DSA CODING TRAINING

Name: Dhejan R
Reg No.: 22IT022
Date: 09/11/2024

1. Maximum Subarray Sum – Kadane‟s Algorithm:

Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.

Input: arr[] = {2, 3, -8, 7, -1, 2, 3}
Output: 11
Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.
Input: arr[] = {-2, -4}
Output: –2
Explanation: The subarray {-2} has the largest sum -2.
Input: arr[] = {5, 4, 1, 7, 8}
Output: 25
Explanation: The subarray {5, 4, 1, 7, 8} has the largest sum 25.

Code Solution:

```
import java.util.Scanner;

public class MaximumSubarraySum {
   public static int maxSubarraySum(int[] arr) {
      int a=arr[0];
      int b=arr[0];

      for (int i=1; i<arr.length; i++) {
         b=Math.max(arr[i], b+arr[i]);
         a=Math.max(a, b);
      }

      return a;
   }

   public static void main(String[] args) {
      Scanner scanner=new Scanner(System.in);

            int maxSum1=maxSubarraySum(new int[] {2,3,-8,7,-1,2,3});
      System.out.println("Output: " + maxSum1);

            int maxSum2=maxSubarraySum(new int[] {-2,-4});
      System.out.println("Output: " + maxSum2);
```

```
        int maxSum3=maxSubarraySum(new int[] {5,4,1,7,8});
        System.out.println("Output: " + maxSum3);

        scanner.close();
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac MaximumSubarraySum.java

C:\Users\Legion\Desktop\DSA Coding Questions>java MaximumSubarraySum
Output: 11
Output: -2
Output: 25

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (1)

2. Maximum Product Subarray

Given an integer array, the task is to find the maximum product of any subarray.

Input: arr[] = {-2, 6, -3, -10, 0, 2}
Output: 180
Explanation: The subarray with maximum product is {6, -3, -10} with product = 6 * (-3) * (-10) = 180
Input: arr[] = {-1, -3, -10, 0, 60}
Output: 60
Explanation: The subarray with maximum product is {60}.

Code Solution:

```
//Maximum Product Subarray

public class MaximumProductSubarray {
    public static int maxProduct(int[] arr) {
        if (arr==null || arr.length==0){
            return 0;
        }

        int maxp=arr[0];
        int minp=arr[0];
        int res=arr[0];

        for (int i=1; i<arr.length; i++) {
```

```
            if (arr[i]<0) {
                int temp=maxp;
                maxp=minp;
                minp=temp;
            }
            maxp=Math.max(arr[i], maxp*arr[i]);
            minp=Math.min(arr[i], minp*arr[i]);
            res=Math.max(res, maxp);
        }

        return res;
    }

    public static void main(String[] args) {
        int[] arr1 = {-2, 6, -3, -10, 0, 2};
        System.out.println("Output: " + maxProduct(arr1));

        int[] arr2 = {-1, -3, -10, 0, 60};
        System.out.println("Output: " + maxProduct(arr2));
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac MaximumProductSubarray.java

C:\Users\Legion\Desktop\DSA Coding Questions>java MaximumProductSubarray
Output: 180
Output: 60

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (1)

3. Search in a sorted and rotated Array

Given a sorted and rotated array arr[] of n distinct elements, the task is to find the index of given key in the array. If the key is not present in the array, return -1.

Input : arr[] = {4, 5, 6, 7, 0, 1, 2}, key = 0
Output : 4
Input : arr[] = { 4, 5, 6, 7, 0, 1, 2 }, key = 3
Output : -1
Input : arr[] = {50, 10, 20, 30, 40}, key = 10
Output : 1

