NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

CS-1004 Object Oriented Programming Spring-2025
ASSIGNMENT-01

Section (All)

Submission Deadline: 17th February, 2025 3:00 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.
8. Static and global variables are not allowed.
9. **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.
10. **Please start early otherwise you will struggle with the assignment.**
11. **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.**
12. **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. 24i-0001_A) and compress the folder as a zip file. (e.g. 24i-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: Morse Code [20 Marks]

Morse code has been one of the most basic communication protocol and still used to convey SOS messages and other urgent communication. In Morse code each English alphabet is encoded by a sequence of '.' (Dot) and '-' (Dash), see Figure 1 for complete mapping between English alphabets and Morse codes. Letters are separated by spaces and words by "/".

Your task is to design a program that can:

i) Convert any given string into a Morse code sequence
Prototype: char* convertToMorseCode(char*)

ii) A Morse code sequence to a string.
Prototype: char* convertToString(char*)

Note: You are not authorized to use string, cstring libraries etc.

Letters, numbers, punctuation

Character	Code	Character	Code	Character	Code	Character	Code	Character	Code
A	· –	J	· – – –	S	· · ·	1	· – – – –	Period [.]	· – · – · –
B	– · · ·	K	– · –	T	–	2	· · – – –	Comma [,]	– – · · – –
C	– · – ·	L	· – · ·	U	· · –	3	· · · – –	Question mark [?]	· · – – · ·
D	– · ·	M	– –	V	· · · –	4	· · · · –	Apostrophe [']	· – – – – ·
E	·	N	– ·	W	· – –	5	· · · · ·	Exclamation mark [!]	– · – · – –
F	· · – ·	O	– – –	X	– · · –	6	– · · · ·	Slash [/], Fraction bar	– · · – ·
G	– – ·	P	· – – ·	Y	– · – –	7	– – · · ·	Parenthesis open [(]	– · – – ·
H	· · · ·	Q	– – · –	Z	– – · ·	8	– – – · ·	Parenthesis close [)]	– · – – · –
I	· ·	R	· – ·	0	– – – – –	9	– – – – ·	Ampersand [&], Wait	· – · · ·

Figure 1: A mapping table between English Alphabets and Morse codes. Source: http://en.wikipedia.org/wiki/Morse code

Q2: Big Integer [30 Marks]

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 multiplying num1 and num2.
 */
{}
```

Factorial of Big Integer

In this problem, you have to take char array of number (e.g., "20") from the user and calculate factorial of that number. ("2432902008176640000").

```
char* factorialOfBigInteger(char * Num1)
/* This function returns a char pointer that dynamically
   allocated and stores result.
 */
{}
```

Note: apply validation check (e.g. no alphabets or special characters allowed for input)

Q3: Text Analysis [40 Marks]

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 the following exercises:

1. Write a function which remove punctuations marks (.~ ! @ # $ % ^ & ; * ( ) _ + = " ; : / ? etc.) From the paragraph.
   Note: Remember
   i) Output must only contain alphabets and space characters.
   ii) There must be single space between two words. At the start or end there will be no space.

```
void removePunctuation(char *str)
/* This function receives string (char*)
And removes 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*& count, char*& letters, int &
size)
/* Parameters:
Input:
Char* str: a multiline string. E.g., "This is a test String"
Output:
int *: array should contain the frequency of characters given in sequence of original
character array.
char *: unique characters
i-e
Index 0 contains frequency of 't' //Upper or lower case has same frequency
Frequency of 'T' is 4
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 3
Index 4 contains frequency of ' '(space)
Frequency of ' ' is 4
```

```
Index 5 contains frequency of 'a'
Frequency of ‘a’ is 1
and so on
int : array size. In this example size = 10*/
{}
```

3. 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 word of the alphabet in the text.

