**National University of Computer and Emerging Sciences**



**Object Oriented Programming**

**Lab Manual 4**

|  |  |
| --- | --- |
| Course Instructor | Hafiz M. Hamza |
| Lab Instructor(s) | Mian Basam  Khawaja Usman  Shakeel Zafar |
| Section | E & F |
| Semester | Spring 2020 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

## Objectives

After performing this lab, students shall be able to:

* Dynamically allocate memory of 2D arrays
* Pass pointers to 2D arrays to functions
* Use new and delete operators
* Solve some basic problems related to matrices

Important Note:

All allocations of 1D/2D pointers should be dynamic.

Delete the array when it is no longer needed.

All the data will be given by user.

Pass the pointers to functions instead of [].

Make proper functions to solve the problems.

Debug your code to find errors/bugs.

Use offset notation to traverse the arrays (implement your functions using subscript notation, test their functionalities and then change subscript notation to offset notation)

**Exercise 1:** Write a function **int\*\* AllocateMemory(int& rows, int& cols)** that takes size of matrix (rows and columns) from user, allocates memory for the matrix and return its pointer.

What is the advantage of sending the two parameters by reference?

**Exercise 2:** Write a function **void InitializeMatrix(int\*\* matrix, const int& rows, const int& cols)** that takes input from user. You may use subscript operator to initialize elements of matrix (only for this exercise).

Why are we passing the parameters as const?

**Exercise 3:** Write a function **void DisplayMatrix(int\*\* matrix, const int& rows, const int& cols)** that displays the matrix in proper format.

**Exercise 4:** Write an inline function **void DeallocateMemory(int\*\* matrix, const int& rows)** that deallocates all the memory.

Test your program. An example run is given below.

Enter total rows:4  
Enter total columns:3  
The array is:  
0 0 0  
0 0 0  
0 0 0   
0 0 0

**Exercise 5:** Write a function called **maxCol** that takes as parameters a pointer to a 2D array and its dimensions. It should return the largest element in each column of the array. Since there is more than one column in 2D array, you have to return an array that contains largest of each column.

For example, if the **Sample Matrix** is

1 4 **8**

**9** 1 6

5 **7**  2

Your function will return array containing maximum elements of all the columns i.e.

9, 7, 8

**Exercise 6:** A square matrix in which all the entries below the main diagonal are zero is called upper triangular. Write a function IsMatrixUpperTriangular which takes a matrix and returns true if the matrix is upper triangular and false otherwise.

For example, matrix A shown below is upper triangular while matrix B is not upper triangular.

|  |  |  |  |
| --- | --- | --- | --- |
| -1 | 2 | 7 | 0 |
| 0 | 5 | 0 | -1 |
| 0 | 0 | 7 | 0 |
| 0 | 0 | 0 | 0 |

Matrix A

|  |  |  |  |
| --- | --- | --- | --- |
| -1 | 2 | 7 | 0 |
| 0 | 5 | 0 | -1 |
| -9 | 0 | 7 | 0 |
| 0 | 0 | 0 | 0 |

Matrix B

**Exercise 7:** Implement the C++ Functions to tokenize the cstring.

int myTokenizer (char \*data, char\*\*& list\_tokens, char delim)

void printTokens (char\*\* list\_tokens, int size)

Don’t use any built-in function of tokenization. The function “myTokenizer” should store the tokens in the list\_tokens, every row of list\_tokens contains a token and split the data on the basis of the delimiter (delim). The function returns the number of possible tokens. Call the function in main and print the list\_tokens.

Sample Run

|  |  |
| --- | --- |
| Input: my,name,is,Mr.,Ali Delim : , Tokens are :  my name is Mr.  Ali | Input: \_I\_Go\_FAST\_Nu\_ Delim: \_ Tokens are:  I Go  FAST  Nu |