Code Solution:

```
// Search in rotated and sorted array

public class rotatedSearch {
   public static int searchTarget(int[] arr, int key) {
      int l=0;
               int r=arr.length-1;

      while (l<=r) {
         int mid=l+(r-l)/2;

         if (arr[mid]==key) {
            return mid;
         }

         if (arr[l]<=arr[mid]) {
            if (arr[l]<=key && key<arr[mid]) {
               r=mid-1;
            } else {
               l=mid+1;
            }
         } else {
            if (arr[mid]<key && key<=arr[r]) {
               l=mid+1;
            } else {
               r=mid-1;
            }
         }
      }

      return -1;
   }

   public static void main(String[] args) {
      int[] arr1={4, 5, 6, 7, 0, 1, 2};
      int key1=0;
      System.out.println("Output: "+searchTarget(arr1, key1));

      int[] arr2={4, 5, 6, 7, 0, 1, 2};
      int key2=3;
      System.out.println("Output: "+searchTarget(arr2, key2));

      int[] arr3={50, 10, 20, 30, 40};
      int key3=10;
      System.out.println("Output: "+searchTarget(arr3, key3));
```

```
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac rotatedSearch.java

C:\Users\Legion\Desktop\DSA Coding Questions>java rotatedSearch
Output: 4
Output: -1
Output: 1

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (log n)
Space Complexity: O (1)

4. Given n non-negative integers a1, a2, ..., a_n where each represents a point at coordinate (i, a_i). 'n' vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

The program should return an integer which corresponds to the maximum area of water that can be contained (maximum area instead of maximum volume sounds weird but this is the 2D plane we are working with for simplicity).

Note: You may not slant the container.

Code Solution:

```
//Container with Most water

public class mostWater{
    public static int maxArea(int[] arr){
        int res=0;
        int l=0, r=arr.length-1;

        while(l!=r){
            int area=(r-l)*Math.min(arr[l], arr[r]);
            res=Math.max(res, area);
            if (arr[l]>arr[r]){
                r--;
            }
            else{
                l++;
            }
        }
        return res;
```

```
        }
        public static void main(String[] args){
            int[] arr1 = {1, 5, 4, 3};
            System.out.println("Output: " + maxArea(arr1));

            int[] arr2 = {3, 1, 2, 4, 5};
            System.out.println("Output: " + maxArea(arr2));
        }
    }
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac mostWater.java

C:\Users\Legion\Desktop\DSA Coding Questions>java mostWater
Output: 6
Output: 12

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (1)

5. Find the Factorial of a large number

Input: 100
Output:
933262154439441526816992388562667004907159682643816214685929638952175
999932299156089414639761565182862536979208272237582511852109168640000
00000000000000000000
Input: 50
Output:
30414093201713378043612608166064768844377641568960512000000000000

Code Solution:

```
// Factorial of large number

import java.math.BigInteger;
import java.util.Scanner;

public class largeFactorial{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        BigInteger f=BigInteger.ONE;

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

```
                f=f.multiply(BigInteger.valueOf(i));
            }
            System.out.println("Input: "+ n);
            System.out.println("Output: "+ f);
        }
    }
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac largeFactorial.java

C:\Users\Legion\Desktop\DSA Coding Questions>java largeFactorial
100
Input: 100
Output: 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000

C:\Users\Legion\Desktop\DSA Coding Questions>java largeFactorial
50
Input: 50
Output: 30414093201713378043612608166064768844377641568960512000000000000

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (1)

6. Trapping Rainwater Problem states that given an array of n non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.

Input: arr[] = {3, 0, 1, 0, 4, 0, 2}
Output: 10
Explanation: The expected rainwater to be trapped is shown in the above image.
Input: arr[] = {3, 0, 2, 0, 4}
Output: 7
Explanation: We trap 0 + 3 + 1 + 3 + 0 = 7 units.
Input: arr[] = {1, 2, 3, 4}
Output: 0
Explanation : We cannot trap water as there is no height bound on both sides
Input: arr[] = {10, 9, 0, 5}
Output: 5
Explanation : We trap 0 + 0 + 5 + 0 = 5

Code Solution:

Output:

Time Complexity:
Space Complexity:

7. Chocolate Distribution Problem

Given an array arr[] of n integers where arr[i] represents the number of chocolates in ith packet. Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that:

Each student gets exactly one packet.