```
void countWords(char* str, int*& array, char**&words, int &
numberOfwords)
/* Parameters:
Input:
Char* str: a multiline string. E.g., “This is a test This string is a test”
Output:
int *: arr should contain the frequency of characters given in sequence of original
character array.
char**: unique words
i-e
Index 0 contains frequency of 'this' //Upper or lower case has same frequency
Frequency of ‘this’ is 2
Index 1 contains frequency of 'is'
Frequency of ‘is’ is 2
Index 2 contains frequency of 'a'
Frequency of ‘a’ is 2
Index 3 contains frequency of 'test'
Frequency of ‘test’ is 2
Index 4 contains frequency of 'string'
Frequency of ‘string’ is 1

int : array size. In this example size = 5
*/{}
```

Q4: Recursive Functions [40 Marks]

1. Given a number n, write a recursive function whether it’s prime number or not. Your function should return true or false. **[10 Marks]**
   Function Prototype: *bool isprimeNumber(int n);*
   Note: No loops allowed whatsoever

2. Write a program that takes a 2D pointer array and calculate sum of even and odd numbers using recursive function. **[10 Marks]**
   Function Prototype:
   *int sum(int **array, int row, int column, int &evenSum, int &oddSum)*
   Note: No loops allowed whatsoever

NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

3. 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(5,1,5) should print following pattern. **[10 Marks]**

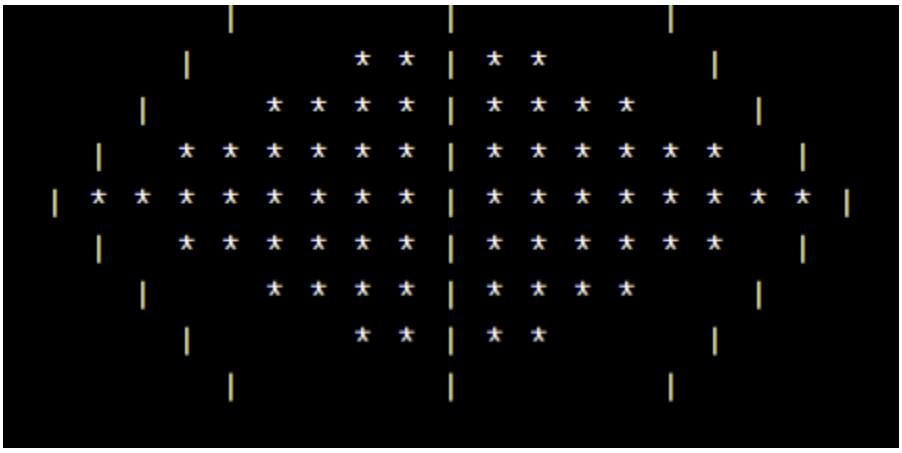
Your function prototype must be as follows:
void PrintPattern1 (int, int, int);

this is for (5,1,5) >>

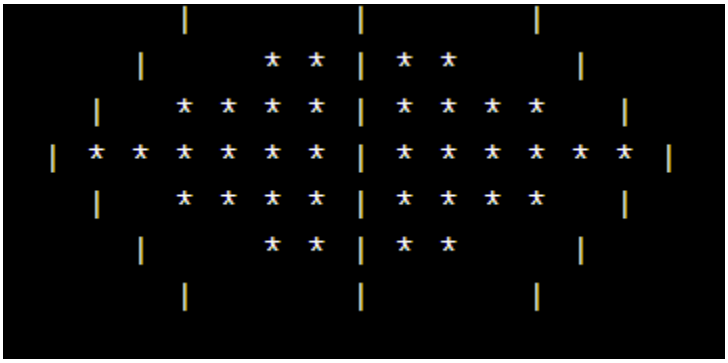
e.g) PrintPattern2(3,1,3)

4. 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. **[10 Marks]**

Your function prototype must be as follows:
void PrintPattern2(int, int ,int);



e.g) PrintPattern3(4,1,4)



Q5: Maze Traversal [35 Marks]

Objective: Develop a C++ function that determines if a path exists for a rat to traverse from the top-left corner (0,0) to a specified target cell (x, y) in a given N×M binary maze. The function should utilize recursive backtracking to explore potential paths and update the maze in place to indicate the solution path.

Warning:
Use of AI tools like ChatGPT, DeepSeek are strictly prohibited for this question. Any AI-generated content will be checked. This is a great opportunity to learn backtracking and recursion. Solve this problem by yourself to build strong problem-solving skills.

Background:

In the pf assignment 4 (Q3), you solved the "Rat in a Maze" problem using an array-based backtracking approach. This time, you will advance to a more sophisticated method by implementing recursive backtracking. This approach will allow you to explore all possible paths from the starting point to the destination, backtracking when a path leads to a dead end. The recursive nature of this method will enable you to efficiently navigate the maze and find the solution.

Assignment Details:

Function Prototype:

