Lab 0: Zero to Sixty

CS5123 Advanced Programming and Data Sturctures

Part 1 Description: In this lab you will devise a simple guessing game that will allow the user input a guess for a number from 1 to 100. Your program will compare the guess with a randomly generated number from 1

Code Bits: Use cout and cin for the output and input (remember to include iostream!) and use while loops for the control with if-else's for the decision on the guess. Once you have it running for a single turn, add a counter for the number of guesses it took the user to find the secret number and display that in the results. Also, use a do-while loop to ask the user if they want to play again. Note: for the random number generation, please include the following lines (you will need to include cstdlib and ctime):

```
srand ( time(NULL));

{your secret number} = rand() % 100 + 1;
```

Hint: When you have them playing multiple times, be careful about the variables!

Additional Objectives:

1. Let them use upper or lower case for their answer to play again (check both).
2. Implement a constant for the size of the guessing range (so that you can change only one line of code and change from guessing from 1 – 10 to 1 – 100, for example.)
3. Compare the number of guesses necessary, on average, for 1 -10, 1- 100, 1- 1000
4. What would it take to change the game to guessing a letter from the alphabet? Give me details to how your algorithm would change and how the code would have to change, but it is not necessary to implement it.
5. Implement a max number of guesses in the same manner (make it a constant and check against the number of guesses each turn). Provide some output to the user about how many guesses they have left.
6. Analyze the strategy of guessing from at least three different users and compare them in a few paragraphs. How does this strategy work within the algorithm you developed? What would you change within your algorithm or code to make the game easier or harder, based on this information?

Part 2 Description: In this lab you will write several functions that take the string that the user inputs and test / manipulate it.

Details:

1. Output the string that was input
2. Output the reverse of the string that was input
3. Determine if the word is palindrome and output that result

4. Determine and output the length of the string
5. Output the first half of the string that was input
6. Output the first character of the string that was input
7. Test the code for proper operation

Code Bits: Use cout and cin for the output and input (remember to include iostream!) You will use the string class for this program. You can use its definition on cplusplus.com to determine the functions that you can use on the objects of type string.

Sample Output:

```
Welcome to C++...
Please enter your word: universe
The word universe backwards is esrevinu
This word is not a palindrome.
The word is 8 characters long.
The first half of the word is univ
The first letter is u
```

```
Welcome to C++...
Please enter your word: noon
The word noon backwards is noon
This word is a palindrome.
The word is 4 characters long.
The first half of the word is no
The first letter is n
```

Additional Objectives:

7. Enter two strings and check for a match, output the above info for each.
8. Enter four strings and list the letters that appear in all strings.

Part 3 Objective: You will write a program that asks the user for mathematical function to be performed, and then return to them the values of those functions. You will ask them first for the function to be performed (by operation), then you will ask them for the necessary variables that need to be input. For example, if "sum" is one of the functions, you will ask them for two numbers to be added together and then return that sum to them. The name to use for your menu options is shown after each function.

Code Bits:

1. Simple Sum (2 variables) – "sum"

2. Square Root (1 variable) – “sqr”
3. Number raised to a power (2 variables – the number and the power) – “pow”
4. Max (5 variables - return the maximal value) – “max”
5. Min (5 variables – return the minimal value) – “min”
6. Average (5 variables – return the average) – “avg”

Design: It is imperative that each of these operations be in their own function, and that you should have a function that determines the operation type (one of the 6 above, use a string), and one that takes in a variable for each of the inputs. You may, and perhaps should, have functions that call other functions. I expect all function prototypes to be listed at the top of your code (after using namespace std;)