The difference between the maximum and minimum number of chocolates in the packets given to the students is minimized.

Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 3

Output: 2

Explanation: If we distribute chocolate packets {3, 2, 4}, we will get the minimum difference, that is 2.

Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 5

Output: 7

Explanation: If we distribute chocolate packets {3, 2, 4, 9, 7}, we will get the minimum difference, that is 9 – 2 = 7.

Code Solution:

```
// Chocolate Distribution Problem
import java.util.Arrays;
import java.util.Scanner;

public class chocolateDistribution{
	public static void main(String[] args){
		Scanner sc=new Scanner(System.in);
		int[] arr={7,3,2,4,9,12,56};

		System.out.println("Input Array: "+Arrays.toString(arr));
		System.out.println("Enter the No. of Students: ");
		int m=sc.nextInt();
		int n=arr.length-1;
		int a=arr[n];

		Arrays.sort(arr);

		for (int i=0; i+m-1<n+1; i++){
			a=Math.min(a,arr[i+m-1]-arr[i]);
		}
		System.out.println("Output: "+a);
	}
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>java chocolateDistribution
Input Array: [7, 3, 2, 4, 9, 12, 56]
Enter the No. of Students:
3
Output: 2

C:\Users\Legion\Desktop\DSA Coding Questions>java chocolateDistribution
Input Array: [7, 3, 2, 4, 9, 12, 56]
Enter the No. of Students:
5
Output: 7

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n log n)
Space Complexity: O (n)

8. Merge Overlapping Intervals
Given an array of time intervals where arr[i] = [starti, endi], the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals.

Input: arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]]
Output: [[1, 4], [6, 8], [9, 10]]
Explanation: In the given intervals, we have only two overlapping intervals [1, 3] and [2, 4]. Therefore, we will merge these two and return [[1, 4}], [6, 8], [9, 10]].
Input: arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]]
Output: [[1, 6], [7, 8]]
Explanation: We will merge the overlapping intervals [[1, 5], [2, 4], [4, 6]] into a single interval [1, 6].

Code Solution:

```
// Merge Overlap Intervals

import java.util.*;

public class Overlap {
   public static int[][] mergeOverlap(int[][] intervals) {
      if (intervals.length == 0) {
         return new int[0][];
      }
      Arrays.sort(intervals, Comparator.comparingInt(a -> a[0]));

      List<int[]> merged = new ArrayList<>();

      int[] current = intervals[0];
      merged.add(current);
```

```
        for (int i = 1; i < intervals.length; i++) {
            int currentEnd = current[1];
            int nextStart = intervals[i][0];
            int nextEnd = intervals[i][1];
            if (currentEnd >= nextStart) {
                current[1] = Math.max(currentEnd, nextEnd);
            } else {
                current = intervals[i];
                merged.add(current);
            }
        }
        return merged.toArray(new int[merged.size()][]);
    }

    public static void main(String[] args) {
        int[][] arr1 = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
        System.out.println("Input: " + Arrays.deepToString(arr1));
        System.out.println("Output: " + Arrays.deepToString(mergeOverlap(arr1)));

        System.out.println();

        int[][] arr2 = {{1, 4}, {6, 8}, {9, 10}};
        System.out.println("Input: " + Arrays.deepToString(arr2));
        System.out.println("Output: " + Arrays.deepToString(mergeOverlap(arr2)));

        System.out.println();

        int[][] arr3 = {{1, 5}, {2, 6}, {3, 7}};
        System.out.println("Input: " + Arrays.deepToString(arr3));
        System.out.println("Output: " + Arrays.deepToString(mergeOverlap(arr3)));

        System.out.println();

        int[][] arr4 = {{7, 8}, {1, 5}, {2, 4}, {4, 6}};
        System.out.println("Input: " + Arrays.deepToString(arr4));
        System.out.println("Output: " + Arrays.deepToString(mergeOverlap(arr4)));