```
bool solveMaze(int** maze, int t_row, int t_column, int rows, int cols, int
thisX = 0, int thisY = 0);
```

- `maze`: A pointer to a dynamically allocated N×M binary matrix representing the maze, where `maze[i][j]` is 1 for an open path and 0 for a wall.
- `t_row`: The row index of the target cell.
- `t_column`: The column index of the target cell.
- `rows`: The number of rows in the maze.
- `cols`: The number of columns in the maze.

Function Requirements:

- The function should modify the `maze` in place to represent the solution path, marking the path with 2's.
- The function should return `true` if a path exists from the top-left corner (0,0) to the target cell (x, y), and `false` otherwise.
- The function should ensure that the rat can only move through cells with a value of 1 (open paths) and should backtrack when encountering walls (cells with a value of 0).

Notes:

- Comment your code to explain the logic and flow, especially the recursive backtracking process.

Sample Input:
1 1 0 1 1 1
0 1 0 1 0 1
0 1 0 1 1 1
1 1 1 0 0 1
1 0 0 1 1 1
1 1 1 1 0 1

Sample Output:
2 2 0 1 1 1
0 2 0 1 0 1
0 2 0 1 1 1
2 2 1 0 0 1
2 0 0 2 2 2
2 2 2 2 0 2

Q6. Sudoku Solver [35 Marks]

Objective: Develop a function in C++ that solves a given Sudoku puzzle using backtracking. The function should utilize pointers and recursion exclusively. The solution will be evaluated through Google Test, so it's essential to follow the specified function prototype and parameter descriptions.

Warning:
Use of AI tools like ChatGPT, DeepSeek are strictly prohibited for this question. Any AI-generated content will be checked. This is a great opportunity to learn backtracking and recursion. Solve this problem by yourself to build strong problem-solving skills.

Background:

What is Sudoku?

NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

Sudoku is a logic-based number-placement puzzle consisting of a 9x9 grid divided into nine 3x3 subgrids. The objective is to fill the grid so that each row, each column, and each 3x3 subgrid contains all the digits from 1 to 9 without repetition.

Backtracking in Sudoku Solving:

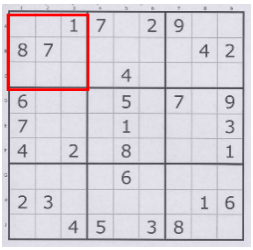
Backtracking is a systematic method for solving problems by trying out possible solutions and undoing them if they lead to a dead end.
The N-Queen problem is an excellent question to learn backtracking. You can read more about it here.

In the Context of Sudoku: You Can Use This Logic or Create Your Own

Start from index (0,0) and try placing numbers 1–9 in any direction. If a number isn't valid, recursively try the next number. If none fit (up to 9), backtrack (return 0) as a previous placement was incorrect. When a valid number is placed, update the grid and recursively call the function, storing the result in a `bool` variable. If it returns 0, the placement was wrong, so try the next number; if it returns 1, the solution is correct, and you return 1.

Assignment Details:

Function Prototype:
`bool solveSudoku(int** board/*Default parameters are here*/);` where board is a pointer to a 9x9 dynamic Sudoku grid, with `board[i][j]` holds an integer (0–9), with 0 indicating an empty cell.



You are allowed to use up to 5 default parameters. For example:
`bool solveSudoku(int** board, int x = 0, int y = 0, int blockx = 0, int blocky = 0, int num = 1);`
Here, `blockx` and `blocky` represent the 3x3 sub grid where the current recursion is taking place.

Function Requirements:

- The function should modify the `board` in place to contain the solved Sudoku puzzle if a solution exists.
- The function should return `true` if the puzzle is solvable and `false` otherwise.

- The function should ensure that the Sudoku rules are maintained: no repeated numbers in any row, column, or 3x3 sub grid.

Notes:

- Comment your code to explain the logic and flow, especially the recursive backtracking process.
- Loop is **ONLY** allowed in the helping function that you will use to check if a number already exists in your row, column or sub grid. The rest needs to be done using backtracking (recursion).

Happy Coding 😊