NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

CS-1004 Object Oriented Programming Spring-2022
ASSIGNMENT-01

Section (All)

Submission Deadline: 01 March, 2022 11:59 pm

Instructions:

1. Assignments are to be done individually. You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class. The code you write must be your own and you must understand each part of your code. You are encouraged to get help from the instructional staff through google classroom.
2. Do not use any String or math libraries (such as cmath, cstring, string etc) and also do not use built-in function (such as strlen, strcmp etc). **Caution**: zero marks will be awarded.
3. Do not edit **Function Prototypes**. **Caution**: zero marks will be awarded.
4. The usage of string is strictly prohibited.
5. Your code must be **generic**.
6. Marks distribution and test Cases are provided for each question. Your code will be evaluated with **similar test cases**. If the required output is generated, you will be awarded full marks. Failing to generate the correct output will result in zero marks. Total Marks: 200.
7. For **PrintPattern** questions, the output should be properly displayed and well presented. There will be no gtests for **PrintPattern** questions. Static and global variables not allowed for recursive functions.
8. **Plagiarism**: Plagiarism of any kind (copying from others, copying from the internet, etc) is not allowed. If found plagiarized, you will be awarded zero marks in the assignment. Repeating such an act can lead to strict disciplinary actions and failure in the course.
9. **Please start early otherwise you will struggle with the assignment.**
10. **Test cases:** Test cases (in gtest) will be shared with you on Google Classroom. **We will be running your code against our test cases, and a test case failure or a segmentation fault/incorrect result or even syntax error will result in zero marks.**
11. **Submission Guidelines**: Dear students, we will be using **auto-grading tools (gtest)**, so failure to submit according to the below format would result in zero marks in the relevant evaluation instrument.
    a. Make your own file named submission.cpp. Please don't include the main function while submitting the file. And don't remove **test cases** (in testcases.cpp) or **function prototypes** (in submission.cpp).
    b. Your submission.cpp file must contain your name, student-id, and assignment # on the top of the file in the comments.
    c. Move you submission.cpp in one folder. The folder must contain only submission.cpp file (no binaries, no exe files etc.,). If we unable to download your submission due to any reason you will be awarded zero mark.
    d. Run and test your program on a lab machine before submission. If there is a syntax error, zero marks will be awarded in that specific question.
    e. Rename the folder as ROLL-NUM_SECTION (e.g. 21i-0001_A) and compress the folder as a zip file. (e.g. 21i-0001_A.zip). Only zip file will be acceptable.
    f. Submit the .zip file on Google Classroom within the deadline.
    g. Submission other than Google classroom (e.g. email etc.) will not be accepted.

h. The student is solely responsible to check the final zip files for issues like corrupt files, viruses in the file, mistakenly exe sent. If we cannot download the file from Google classroom due to any reason it will lead to zero marks in the assignment. You are required to use Visual Studio 19 or above for the assignment.

Note: Follow the given instruction to the letter, failing to do so will result in a zero.

Q1: Tap Code [Marks:20]

The **tap code**, sometimes called the **knock code**, is a way to encode text messages on a letter-by-letter basis in a very simple way. Tap code has been one of the most basic communication protocols and still used to convey SOS messages and other urgent communication. The tap code uses a 5×5 grid of letters representing all the English alphabets, see Figure 1. To communicate the word "water", the cipher would be the following (with the pause between each number in a pair being shorter (single space) than the pause between letters (two spaces)), see Figure 2

	1	**2**	**3**	**4**	**5**
1	A	B	C/K	D	E
2	F	G	H	I	J
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Figure 1: **Polybius square** 5x5 grid

W	A	T	E	R
5, 2	1, 1	4, 4	1, 5	4, 2
••••• ••	• •	•••• ••••	• •••••	•••• ••

Figure 2

Your task is to design a program that can

i) convert any given string into a Tap code sequence
Prototype: char* convertToTapCode(char*)

ii) and A Tap code sequence to a string (char*)
Prototype: char* convertToString(char*)

Note: You are not authorized to use string data type; however, you can use char*

Reference: https://en.wikipedia.org/wiki/Tap_code

Q2: Big Integer [Marks:30]

Big Integer (char array) is used for the mathematical operations that involve very big integer calculations that are outside the limit of all available primitive data types. For example, factorial of 100 contains 158 digits in it so we can't store it in any primitive data type available. We can store as large Integer in char array. Your goal is to implement following functions.

Addition of Big Integer

In this problem, you have to take two char arrays of numbers (e.g., "1203009954" and "109876201453") from the user and add these two numbers and store answer in a third char array digit by digit ("111079211407").

```
char* additionOfBigInteger(char * Num1, char* Num2)
/* This function returns a char pointer that dynamically
   allocated and stores result after adding num1 and num2.
 */
{
}
```

Subtraction of Big Integer

In this problem, you have to take two char arrays of numbers (e.g., and "1203009954" "109876201453") from the user and subtract these two numbers and store answer in a third char array digit by digit ("-108673191499").

```
char* subtractionOfBigInteger(char * Num1, char* Num2)
/* This function returns a char pointer that dynamically
   allocated and stores result after subtracting num2 from num1.
 */
{
}
```

Multiplication of Big Integer

In this problem, you have to take two char arrays of numbers (e.g., and "1203009954" "109876201453") from the user and multiply these two numbers and store answer in a third char array digit by digit ("132182164055668263162").

```
char* multiplicationOfBigInteger(char * Num1, char* Num2)
/* This function returns a char pointer that dynamically
   allocated and stores result after xing num1 and num2.
 */
{
}
```

Note: apply validation check (e.g. no alphabets or special characters "(+_-)(*&^%$#@!~`'";:,<>./?" Etc) is not allowed for input).

NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

Q3: Text Analysis [Marks:50]

The availability of computers with string-manipulation capabilities has resulted in some rather interesting approaches to analyzing the writings of great authors. This exercise examines two methods for analyzing texts with a computer. You have to use char * for following exercises.

1. Write a function which remove punctuations marks (.~ ! @ # $ % ^ & * ( ) _ + = " ; : / ?) from the paragraph.
   Note: Remember
   i) Output must only contain alphabets and space characters.
   ii) There must be single space between two words.

```
void removePunctuation(char *str)
/* This function receives string (char*)
And remove punctuation marks.
*/
{
}
```

   Example:
   Input: "To be, or not to be: that is the question:"
   Output: "To be or not to be that is the question"
2. Write a function that receives a string consisting of several lines of text (paragraph) and returns an array indicating the number of occurrences of each letter of the alphabet in the text.

```
void countLetters(char* str, int*& array, int & size)
/* Parameters:
Input:
Char* str: a multiline string. E.g., "This is a test String"
Output:
int *: arr should contain the frequency of characters given in sequence of original
character array.
i-e
Index 0 contains frequency of 'T' //case sensitive
Frequency of 'T' is 1
Index 1 contains frequency of 'h'
Frequency of 'h' is 1
Index 2 contains frequency of 'i'
Frequency of 'i' is 3
Index 3 contains frequency of 's'
Frequency of 's' is 2
Index 4 contains frequency of ' '(space)
Frequency of ' ' is 4
Index 5 contains frequency of 'a'
Frequency of 'a' is 1
Index 6 contains frequency of 't'
Frequency of 't' is 3
and so on

int : array size. In this example size = 12
*/
{
}
```

Q4: Recursive Functions

Marks:10

1. Given a number n, write a recursive function whether it's prime number or not. Your function should return true or false.
   Function Prototype: bool isprimeNumber(int n);

Marks:15

2. Given a string, write a recursive function to find its first uppercase letter. Your function should return the letter.
   Function Prototype: char findUppercase(char* str);

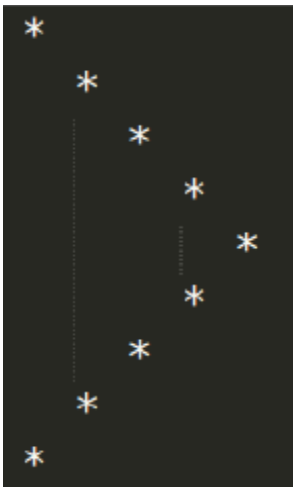
Marks:15

3. Write a program that takes a 2D pointer array and calculate sum of even and odd using recursive function.
   *int sum(int **array, int row, int column, int &evenSum, int &oddSum)*

Marks:20

4. Write a C++ recursive function PrintPattern1 to print following pattern using recursion. No loops allowed whatsoever, and you can write other helping (recursive) functions. For example, calling your function with these argument PrintPattern1(1,10) should print following pattern. Your function prototype must be as follows:

```
void PrintPattern1(int start, int end);
```



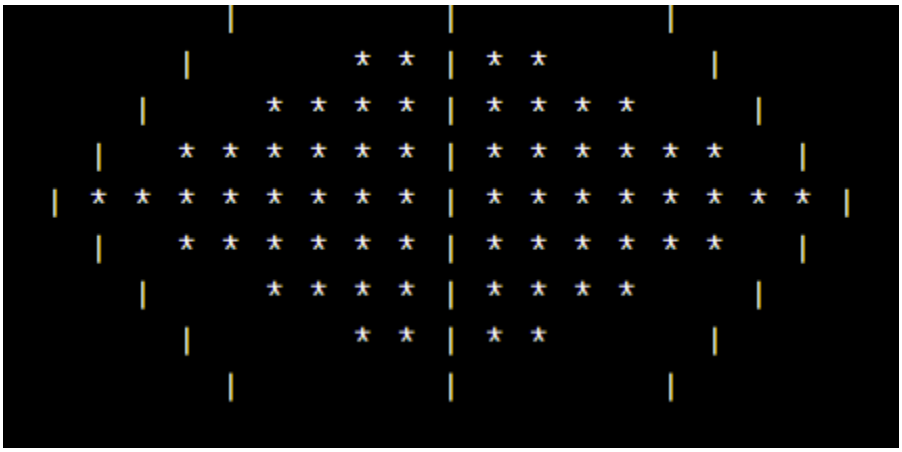
Marks:20

5. Write a C++ recursive function PrintPattern2 to print following pattern using recursion. No loops allowed whatsoever, and you can write other helping (recursive) functions. For example, calling your function with these argument PrintPattern2(5,1,5) should print following pattern. Your function prototype must be as follows:
   void PrintPattern2(int, int ,int);

PrintPattern2(3,1,3)

Marks:20

6. Write a C++ recursive function PrintPattern3 to print following pattern using recursion. No loops allowed whatsoever, and you can write other helping (recursive) functions. For example calling your function with these a argument PrintPattern3(5,1,5) should print following pattern. Your function prototype must be as follows:
   void PrintPattern3(int, int ,int);



PrintPattern3(4,1,4)

Happy Coding ☺