Object Oriented Programming Lab

SPRING 2024

LAB 03

FAST National University of

Computerand Emerging Sciences

Learning Outcomes

In this lab you are expected to learn the following:

- Double Pointers
- Dynamic Memory Allocation

POINTERS

A pointer is a variable that stores an address of a memory location.

Pointer Declaration and initialization

Pointers is declared as follow.

Type* <variable Name>;

```
#include<iostream>
using namespace std;

int main(){

    int *x=NULL;
    // or
    int* x=NULL;
    //or
    int * x=nullptr;

    return 0;
}
```

Address-of operator (&)

The ampersand, **&**, called the **address of operator**, is a unary operator that returns the addressof its operand.

```cpp
#include<iostream>
using namespace std;

int main(){

    //declare and initialize string* variable
    string *str=nullptr;
    //declare and initialize string variable
    string temp="hello";
    //store the address of temp to str, both will
    //start pointing to same memory location
    str=&temp;


    //print the address of temp
    cout<<"temp Address "<<&temp<<endl;

    //print value in str
    cout<<"str Value "<<str<<endl;

    return 0;
}
```

```
temp Address 0x7ffdc2d7c440
str Value 0x7ffdc2d7c440
```

Dereference operator (*)

The unary operator * is the dereferencing operator that returns the value of its operand.

```cpp
#include<iostream>
using namespace std;

int main(){

    //declare and initialize string* variable
    string *str=nullptr;
    //declare and initialize string variable
    string temp="hello";
    //store the address of temp to str, both will
    //start pointing to same memory location
    str=&temp;


    //print the address of temp
    cout<<"temp Value "<< temp<<endl;

    //Dereferencing to print value in str
    cout<<"str Value "<< *str<<endl;

    return 0;
}
```

```
temp Value hello
str Value hello
```

Initializing double pointers

```
#include <iostream>
using namespace std;

int main()
{
    int** dpointer=new int*[5];
    for(int i=0;i<5;i++)
    {
        *(dpointer+i)=new int[6];
            //or
        dpointer[i]=new int[6];

    }


    return 0;
}
```

Traversing double pointers

```
for(int i=0; i<5; i++)
{
    for(int j=0; j<6;j++)
    {
        cout<<*(*(dpointer+i)+j); //pointer notation
        //or
        cout<<dpointer[i][j]; //Array notation
    }
}
```

Problem 1:

Write a function that finds and returns the maximum element in a dynamically allocated 2D array. Use pointers for array traversal.

Function Prototype: int findMaxElement(int **matrix, int rows, int columns)

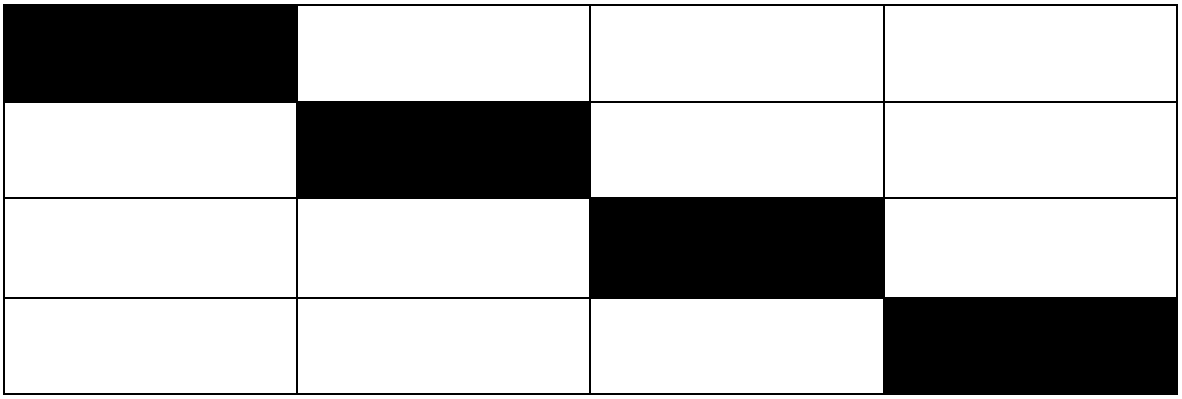
Problem 2

Write a function named as calDiagonal that receives three arguments:

(i) a 2D pointer p;
(ii) number of rows sizeA;
(iii) number of columns sizeB;

The function will return sum of left diagonal columns which are shown as black area in the following figure of the 2D array. Remember you have to calculate sum of values of the array shown in the black area in the given figure only and return the sum

Function Prototype: int calDiagonal(intp,int sizeA,int sizeB)**



Problem 3:

Given a m x n matrix grid which is sorted in non-increasing order both row-wise and column-wise, return the number of ***negative*** numbers in *a* grid.

