1. Maximum Subarray Sum – Kadane‟s Algorithm:

Code

import java.util.\*;

public class MaximumSubArraySum{

    public static void main(String[] args){

        System.out.print("Enter the size of the array : ");

        Scanner sc=new Scanner(System.in);

        int l=sc.nextInt();

        int sums=0;

        int maxi=Integer.MIN\_VALUE;

        for(int i=0;i<l;i++){

            int val=sc.nextInt();

            sums+=val;

            if(sums>maxi) maxi=sums;

            if (sums<0) sums=0;

        }

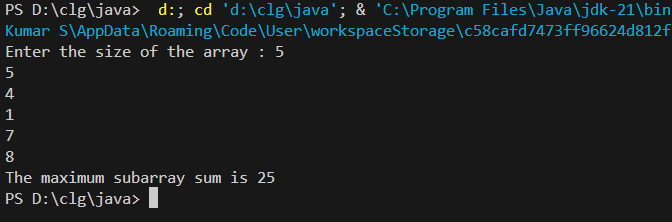
        System.out.println("The maximum subarray sum is "+maxi);

        sc.close();

}

}

Output:



Time complexity : o(n)

2.Maximum Product Subarray

Code

import java.util.\*;

public class MaximumSubProductSum{

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        System.out.print("Enter the array ");

        String S=sc.nextLine();

        String[] lst=S.split(" ");

        int maxi=Integer.MIN\_VALUE;

        for(int i=0;i<lst.length;i++){

            int temp=1;

            for (int j=i;j<lst.length;j++){

                temp\*=Integer.parseInt(lst[j]);

            if(temp>maxi) maxi=temp;

            }

        }

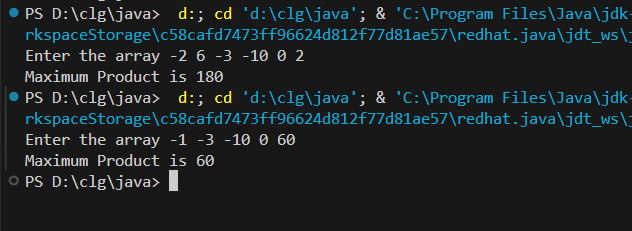
        System.out.println("Maximum Product is "+maxi);

        sc.close();

    }

}

Output



Time Complexity : o(n2)

3. . Search in a sorted and rotated Array

Code

import java.util.\*;

public class serachInRoated{

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        System.out.print("Enter the Array : ");

        String s=sc.nextLine();

        System.out.print("Enter the Targer : ");

        int target=sc.nextInt();

        String[] slst=s.split(" ");

        int[] lst=new int[slst.length];

        for(int i=0;i<lst.length;i++){

            lst[i]=Integer.parseInt(slst[i]);

        }

        int start=0;

        int end=lst.length-1;

        boolean flag=false;

        int r=0;

        while(start<=end){

            int mid=(start+end)/2;

            if(lst[mid]==target){

                flag=true;

                r=mid;

                break;

            }

            else if(lst[mid]<target){

                if(lst[start]<=target && lst[mid-1]>=target) end=mid-1;

                else start=mid+1;

            }

            else if(lst[mid]>target){

                if(lst[mid+1]<=target && lst[end]>=target) start=mid+1;

                else end=mid-1;

            }

        }

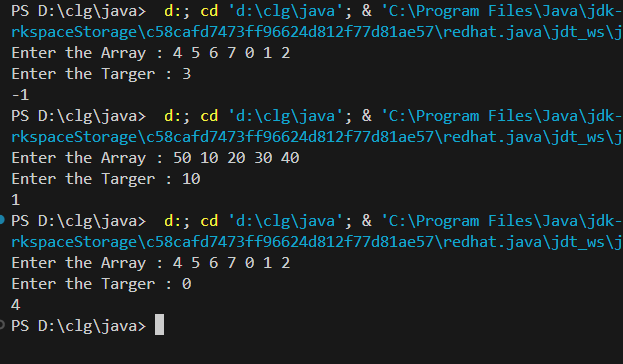
        if(flag) System.out.println(r);

        else System.out.println(-1);

    }

}

Output



Time Complexity : O(nlogn)

4. Container with Most Water

Code

import java.util.\*;

public class MostWater{

    public static void main(String[] args){

        System.out.print("Enter the Array :");

        Scanner sc=new Scanner(System.in);

        String s=sc.nextLine();

        String[] slst=s.split(" ");

        int n=slst.length;

        int[] lst=new int[n];

        int maxi=Integer.MIN\_VALUE;

        for(int i=0; i<n; i++) lst[i]=Integer.parseInt(slst[i]);

        int start=0;

        int end=lst.length-1;

        while(start<end){

            int area=Math.min(lst[start],lst[end])\*(end -start);

            maxi=Math.max(maxi,area);

            if(lst[start]<lst[end]) start++;

            else end--;