        System.out.println();

        int[][] arr5 = {{1, 10}, {2, 5}, {6, 8}};
        System.out.println("Input: " + Arrays.deepToString(arr5));
        System.out.println("Output: " + Arrays.deepToString(mergeOverlap(arr5)));
```

```
        System.out.println();

        int[][] arr6 = {{1, 3}, {2, 4}, {5, 7}, {6, 8}};
        System.out.println("Input: " + Arrays.deepToString(arr6));
        System.out.println("Output: " + ArraysdeepToString(mergeOverlap(arr6)));
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>java Overlap
Input: [[1, 3], [2, 4], [6, 8], [9, 10]]
Output: [[1, 4], [6, 8], [9, 10]]

Input: [[1, 4], [6, 8], [9, 10]]
Output: [[1, 4], [6, 8], [9, 10]]

Input: [[1, 5], [2, 6], [3, 7]]
Output: [[1, 7]]

Input: [[7, 8], [1, 5], [2, 4], [4, 6]]
Output: [[1, 6], [7, 8]]

Input: [[1, 10], [2, 5], [6, 8]]
Output: [[1, 10]]

Input: [[1, 3], [2, 4], [5, 7], [6, 8]]
Output: [[1, 4], [5, 8]]

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n log n)
Space Complexity: O (n)

9. A Boolean Matrix Question
Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1.

Input: {{1, 0},
{0, 0}}
Output: {{1, 1},
{1, 0}}
Input: {{0, 0, 0},
{0, 0, 1}}
Output: {{0, 0, 1},
{1, 1, 1}}
Input: {{1, 0, 0, 1},
{0, 0, 1, 0},
{0, 0, 0, 0}}
Output: {{1, 1, 1, 1},
{1, 1, 1, 1},

{1, 0, 1, 1}}

Code Solution:

```
// Boolean Matrix Question
import java.util.Arrays;

public class Matrix {
   public static void booleanMatrix(int[][] arr) {
      int r=arr.length;
      int c=arr[0].length;
      for (int i=0; i<r; i++) {
         for (int j=0; j<c; j++) {
            if (arr[i][j]==1) {
               for (int x=0; x<r; x++) {
                  if (arr[x][j]==0) {
                     arr[x][j]=2;
                  }
               }
               for (int x=0; x<c; x++) {
                  if (arr[i][x]==0) {
                     arr[i][x]=2;
                  }
               }
            }
         }
      }

      for (int i=0; i<r; i++) {
         for (int j=0; j<c; j++) {
            if (arr[i][j]==2) {
               arr[i][j]=1;
            }
         }
      }
   }

   public static void main(String[] args) {
      int[][] arr={
         {1, 0, 0, 1},
         {0, 0, 1, 0},
         {0, 0, 0, 0}
      };

            System.out.println("Input:");
```

```
        for (int i=0; i<arr.length; i++) {
            for (int j=0; j<arr[0].length; j++) {
                System.out.print(arr[i][j]+" ");
            }
            System.out.println();
        }
        booleanMatrix(arr);

                System.out.println();

                System.out.println("Output:");
        for (int i=0; i<arr.length; i++) {
            for (int j=0; j<arr[0].length; j++) {
                System.out.print(arr[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac Matrix.java

C:\Users\Legion\Desktop\DSA Coding Questions>java Matrix
Input:
1 0 0 1
0 0 1 0
0 0 0 0

Output:
1 1 1 1
1 1 1 1
1 0 1 1

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (r*c*(r+c))
Space Complexity: O (1)

10. Print a given matrix in spiral form
Given an m x n matrix, the task is to print all elements of the matrix in spiral form.

Input: matrix = {{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12},
{13, 14, 15, 16 }}

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Input: matrix = {{1, 2, 3, 4, 5, 6},
{7, 8, 9, 10, 11, 12},
{13, 14, 15, 16, 17, 18}}
Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
Explanation: The output is matrix in spiral format.

Code Solution:

