# Introduction to Computer Programming (CSE1001)

## Assignment-7

## One-Dimensional Array

| Question No | Questions | Course Outcome |
|---|---|---|
| 1. | Write a **Java Program** that creates an array of size **N** and stores random values in it. Implement two separate methods to process the array:<br>- A method to calculate the sum of the array elements:<br>`public static int findSum(int[] arr);`<br>- A method to calculate the average of the array elements:<br>`public static double findAverage(int[] arr);`<br>Use these methods in the main method and display the sum and average of the values stored in the array.<br>**Sample run:**<br>Enter the size of the array: 5<br>Random values stored in the array: 12 7 25 3 18<br>Sum of elements = 65<br>Average of elements = 13.0 | CO3 |
| 2. | Write a **Java Program** that uses an array to read a sequence of integers between **1 and 100**, ending when the user enters **0**. Store and count the occurrences of each number in the array.<br>Your program must use the following methods:<br>1. A method to read the input values and update the counts:<br>`public static void readInput(int[] counts);`<br>2. A method to display how many times each number appears:<br>`public static void displayCounts(int[] counts);`<br>Use these methods in the main method to count and display the occurrences of the numbers.<br>**Sample run:**<br>`Enter integers between 1 and 100: 2 5 6 5 4 3 23 43 2 0`<br>`2 occurs 2 times`<br>`3 occurs 1 time`<br>`4 occurs 1 time`<br>`5 occurs 2 times`<br>`6 occurs 1 time`<br>`23 occurs 1 time`<br>`43 occurs 1 time`<br>**Note** that if a number occurs more than one time, the plural word "*times*" is used in the output. | CO4 |

3.

Write a **Java Program** that reads **10 integers** from the keyboard and stores them in an array. After the array is filled, prompt the user to enter a number to search for. Use the following methods:

1. A method to read and store the 10 integers into the array:

```
public static void readArray(int[] arr);
```

2. A method to count how many times the searched number appears in the array:

```
public static int countOccurrences(int[] arr, int key);
```

In the main method, call these methods and display whether the number is found and how many times it occurs.

**Sample run:**

```
Enter 10 integers:
5 7 3 5 9 5 2 8 3 5
Enter the number to search: 5
The number 5 appears 4 time(s) in the array.
```

CO4

4.

Write a **method** that finds the smallest element in an array of `double` values using the following header:

```
public static double min(double[] array)
```

Write a **Java Program** that prompts the user to enter ten numbers, invokes this method to return the minimum value, and displays the minimum value.

**Sample run:**

```
Enter ten numbers: 1.9 2.5 3.7 2 1.5 6 3 4 5 2
The minimum number is: 1.5
```

CO4

5.

Write a **Java Program** that reads **n** elements into an array and finds the **second largest value**.

Implement a method to determine the second largest element:

public static int findSecondLargest(int[] arr);

Use this **method** in the main method to display the second largest value in the array.

**Note:** Don’t use any **sorting methods**.

**Sample run:**

```
Enter 10 integers:
5 7 3 5 9 5 2 8 3 5
Enter the number to search: 5
The number 5 appears 4 time(s) in the array.
```

CO4

6.

Write a **Java Program** that asks the user to enter the size of an array, then creates the array and inputs the required number of values. Use methods to perform the following tasks:

1. A method to read and store the values in the array:

```
public static void readArray(int[] arr);
```

2. A method to find the maximum value in the array:

```
public static int findMax(int[] arr);
```

3. A method to find the minimum value in the array:

```
public static int findMin(int[] arr);
```

CO6

4. A method to count how many times a given value appears:

```
public static int countOccurrences(int[] arr, int value);
```

5. A method to find the first position of the maximum value:

```
public static int firstPositionOfMax(int[] arr, int max);
```

6. A method to find the last position of the minimum value:

```
public static int lastPositionOfMin(int[] arr, int min);
```

Use these methods in the main method to display:

- The maximum value and how many times it occurs,
- The minimum value and how many times it occurs,
- The first position of the maximum value,
- The last position of the minimum value.

**Sample run:**

```
Enter the size of the array: 8
Enter 8 values:
5 9 2 9 4 2 9 2

Maximum value: 9
It occurs 3 time(s)
First position of maximum: 2

Minimum value: 2
It occurs 3 time(s)
Last position of minimum: 8
```

7.

Write a Java program that reads **10 integers** from the keyboard and stores them in an array. After storing the values, rotate the array elements **clockwise by one position** without using any additional array.

Implement the following **methods**:

1. A method to read and store the 10 integers:

```
public static void readArray(int[] arr);
```

2. A method to rotate the array elements clockwise by one position (in-place):

```
public static void rotateClockwise(int[] arr);
```

3. A method to print the array:

public static void printArray(int[] arr);

Use these **methods** in the main method to input the array, rotate it, and display the rotated result.

