles are stored in their encrypted form in the *Puzzle* structs.

Here are the first four lines from the `WOF-Puzzles.txt` file:

Encrypted	After Decryption
Around the House	Around the House
KQZKKD UNQQTBX	FLUFFY PILLOWS
Around the House	Around the House
KZQQ-QJSLYM BFQQ RNWWTW	FULL-LENGTH WALL MIRROR

The first puzzle is "FLUFFY PILLOWS" (after it is decrypted). It is in the category, "Around the House". The second puzzle, "FULL-LENGTH WALL MIRROR" (after it is decrypted) is in the same category. Puzzles are stored in uppercase letters.

3. Selecting and Displaying Three Categories

Your program should randomly select three categories of puzzles from the array of categories and present them to the player. The player must then choose from one of the categories. This category will be used to randomly select a puzzle in the next stage.

4. Selecting and Displaying Puzzle

After the player chooses a category, your program must randomly select a puzzle in that category from the array of *Puzzle* structs. Since a puzzle in **any** category can be randomly selected, it is important to repeatedly generate random numbers until a puzzle in the desired category is found.

After selecting the puzzle, it must be decrypted using the Shift-5 decryption algorithm. It is then displayed to the player with the letters "blanked off". The character '#' is used to hide the letters. If there are spaces or dashes ('-') in the puzzle, these are revealed to the player, for example, the puzzle "FULL-LENGTH WALL MIRROR" would be displayed as follows:

```
####-###### #### ######
```

5. Revealing the Letters (R, S, T, L, N, E)

After the puzzle is displayed, it is checked to determine if the letters 'R', 'S', 'T', 'L', 'N', or 'E' are present in the puzzle. If so, **every** occurrence of these letters is revealed. So, the puzzle "FULL-LENGTH WALL MIRROR" would now be displayed as follows:

```
##LL-LEN#T# ##LL ##RR#R
```

6. Letting the Player Choose 3 Consonants and 1 Vowel

The player is now given a chance to choose 3 consonants and a vowel. After the player chooses these four letters, an input statement should be written to find out if the player has the wildcard. If so, the player can choose a fourth consonant.

If the letters chosen by the player are present in the puzzle, they are revealed. Suppose that the player chose 'G', 'B', 'C' and '0' and that they did not have the wildcard. The puzzle would now be displayed as follows (since only 'G' and 'O' were present in the puzzle):

```
##LL-LENGT# ##LL ##RROR
```

7. Giving the Player Three Attempts to Guess the Puzzle

At this stage in the game, the player is given three attempts to guess the puzzle. If after three attempts the player did not guess the puzzle, an appropriate message is displayed, and the game comes to an end. If the player correctly guessed the puzzle, a prize must be randomly selected for the player. The prize possibilities are shown in the following table:

Prize	Probability
Car	3/25
$40,000.00	19/25
$45,000.00	2/25
Jackpot ($100,000.00)	1/25

The game comes to an end after the prize is given to the player.

The Shift-5 Encryption Algorithm

The Shift-5 encryption algorithm works by adding 5 to the ASCII code of each letter. Non-letters are not affected by the algorithm. So, 'A' will be encrypted to 'F', 'B' will be encrypted to 'G', 'C' will be encrypted to 'H', and so on.

If after adding 5, the resulting ASCII code is greater than 'Z', the counting continues from 'A'. So, 'U' will be encrypted to 'Z', 'V' will be encrypted to 'A', 'W' will be encrypted to 'B', and so on.

The decryption process works in the reverse manner by subtracting 5 from the ASCII code of each letter. So, 'F' will be decrypted to 'A', 'G' will be decrypted to 'B', 'H' will be decrypted to 'C', and so on.

If after subtracting 5, the resulting ASCII code is less than 'A', the subtraction continues from 'Z'. So, 'A' will be decrypted to 'V', 'B' will be decrypted to 'W', 'C' will be decrypted to 'X', and so on.

Testing Your Program

To get the same results shown on the *Sample Output* document, you **must** initialize the random number generator with the value 100. You must also enter the same input when requested to do so.

After you get your program to work, you can initialize the random number generator in the normal way using "srand(time(NULL))". This will cause different random numbers to be generate each time the program is executed.

What to Submit

Name your program `Assignment2.cpp` and upload it on myeLearning or before the deadline date. No other file must be uploaded.

Programming Guidelines

1. Suppose that *str1* is a C-string and *str2* is a variable of type *string*. The characters in *str1* can be copied to *str2* using an assignment statement. For example,

```
char str1 [80] = "Hello there!";
string str2;

str2 = str1;
```

2. Suppose that *inputFile* is an *ifstream*. To read **all** the characters from **one** line and store them in a C-string, use the following statements:

```
char line[80];
inputFile.getline(line, 80);
```

3. Use a C-string to read the categories and puzzles from the data files. To copy a category from a C-string to a variable of type *string*, use the code from (1) above.

4. To read **all** the characters on a line typed by the user at the keyboard (needed in the last stage of the game), use the following statements:

```
char line[80];
cin.getline(line, 80);
```

   You should use the following statement before the loop which prompts the user to enter his/her guesses:

```
cin.ignore(INT_MAX, '\n');
```

5. In Stage 3, "Selecting and Displaying Three Categories", the three categories should be randomly selected from the array of categories. It should be noted that the second category chosen could be the same as the first one. Also, the third category chosen could be the same as the previous two. So, code must be written to ensure that the three categories selected are different.

6. You don't have to cater for errors such as incorrect input by the player (including the choosing of letters already revealed). However, you should cater for letters and guesses being typed in either lowercase or uppercase.

7. You can use any of the C-string functions discussed during the course lectures, as well as well as the built-in C-string and *string* functions (e.g., *strlen*, *strcmp*, *strcpy*, etc.).

8. You are free to write your own functions for this assignment. The following table contains a list of recommended functions. Your program should contain at least 5 functions which perform useful tasks such as those listed in the table.

Function	**Description**
readCategories	Reads the categories from the file, `Categories.txt`, and stores them in an array of type *string*.
readPuzzles	Reads the puzzles from the file, `WOF-Puzzles.txt`, and stores them in an array of *Puzzle* structs.
chooseCategory	Randomly selects three categories, presents them to the player, and gets the player's choice.
choosePuzzle	Randomly selects a puzzle from the category chosen by the player.
decryptPuzzle	Decrypts a puzzle using the reverse Shift-5 algorithm.
blankOutPuzzle	Create a copy of the puzzle (for display purposes), blanking out all letters except spaces and dashes.
checkWildcard	Checks if the player has the wildcard.
revealLetter	If a letter is present in the puzzle, reveals the letter in the puzzle displayed.
rstlne	Reveals the letters 'R', 'S', 'T', 'L', 'N', or 'E' if they are present in the puzzle.
revealPrize	Reveals the prize to be awarded to the player if the puzzle was correctly guessed.
toUpperCaseStr	Converts all the lowercase letters in a string to uppercase.

****** End of Assignment Description ******