```
// Spiral Matrix

import java.util.ArrayList;
import java.util.List;

public class SpiralMatrix {
   public List<Integer> spiralOrder(int[][] matrix) {
      List<Integer> res=new ArrayList<>();
      int l=0, r=matrix[0].length;
      int t=0, b=matrix.length;

      while (l<r && t<b) {
         for (int i=l; i<r; i++) {
            res.add(matrix[t][i]);
         }
         t++;

         for (int i=t; i<b; i++) {
            res.add(matrix[i][r-1]);
         }
         r--;

         if (!(l<r && t<b)) {
            break;
         }

         for (int i=r-1; i>=l; i--) {
            res.add(matrix[b-1][i]);
         }
         b--;

         for (int i=b-1; i>=t; i--) {
            res.add(matrix[i][l]);
         }
         l++;
```

```
        }

        return res;
    }

    public static void main(String[] args) {
        SpiralMatrix spiralMatrix = new SpiralMatrix();
        int[][] matrix = {
            {1, 2, 3, 4},
            {5, 6, 7, 8},
            {9, 10, 11, 12},
            {13, 14, 15, 16}
        };

        List<Integer> result = spiralMatrix.spiralOrder(matrix);
        System.out.println("Spiral order of the matrix: " + result);
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac SpiralMatrix.java

C:\Users\Legion\Desktop\DSA Coding Questions>java SpiralMatrix
Spiral order of the matrix: [1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10]

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (r*c)
Space Complexity: O (r*c)

13. Check if given Parentheses expression is balanced or not
Given a string str of length N, consisting of “(” and “)” only, the task is to check whether it is balanced or not.

Input: str = “((()))()()”
Output: Balanced
Input: str = “())((())”
Output: Not Balanced

Code Solution:

```
// balancing parentheses

import java.util.Stack;

public class BalancedParentheses {
    public static boolean isBalanced(String str) {
```

```
        Stack<Character> stack=new Stack<>();
        for (char ch:str.toCharArray()) {
            if (ch=='(') {
                stack.push(ch);
            }
                        else if (ch==')') {
                if (stack.isEmpty()) {
                                    return false;
                }
                stack.pop();
            }
        }
        return stack.isEmpty();
    }

    public static void main(String[] args) {
        String str1="((()))()()";
        String str2="())((())";

        System.out.println("Input: "+str1);
        System.out.println("Output: "+(isBalanced(str1)?"Balanced":"Not Balanced"));

        System.out.println("Input: "+str2);
        System.out.println("Output: "+(isBalanced(str2)?"Balanced":"Not Balanced"));
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Legion\Desktop\DSA Coding Questions>javac BalancedParentheses.java

C:\Users\Legion\Desktop\DSA Coding Questions>java BalancedParentheses
Input: ((()))()()
Output: Balanced
Input: ())((())
Output: Not Balanced

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (n)

14. Check if two Strings are Anagrams of each other

Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different.

Input: s1 = “geeks” s2 = “kseeg”
Output: true
Explanation: Both the string have same characters with same frequency. So, they are anagrams.
Input: s1 = “allergy” s2 = “allergic”
Output: false
Explanation: Characters in both the strings are not same. s1 has extra character „y“ and s2 has extra characters „i“ and „c“, so they are not anagrams.
Input: s1 = “g”, s2 = “g”
Output: true
Explanation: Characters in both the strings are same, so they are anagrams.

Code Solution:

```
// Anagram checking for s1 and s2
import java.util.Arrays;

public class AnagramChecker {
   public static boolean areAnagrams(String s1, String s2) {
      if (s1.length()!=s2.length()) {
         return false;
      }
      char[] arr1=s1.toCharArray();
      char[] arr2=s2.toCharArray();

      Arrays.sort(arr1);
      Arrays.sort(arr2);

      return Arrays.equals(arr1,arr2);
   }

   public static void main(String[] args) {
      String s1="geeks";
      String s2="kseeg";
      System.out.println("Input: s1="+s1+", s2="+s2);
      System.out.println("Output: "+areAnagrams(s1,s2));

      s1="allergy";
      s2="allergic";
      System.out.println("Input: s1="+s1+", s2="+s2);
      System.out.println("Output: "+areAnagrams(s1,s2));
```

