Merit-Quality-Excellence

Sukkur IBA University Khairpur Campus

Data Structures

LAB No: 01

Instructor: Marina Gul

Objective of Lab No. 1:
After performing lab1, students will be able to:
- Understand basic concepts of Arrays

Practice 01
Perform following task on an array,

a) **Input size of an array**
b) **Define and create array with the input size on random value**
c) **Display the defined array**
d) **Implement insert, delete and search operation (Binary)**

```
import java.util.Random;
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {

        // Take input the size of an array
        Scanner sc = new Scanner(System.in);  // Create a Scanner object
        System.out.println("Enter the size of an Array");
        int size= sc.nextInt();  // Read user input

        //create array with the input size on random value
        Random rd = new Random(); // creating Random object
        rd.setSeed(System.currentTimeMillis()); // use current time as a seed
        // fill the data array with pseudo-random numbers from 0 to 99,
inclusive
        int[] arr = new int[size];
        for (int i = 0; i < arr.length; i++)
            arr[i] = rd.nextInt(100); // storing random integers in an array


        //Display the defined array
        System.out.println(Arrays.toString(arr));

        //delete an element from array
        arr = MyArray.removeTheElement(arr, 2);
        System.out.println(Arrays.toString(arr));
```

```java
    }
}
/** Program showing some array uses. */

class MyArray{
    // Function to remove the element
    public static int[] removeTheElement(int[] arr, int index)
    {

        // If the array is empty or the index is not in array range
        // return the original array
        if (arr == null || index < 0
                || index >= arr.length) {

            return arr;
        }
        int i=0;
        //start form the index till end of array
        for (i = index; i< arr.length-1; i++)
            arr[i]=arr[i+1];

            // for last index
           arr[i]=0;

        // return the resultant array
        return arr;
    }
    // Returns index of x if it is present in arr[].
    public static int binarySearch(int arr[], int x)
    {
        int l = 0, r = arr.length - 1;
        while (l <= r) {
            int m = l + (r - l) / 2;

            // Check if x is present at mid
            if (arr[m] == x)
                return m;

            // If x greater, ignore left half
            if (arr[m] < x)
                l = m + 1;

            // If x is smaller, ignore right half
            else
                r = m - 1;
        }

        // If we reach here, then element was
        // not present
        return -1;
    }

}
```

Practice 02

Write a function in Java (or any language) to read a list of 10 integer numbers and arrange them in ascending/descending order.

Whenever we do hear sorting algorithms come into play such as selection sort, bubble sort, insertion sort, radix sort, bucket sort, etc. but if we look closer here we are not asked to use any kind of algorithms. It is as simple sorting with the help of linear and non-linear data structures present within java. So there is sorting done with the help of brute force in java with the help of loops and there are two in-built methods to sort in Java.
(See page 104, Chapter 3, M.T. Goodrich)

Ways of sorting in Java

1. Using loops
2. Using sort() method of Arrays class or Collections class

Using loops

```java
public class Main {
    public static void main(String[] args) {
            int arr[] = { 4, 3, 2, 1 }; // Custom input array
        // Outer loop
        for (int i = 0; i < arr.length; i++) {
            // Inner nested loop pointing 1 index ahead
            for (int j = i + 1; j < arr.length; j++) {
                // Checking elements
                int temp = 0;
                if (arr[j] < arr[i]) {
                    // Swapping
                    temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;}
            }
            // Printing sorted array elements
            System.out.print(arr[i] + " ");
        }
    }
}
```

Using sort() method of Arrays class or Collections class

Arrays.Sort() works for arrays which can be of primitive data type also which in turn by default sorts in ascending order.

```java
import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
            int[] arr = { 13, 7, 6, 45, 21, 9, 101, 102 };     // Custom input
array

                // Calling the sort() method present inside Arrays class
                Arrays.sort(arr);
                // Printing and display sorted array
                System.out.printf("Modified arr[] : %s",
                        Arrays.toString(arr));
            }
        }

//Sorting on a subarray

import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
        // Java program to sort a subarray using Arrays.sort()
            int[] arr = { 13, 7, 6, 45, 21, 9, 2, 100 };    // Custom input
array
        // Sort subarray from index 1 to 4, i.e., only sort subarray {7, 6,
45, 21} and
                //keep other elements as it is.
                Arrays.sort(arr, 1, 5);

                // Printing sorted array
                System.out.printf("Modified arr[] : %s",
                        Arrays.toString(arr));
            }
        }
```

Practice 03

Write a function in Java (or any language) to generate an array of 10 integer numbers using random function and sort them in ascending/descending order.

