National University of Computer and Emerging Sciences



Lab Manual 02

Object Oriented Programming

|  |  |
| --- | --- |
| Course Instructor | Miss Abeeda Akram |
| Lab Instructor (s) | Miss. Abiha Aftab  Mr. Dilawar Shabbir |
| Section | BCS – 2B |
| Semester | Spring 2021 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

## Objectives

After performing this lab, students shall be able to:

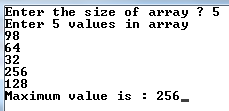
* Have an improved understanding of pointers.
* Create and manipulate 1D dynamic array.
* Allocation and de-allocation of 1D array.

Passing dynamic arrays into functions.

**TASK 1:**

Write a program to allocate memory locations for an array of ‘int’ type using ‘new’ operator. The program should input value for the length of array during program execution. Enter values into array and find the maximum value in the Array.

Result should be:



**TASK 2:**

Write a program to find maximum, minimum and average CGPA of students.

* Create an array to store CGPA of 10 students. (Use proper data type)
* Introduce a pointer and point to array base address
* Use pointer offset method to access array elements.
* Input values from user and display whole array too.
* Deallocate array properly.

**TASK 3:**

Write a C++ program that declares and initializes a float array dynamically and finds the index of the first occurrence of the second largest element in the array.

**For Example:**

**Input:**

Please enter size: 5

Please enter elements: 1.5

7.8

3.2

9.0

7.1

**Output:**

Second Largest element is: 7.8

Index of second largest element is: 1

**TASK 4:**

### Exercise 1 [Input Array]:

Write a function **int\* InputArray(int& size)** that asks user to enter size of required array, allocates the memory on heap, takes input in array and returns its pointer.

### Exercise 2 [Output Array]:

Write a program **void OutputArray(int\* myArray, const int& size)** that takes a pointer to an integer array and prints its data.

Write main function to test above functionality.

### Exercise 3 [Compress Array]:

Write a function **int\* CompressArray(int\* oiginalArr, int& size)** that takes a sorted array and removes duplicate elements from this array.

**Sample Run:**

|  |
| --- |
| **//Input:**  **Enter Size of array:** 10  **Enter 10 elements:** 1 2 2 2 3 3 3 3 3 7  **//Output**  **Array after Compression:** 1 2 3 7 |

Your function will compress the original array, allocate new array of compressed size (compressed size is 4 in above example) on heap, copy updated array in new array and return the new array.

Take input from user by calling **int\* InputArray(int& size)** (function you implemented in Exercise 1). Call CompressArray, call OutputArray(function you implemented in Exercise 2) to display the final output.