```
        s1="g";
        s2="g";
        System.out.println("Input: s1="+s1+", s2="+s2);
        System.out.println("Output: "+areAnagrams(s1,s2));
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Legion\Desktop\DSA Coding Questions>javac AnagramChecker.java

C:\Users\Legion\Desktop\DSA Coding Questions>java AnagramChecker
Input: s1=geeks, s2=kseeg
Output: true
Input: s1=allergy, s2=allergic
Output: false
Input: s1=g, s2=g
Output: true

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n log n)
Space Complexity: O (n)

15. Longest Palindromic Substring
Given a string str, the task is to find the longest substring which is a palindrome. If there are multiple answers, then return the first appearing substring.
Input: str = "forgeeksskeegfor"
Output: "geeksskeeg"
Explanation: There are several possible palindromic substrings like "kssk", "ss", "eeksskee" etc. But the substring "geeksskeeg" is the longest among all.

Input: str = "Geeks"
Output: "ee"
Input: str = "abc"
Output: "a"
Input: str = ""
Output: ""

Code Solution:

```
//Longest plaindromic Substring

public class LongestPalindromicSubstring {
   public static String longestPalindrome(String str) {
```

```
    if (str==null || str.length()<1) {
        return "";
    }

    int start=0;
            int end=0;

    for (int i=0; i<str.length(); i++)
            {
        int len1=expandAroundCenter(str,i,i);
        int len2=expandAroundCenter(str,i,i+1);
        int len=Math.max(len1,len2);

        if (len>end-start)
                    {
            start=i-(len-1)/2;
            end=i+len/2;
        }
    }

    return str.substring(start, end+1);
}

private static int expandAroundCenter(String str, int l, int r) {
    while (l>=0 && r<str.length() && str.charAt(l)==str.charAt(r))
            {
        l--;
        r++;
    }
    return r-l-1;
}

public static void main(String[] args) {
    String str1="forgeeksskeegfor";
    String str2="Geeks";
    String str3="abc";
    String str4="";

    System.out.println("Input: "+str1);
    System.out.println("Output: "+longestPalindrome(str1));

    System.out.println("Input: "+str2);
    System.out.println("Output: "+longestPalindrome(str2));
```

```
        System.out.println("Input: "+str3);
        System.out.println("Output: "+longestPalindrome(str3));

        System.out.println("Input: "+str4);
        System.out.println("Output: "+longestPalindrome(str4));
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Legion\Desktop\DSA Coding Questions>javac LongestPalindromicSubstring.java

C:\Users\Legion\Desktop\DSA Coding Questions>java LongestPalindromicSubstring
Input: forgeeksskeegfor
Output: geeksskeeg
Input: Geeks
Output: ee
Input: abc
Output: c
Input:
Output:

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n^2)
Space Complexity: O (1)

16. Longest Common Prefix using Sorting

Given an array of strings arr[]. The task is to return the longest common prefix among each and every strings present in the array. If there‟s no prefix common in all the strings, return “-1”.

Input: arr[] = [“geeksforgeeks”, “geeks”, “geek”, “geezer”]
Output: gee
Explanation: “gee” is the longest common prefix in all the given strings.
Input: arr[] = [“hello”, “world”]
Output: -1
Explanation: There‟s no common prefix in the given strings.

Code Solution:

```
//Longest common prefix using sorting

import java.util.Arrays;

public class LongestCommonPrefix {
    public static String longestCommonPrefix(String[] arr) {
        if (arr==null || arr.length==0) {
```

```
            return "-1";
        }

        Arrays.sort(arr);

        String first=arr[0];
        String last=arr[arr.length-1];
        int minLength=Math.min(first.length(),last.length());

        int i=0;
        while (i<minLength && first.charAt(i)==last.charAt(i)) {
            i++;
        }

        String commonPrefix=first.substring(0,i);
        return commonPrefix.isEmpty()?"-1":commonPrefix;
    }

    public static void main(String[] args) {
        String[] arr1={"geeksforgeeks","geeks","geek","geezer"};
        String[] arr2={"hello","world"};

