**LAB 08 CP**

**Arrays**

**Objective :** **To understand the basic concepts of arrays, reading elements in two dimensional array, array multiplication and array search:**

|  |  |  |  |
| --- | --- | --- | --- |
| No# | Topic/Task | Time | Assessment Marks |
| **Topic Demonstration By Instructor :   30 minutes** | | | |
| 1 | Walk through Task | 15 Minutes | 01 |
| 2 | Array sorting | 15 minutes | 01 |
| 3 | Binary search | 15 minutes | 02 |
| 4 | Multi-dimensional array | 15 minutes | 02 |
| 5 | Transpose of matrix | 15 minutes | 02 |
| 6 | Array elements rotation | 15 minutes | 02 |

# **Lab Policy :**

· All the students have to do this lab individually

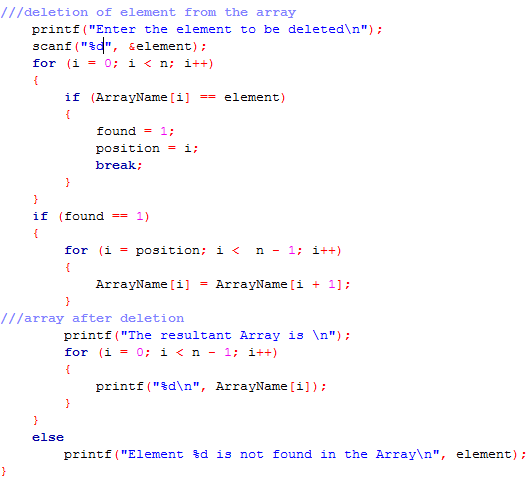
· Anyone doing cheating will be assigned **ZERO** in assessment marks

· All the tasks are mandatory to obtain the assessment marks

· During lab No one is allowed to move outside without permission

**Task-1 Walk through**

One concept of deletion of an element from the array is to override that element in the array with the next element. Initialize an array with 10 elements, take a user input of an element to delete from the array after the deletion display the array elements. This concept is mapped as :



**Exercise 1**

In ascending order the elements are gathered/stored that each coming element is greater/larger than the previous one. Make an array of size of user choice. Input the elements in random order, Sort the elements of array in ascending order and print on console screen.

**Exercise 2**

**binary search**, also known as **half-interval search,**Binary search works on sorted arrays. A binary search begins by comparing the middle element of the array with the target value. If the target value matches the middle element, its position in the array is returned. If the target value is less than or greater than the middle element, the search continues in the lower or upper half of the array, respectively, eliminating the other half from consideration, make an array and do the binary search by taking a target value from user input.

**Exercise 3**

An array keeps track of multiple pieces of information in linear order, a one-dimensional list. However, the data associated with certain systems (a digital image, a board game, etc.) lives in two dimensions. To visualize this data, we need a multi-dimensional data structure, that is, a multi-dimensional array. A two-dimensional array is really nothing more than an array of arrays (a three-dimensional array is an array of arrays of arrays).

And a two-dimensional array looks like this:

int[][] myArray = { {0,1,2,3}, {3,2,1,0}, {3,5,6,1}, {3,8,3,4} };

Develop a 2-D array with 3 rows and 4 columns. Populate it using user input and print the contents of the array, appropriately labeled, with 3 rows and 4 columns. For readability, include a space between each number. Your printout should look like the following, with the x's representing numbers in the array

x x x x

x x x x

x x x x

**Exercise 4**

Two-dimensional array is as a matrix. A matrix can be thought of as a grid of numbers, arranged in rows and columns, kind of like a bingo board. Create a 2-dimensional array and do the matrix multiplication and display the resultant matrix on the console screen.

**Exercise 5**

Transpose of Matrix is the conversion of rows into columns OR columns into rows. Arrays are same as matrix, the transpose of 2–Dimensional array is the transpose of matrix. Make a 2-Dimensional array of the size of user choice. Do the transpose of a matrix and print it.

**Exercise 6**

A *left rotation* operation on an array of size *n* shifts each of the array's elements 1unit to the left. For example, if 2 left rotations are performed on array [1, 2, 3, 4, 5], then the array would become [3, 4, 5, 1, 2].

Given an array of *n* integers and a number, *d*, perform *d* left rotations on the array. Then print the updated array as a single line of space-separated integers.