**Sample run:**

```
Enter ten numbers: 1 2 3 4 5 6 7 8 9 10
Array elements before clock-wise rotation:
1 2 3 4 5 6 7 8 9 10
Array elements after clock-wise rotation:
10 1 2 3 4 5 6 7 8 9
```

CO4

8.

Write a Java program that creates an integer array of size **10** and fills it with random values between **2 and 30**. Use methods to perform the following tasks:

1. A method to generate and store random values in the array:

```
public static void fillArray(int[] arr);
```

CO4

| | | |
|---|---|---|
| | 2. A method to print the array elements:<br>`public static void printArray(int[] arr);`<br>3. A method to check whether a number is prime:<br>`public static boolean isPrime(int num);`<br>4. A method to count the number of prime numbers in the array:<br>`public static int countPrimes(int[] arr);`<br>Use these methods in the main method to display the array and the total count of prime numbers it contains.<br>**Sample run:**<br>`Array elements: 5 12 17 8 23 2 29 14 19 6`<br>`Number of prime numbers in the array: 6` | |
| 9. | Write a **Java Program** that reads **10 double numbers** from the user and stores them in an array.<br>Implement a method to sort the array using the **bubble sort algorithm**.<br>Use the following method header:<br>`public static void bubbleSort(double[] arr);`<br>The **bubble sort** method should repeatedly compare neighbouring elements in the array and swap them if they are not in order, gradually moving smaller values to the beginning of the array and larger values to the end.<br>Use this **method** in the main method to sort the array and display the sorted numbers. | CO6 |
| 10. | Write a **method** that returns a new array by eliminating the duplicate values in the array using the following method header:<br>`public static int[] eliminateDuplicates(int[] list)`<br>Write a **Java Program** that reads in ten integers, invokes the method, and displays the result.<br>**Sample run:**<br>`Enter ten numbers: 1 2 3 2 1 6 3 4 5 2`<br>`The distinct numbers are: 1 2 3 6 4 5` | CO4 |
| | **HOME ASSIGNMENT** | |
| 11. | Write a **method** that finds the smallest element in an array of `double` values using the following header:<br>`public static double max(double[] array)`<br>Write a **Java Program** that prompts the user to enter ten numbers, invokes this method to return the minimum value, and displays the minimum value.<br>**Sample run:**<br>`Enter ten numbers: 1.9 2.5 3.7 2 1.5 6 3 4 5 2`<br>`The minimum number is: 6` | CO6 |

12. Write a **Java Program** that reads **10 double numbers** from the user and stores them in an array.

Implement a method to sort the array using the **bubble sort algorithm**.
Use the following method header:

```
        public static void bubbleSort(double[] arr);
```

The bubble sort method should repeatedly compare neighboring elements in the array and swap them if they are not in order, gradually moving smaller values to the beginning of the array and larger values to the end.

Use this method in the main method to sort the array and display the sorted numbers.

**Sample run:**

```
Enter 10 numbers:
5.2 3.1 8.4 1.5 7.6 2.3 9.0 4.8 6.7 0.9

Sorted numbers:
0.9 1.5 2.3 3.1 4.8 5.2 6.7 7.6 8.4 9.0
```

CO4

13. Write a **Java Program** that creates an array of size **10** and populates it with random values between **50 and 100**. Implement the following methods:

1. A method to fill the array with random values:
   ```
   public static void fillArray(int[] arr);
   ```
2. A method to print the array elements:
   ```
   public static void printArray(int[] arr);
   ```
3. A method to reverse the order of the array elements:
   ```
   public static void reverseArray(int[] arr);
   ```

Use these **methods** in the main method to display the array **before** and **after** the reversal operation.

**Sample run:**

```
Original array:
52 67 89 73 56 90 61 77 84 99
Reversed array:
99 84 77 61 90 56 73 89 67 52
```

CO4

14. Write a **Java Program** that reads a **decimal integer** from the user and converts it to its **octal representation**.

Implement a method to perform the conversion:

```
     public static String decimalToOctal(int decimal);
```

Use this method in the main method to display the octal equivalent of the input decimal number.

**Sample run:**

```
Enter a decimal number: 125
Octal representation: 175
```

CO4

15. Write a **Java Program** that inputs 10 integer values from the user and moves all zeros to the beginning of the array while maintaining the relative order of all non-zero elements.

**Sample run 1:**

```
Enter 10 integers: 1 2 0 9 3 0 4 0 8 12
After moving all zeros to the beginning:
0 0 0 1 2 9 3 4 8 12
```

**Sample run 2:**

```
Enter 10 integers: 1 2 3 4 5 6 7 8 9 10
After moving all zeros to the beginning:
1 2 3 4 5 6 7 8 9 10
```

***************