EE 5103 - Assignment 2

=====================

Total points: 10

Due: Mar 02 by 11:59 PM

Submission instructions:

Name your source files as <LastNameFirstName-q1>.cpp , <LastNameFirstName-q2>.cpp, etc. (e.g. KrishnanRam-q1.cpp). Place them in a folder called <LastNameFirstName> and zip the folder to get <LastNameFirstName>.zip. Upload the file to blackboard.

==================================

<u>Q1 (1 points)</u>

Write a simple program in C++ to display the output below exactly as shown (i.e., it should include white space to position the pyramid toward the right, as shown).

```
>./a.out


                         *
                        ^^^
                       *****
                      ^^^^^^^
                     *********
                    ^^^^^^^^^^^
```

<u>Important:</u>

You are not allowed to use a "string" (or a character string literal) data type in your code. So DO NOT take an apporach similar to the following:

```
cout << "                         *" << endl;
cout << "                        ^^^" << endl;
cout << "                       *****" << endl;
```

Instead, use a for loop or a while loop to print various characters. Observe that the first line contains 1 char, the second line contains 3 chars, the third line contains 5 chars, etc. Also, the characters alternate from one line to another. Use these observations to your advantage in framing your logic inside your for/while loop.

Q2 (1 points)

Write a C++ program to display all unique permutations of an input string. For an input string of “dog”, the output should be: “dog”, “dgo”, “odg”, “ogd”, “god”, “gdo”. However, for “Bob”, the output should be: “bob”, “bbo”, “obb”.

Q3 (1 points)

Write a C++ program to display all subsets of a given set of numbers (read about power set of a given set, if you are not familiar with this concept). For an input set of numbers of {1,2,3}, the output should be displayed exactly in the following format using { and }:

```
{1,2,3}

{1,2}

{1,3}

{2,3}

{1}

{2}

{3}

{}
```

Q4 (2 points)

Write a C++ program that reads a telephone number from the user and outputs if the telephone number is in the right format. For the purpose of this question, a telephone number is in the right format, if the format is (qqq) qqq-qqqq. Here q is a digit from 0 to 9. Also, the very first q cannot be a zero.

Q5 (2 points)

Write a C++ program that reads a sequence of doubles. The program should then print the mean, median, mode, standard deviation and variance of that sequence.

Q6 (3 points)

Write a program that reverses the contents of an array of characters, counts the number of occurrences of each character in the array, and finally displays the reversed array and the computed counts. **The goal is to practice functions. You MUST use the high-level approach (pseudocode) and the functions below:**

```
main()

{

        1. int arraySize=10;
        2. Declare an array of chars: char d[arraySize];
        3. Initialize the array by storing some character in each array index.
        4. Call a function to print the array: display_array_of_chars(d, arraySize)
        5. Call a function to reverse the array: reverse_array_of_chars(d, arraySize)
        6. Call a function to print the reversed array: display_array_of_chars(d, arraySize)
        7. Declare an array to store the counts to be computed: int counts[arraySize];
        8. Declare an array to store the characters for which counts have already been computed: char
           counted_chars[arraySize];
        9. int index=0;
        10. for (int i=0; i<arraySize; i++)
              a. if (!has_been_counted(counted_chars, d[i], index) (i.e., if the character d[i] has not
                 already been counted then)
                 {
                     counts[index]=count_occurrences(d, d[i], arraySize);
                     counted_chars[index]=d[i];
                     index++;
                 }
        11. Call a function to display the counts: display_counts(counts, counted_chars, index)

}

void reverse_array_of_chars(char* a, int size)

{

    1. Navigate the array and reverse the contents as follows:
        a. Swap 1st and last element of array a (that is, swap(a[0], a[size-1]))
        b. Swap 2nd and second to last element of array a (that is, swap(a[1], a[size-2]))
        c. And so on.

}

void swap(char* p1, char* p2)

{

        1. swap values pointed to by p1 and p2

}

void display_array_of_chars(char* b, int sz)

{

    1. Loop through the array and display the chars in the array

}

int count_occurrences(char* a, char b, int sz)

{

    1. Loop through the array a and count the number of times the character b occurs in the array a
    2. Return the count
}
void display_counts(int* a, char* b, int sz)
{
    1. Loop through the array a to display the counts of the characters in b
}
bool has_been_counted(char* counted_chars, char x, int sz)
{
        1. Return true if x exists in the array counted_chars, else false
}
```

====

Sample output for an input array d of {‘h’,’e’,’l’,’l’,’o’,’w’,’o’,’r’,’l’,’d’}:

====

```
The input array is: helloworld
The reversed array is: dlrowolleh
# of occurrences of the letter ‘d’: 1
# of occurrences of the letter ‘l’: 3
# of occurrences of the letter ‘r’: 1
# of occurrences of the letter ‘o’: 2
# of occurrences of the letter ‘w’: 1
# of occurrences of the letter ‘e’: 1
# of occurrences of the letter ‘h’: 1
```