        System.out.println("Input: "+Arrays.toString(arr1));
        System.out.println("Output: "+longestCommonPrefix(arr1));

        System.out.println("Input: "+Arrays.toString(arr2));
        System.out.println("Output: "+longestCommonPrefix(arr2));
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Legion\Desktop\DSA Coding Questions>javac LongestCommonPrefix.java

C:\Users\Legion\Desktop\DSA Coding Questions>java LongestCommonPrefix
Input: [geeksforgeeks, geeks, geek, geezer]
Output: gee
Input: [hello, world]
Output: -1

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n log n)

Space Complexity: O (1)

17. Delete middle element of a stack

Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure.

Input : Stack[] = [1, 2, 3, 4, 5]
Output : Stack[] = [1, 2, 4, 5]
Input : Stack[] = [1, 2, 3, 4, 5, 6]
Output : Stack[] = [1, 2, 4, 5, 6]

Code Solution:

```
//Delete Middle element of a stack

import java.util.Scanner;
import java.util.Stack;

public class DeleteMiddleElement {

   public static void deleteMiddle(Stack<Integer> stack, int currentIndex, int
middleIndex) {
      if (stack.isEmpty() || currentIndex==middleIndex) {
         stack.pop();
         return;
      }

      int temp=stack.pop();
      deleteMiddle(stack, currentIndex+1, middleIndex);
      stack.push(temp);
   }

   public static void deleteMiddle(Stack<Integer> stack) {
      if (!stack.isEmpty())
               {
         int middleIndex=stack.size()/2;
         deleteMiddle(stack, 0, middleIndex);
      }
   }

   public static void main(String[] args) {
      Scanner scanner=new Scanner(System.in);
      Stack<Integer> stack=new Stack<>();

      System.out.println("Enter the number of elements in the stack:");
      int n=scanner.nextInt();
```

```
        System.out.println("Enter the elements of the stack:");
        for (int i=0; i<n; i++) {
            int element=scanner.nextInt();
            stack.push(element);
        }

        System.out.println("Original Stack: "+stack);
        deleteMiddle(stack);
        System.out.println("Stack after deleting middle element: "+stack);

        scanner.close();
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>java DeleteMiddleElement
Enter the number of elements in the stack:
5
Enter the elements of the stack:
1
2
3
4
5
Original Stack: [1, 2, 3, 4, 5]
Stack after deleting middle element: [1, 2, 4, 5]

C:\Users\Legion\Desktop\DSA Coding Questions>
```

```
C:\Users\Legion\Desktop\DSA Coding Questions>java DeleteMiddleElement
Enter the number of elements in the stack:
1
Enter the elements of the stack:
42
Original Stack: [42]
Stack after deleting middle element: []

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (n)

18. Next Greater Element (NGE) for every element in given Array
Given an array, print the Next Greater Element (NGE) for every element.
Note: The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Input: arr[] = [4 , 5 , 2 , 25]
Output: 4 –> 5
5 –> 25
2 –> 25
25 –> -1
Explanation: Except 25 every element has an element greater than them present on the right side
Input: arr[] = [13 , 7, 6 , 12]
Output: 13 –> -1
7 –> 12
6 –> 12
12 –> -1
Explanation: 13 and 12 don‟t have any element greater than them present on the right side

Code Solution:

```
// find the next greater element

import java.util.Stack;
import java.util.HashMap;

public class NextGreaterElement {
   public static void printNextGreaterElements(int[] arr) {
      Stack<Integer> stack=new Stack<>();
      HashMap<Integer, Integer> ngeMap=new HashMap<>();

      for (int i=arr.length-1; i>=0; i--) {
         int current=arr[i];

         while (!stack.isEmpty() && stack.peek()<=current) {
            stack.pop();
         }

         int nextGreater=stack.isEmpty()?-1:stack.peek();
         ngeMap.put(current, nextGreater);

         stack.push(current);
      }
      for (int num:arr) {
         System.out.println(num+" -> "+ngeMap.get(num));
      }
   }

   public static void main(String[] args) {
```

```
        int[] arr1={4, 5, 2, 25};
        System.out.println("Array: "+java.util.Arrays.toString(arr1));
        printNextGreaterElements(arr1);

        int[] arr2={13, 7, 6, 12};
        System.out.println("\nArray: "+java.util.Arrays.toString(arr2));
        printNextGreaterElements(arr2);
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Legion\Desktop\DSA Coding Questions>javac NextGreaterElement.java