Function Prototype: int countNegatives(int grid, int rows, int columns)**

Example 1

Input grid = [[4,3,2,-1],[3,2,1,-1],[1,1,-1,-2],[-1,-1,-2,-3]]
Output 8
Explanation There are 8 negatives number in the matrix.

Example 2

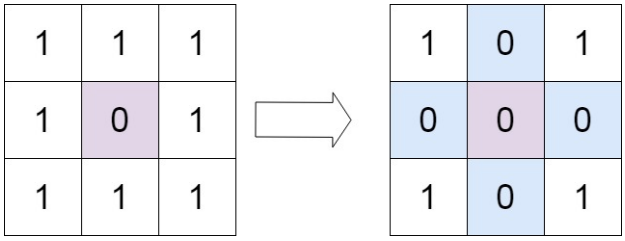
Input grid = [[3,2],[1,0]]
Output 0

Problem 4:

Given an m x n integer matrix matrix, if an element is 0, set its entire row and column to 0's. You must do it in place.

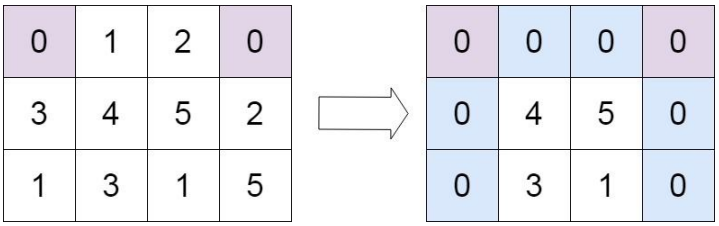
Function Prototype: int setZeroes(int** matrix,int rows, int columns)**

Example 1:



Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]

Example 2:



Input: matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]
Output: [[0,0,0,0],[0,4,5,0],[0,3,1,0]]

Problem 5:

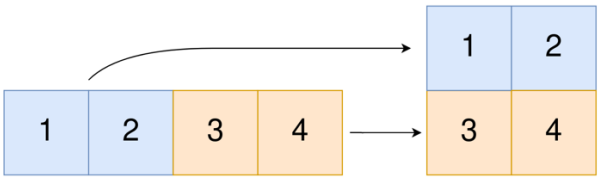
You are given a **0-indexed** 1-dimensional (1D) integer array original, and two integers, m and n. You are tasked with creating a 2-dimensional (2D) array with m rows and n columns using **all** the elements from the original.

The elements from indices 0 to n - 1 (**inclusive**) of original should form the first row of the constructed 2D array, the elements from indices n to 2 * n - 1 (**inclusive**) should form the second row of the constructed 2D array, and so on.

Return an m x n 2D array constructed according to the above procedure, or an empty 2D array if it is impossible.

Function Prototype: Int** construct2DArray(int* original, int m, int n)

Example 1:



Input: original = [1,2,3,4], m = 2, n = 2

Output: [[1,2],[3,4]]
Explanation: The constructed 2D array should contain 2 rows and 2 columns.
The first group of n=2 elements in original, [1,2], becomes the first row in the constructed 2D array.
The second group of n=2 elements in original, [3,4], becomes the second row in the constructed 2D array.

Submission Instructions:

1. Create a single cpp file containing all the functions of the problems. Save the cpp file as Q1.cpp.
2. Now create a new folder with the name ROLLNO_SEC_LAB01 e.g.i23XXXX_A_LAB03.
3. Now you have to submit this zipped file on Google Classroom.
4. If you don't follow the above-mentioned submission instructions, you will be markedzero.
5. Plagiarism in the Lab Task will result in zero marks in the whole category.