Java has a built-in class, java.util.Random, whose instances are pseudorandom number generators, that is, objects that compute a sequence of numbers that are statistically random. These sequences are not actually random, however, in that it is possible to predict the next number in the sequence given the past list of numbers.

(See page 113, Chapter 3, M.T. Goodrich)

```
import java.util.Arrays;
import java.util.Random;
public class Main {
    public static void main(String[] args) {
        int data[ ] = new int[10];
        Random rand = new Random(); // a pseudo-random number generator
        // use current time as a seed
        rand.setSeed(System.currentTimeMillis());

        // Fill the data array with pseudo-random numbers from 0 to 99,
inclusive
        for (int i = 0; i < data.length; i++)
            data[i] = rand.nextInt(100);  // the next pseudo-random number
        // make a copy of the data array
        int[ ] orig = Arrays.copyOf(data, data.length);
System.out.println("arrays equal before sort: "+ Arrays.equals(data, orig));
```

```
        // sorting the data array (orig is unchanged)
        Arrays.sort(data);

        System.out.println("arrays equal after sort: " + Arrays.equals(data,
orig));
        System.out.println("orig = " + Arrays.toString(orig));
        System.out.println("data = " + Arrays.toString(data));
    }
}
```

Practice 04

Given an integer n. We have n*n values of a 2-d array, and n values of 1-d array. Task is to find the sum of the left diagonal values of the 2-d array and the max element of the 1-d array and print them with space in between.

```
public class Main {

  public static void main(String[] args) {

    int n = 3;


    int[][] array2D = {

      {1, 2, 3},

      {4, 5, 6},

      {7, 8, 9}

    };

     int[] array1D = {10, 20, 30};

    // Calculate the sum of the left diagonal

    int diagonalSum = 0;

    for (int i = 0; i < n; i++) {

      diagonalSum += array2D[i][i];

    }

    // Find the maximum element in the 1D array

    int maxElement = array1D[0];

    for (int i = 1; i < n; i++) {

      if (array1D[i] > maxElement) {

        maxElement = array1D[i];
```

```
            }

        }

                // Print the result

        System.out.println(diagonalSum + " " + maxElement);

    }

}
```

Exercise

1. Write a function in Java (or any language) to read a list of 10 integer numbers and arrange them in such a manner that all the even numbers start from the left and all the odd numbers start from the right.

Input: 1 2 3 5 7 2 2 7 8 9

Output: 1 3 5 7 7 9 2 2 2 8

2. Write a function named **noDup()** that takes a 2D array of size 4x5 and a 1D array of size 20. It should then copy all the elements of 2D array into 1D array but should avoid duplication.

3. Create a file named NLArray.java and design following functions or performing **NLP**.

- **String [] wordTokenize (String fileName)** → Read any text file and return list of words from that file. (Ignore . , : and all these types operators)

- **String[] extractEmail (String fileName)** → Read any text file and return all emails from and file

Note: Read about Natural Language Processing (NLP), Word Tokenizing, Stop Words, Information Extraction/Retrieval for Knowledge.

4. Design following methods in above same class NLArray.java for **Image Cropping.**

- **void extractBoundaries (int arr[][])** → This function should extract boundaries and print from arr (Boundaries include 1st row, 1st col, last row, last col).

- **void cropCenterPart (int arr[][])** → This function should extract center part and print from arr, center part includes everything except Boundaries (Boundaries include 1st row, 1st col, last row, last col).