C:\Users\Legion\Desktop\DSA Coding Questions>java NextGreaterElement
Array: [4, 5, 2, 25]
4 -> 5
5 -> 25
2 -> 25
25 -> -1

Array: [13, 7, 6, 12]
13 -> -1
7 -> 12
6 -> 12
12 -> -1

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (n)

19. Print Right View of a Binary Tree
Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

Code Solution:

```
// Print Right View of the Binary Tree

import java.util.*;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode() {}
```

```
    TreeNode(int val) {
        this.val=val;
    }

    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val=val;
        this.left=left;
        this.right=right;
    }
}

public class BinaryTreeRightView {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> result=new ArrayList<>();
        if (root==null) {
            return result;
        }

        Queue<TreeNode> queue=new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int levelSize=queue.size();
            TreeNode rightNode=null;

            for (int i=0; i<levelSize; i++) {
                TreeNode curr=queue.poll();
                rightNode=curr;

                if (curr.left!=null) {
                    queue.add(curr.left);
                }
                if (curr.right!=null) {
                    queue.add(curr.right);
                }
            }

            result.add(rightNode.val);
        }

        return result;
    }
```

```
    public static void main(String[] args) {
        TreeNode root=new TreeNode(1);
        root.left=new TreeNode(2);
        root.right=new TreeNode(3);
        root.left.left=new TreeNode(4);
        root.left.right=new TreeNode(5);
        root.right.right=new TreeNode(6);
        root.left.left.right=new TreeNode(7);

        BinaryTreeRightView  solution=new BinaryTreeRightView ();
        List<Integer> rightView=solution.rightSideView(root);

        System.out.println("Right View of the Binary Tree: "+rightView);
    }
}
```

Output:

```
C:\Users\Legion\Desktop\DSA Coding Questions>javac BinaryTreeRightView.java

C:\Users\Legion\Desktop\DSA Coding Questions>java BinaryTreeRightView
Right View of the Binary Tree: [1, 3, 6, 7]

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (n)

20. Maximum Depth or Height of Binary Tree
Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

Code Solution:

```
//Maximum Depth of Tree

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode() {}

    TreeNode(int val) {
```

```
        this.val=val;
    }

    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val=val;
        this.left=left;
        this.right=right;
    }
}

public class BinaryTreeHeight {
    public int maxDepth(TreeNode root) {
        if (root==null) {
            return 0;
        }

        int leftHeight=maxDepth(root.left);
        int rightHeight=maxDepth(root.right);

        return Math.max(leftHeight,rightHeight)+1;
    }

    public static void main(String[] args) {
        TreeNode root=new TreeNode(1);
        root.left=new TreeNode(2);
        root.right=new TreeNode(3);
        root.left.left=new TreeNode(4);
        root.left.right=new TreeNode(5);
        root.right.right=new TreeNode(6);
        root.left.left.right=new TreeNode(7);

        BinaryTreeHeight tree=new BinaryTreeHeight();
        int height=tree.maxDepth(root);

        System.out.println("Maximum Depth of the Binary Tree: "+height);
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Legion\Desktop\DSA Coding Questions>javac BinaryTreeHeight.java

C:\Users\Legion\Desktop\DSA Coding Questions>java BinaryTreeHeight
Maximum Depth of the Binary Tree: 4

C:\Users\Legion\Desktop\DSA Coding Questions>
```

Time Complexity: O (n)
Space Complexity: O (n)