        }

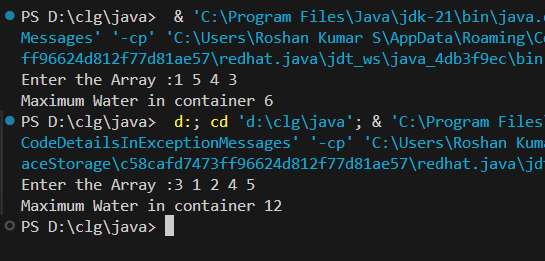
        System.out.println("Maximum Water in container "+maxi);

        sc.close();

    }

}

Output



Time Complexity : O(n)

5. Find the Factorial of a large number

Code

import java.math.BigInteger;

import java.util.\*;

public class Factorial{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

System.out.print("Enter the number : ");

long n=sc.nextLong();

BigInteger fact=BigInteger.ONE;

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

fact=fact.multiply(BigInteger.valueOf(i));

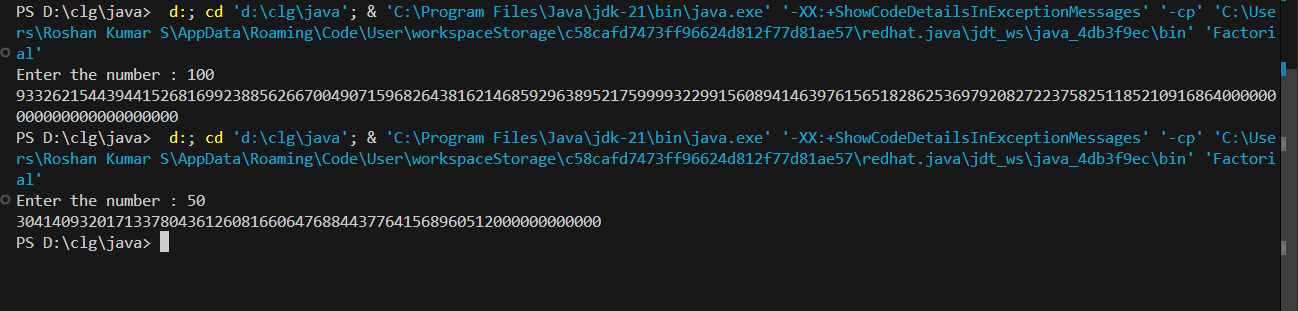
}

System.out.println(fact);

}

}

Output



Time Complexity = O(n)

6.

**Code**  
import java.util.\*;

public class waterTrap{

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the Array : ");

        String s=sc.nextLine();

        String[] slst=s.split(" ");

        int[] lst=new int[slst.length];

        for(int i=0;i<lst.length;i++){

            lst[i]=Integer.parseInt(slst[i]);

        }

        int l =0;

        int r = lst.length - 1;

        int lM = lst[l];

        int rM = lst[r];

        int water = 0;

        while (l < r){

            if (lM < rM) {

                l++;

                lM = Math.max(lM, lst[l]);

                water += lM - lst[l];

            } else {

                r--;

                rM = Math.max(rM, lst[r]);

                water += rM - lst[r];

            }

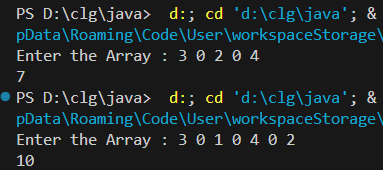
        }

        System.out.println(water);

    }

}

Output



**Time Complexity: O(n)**

7.

**Code**

import java.util.\*;

public class ChocolateDis{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        ArrayList<Integer> lst = new ArrayList<Integer>();

        System.out.print("Enter the array : ");

        String sarry = sc.nextLine();

        String[] arr = sarry.split(" ");

        for (int i = 0; i < arr.length; i++) {lst.add(Integer.parseInt(arr[i]));}

        System.out.print("Enter the m :");

        int m= sc.nextInt();

        int n = lst.size();

        Collections.sort(lst);

        int md=Integer.MAX\_VALUE;

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

            int diff = lst.get(i + m - 1) - lst.get(i);

            md = Math.min(md, diff);

        }

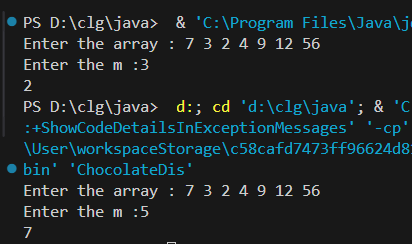
        System.out.println(md);

    }

}

}

**Output**



**Time Complexity :** O(nlogn)

8.

**Code**

import java.util.\*;

public class MergeInterval {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of intervals: ");

        int n = sc.nextInt();

        int[][] arr = new int[n][2];

        System.out.println("Enter the intervals:");

        for (int i = 0; i < n; i++) {

            arr[i][0] = sc.nextInt();

            arr[i][1] = sc.nextInt();

        }

        if (arr.length <= 1) {

            System.out.println("Merged intervals:");

            for (int[] i : arr) {

                System.out.println(Arrays.toString(i));

            }

        } else {

            Arrays.sort(arr, (a, b) -> Integer.compare(a[0], b[0]));

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

            int[] curr = arr[0];

            merged.add(curr);

            for (int[] i : arr) {

                int currEnd = curr[1];

                int nextStart = i[0];

                int nextEnd = i[1];

                if (currEnd >= nextStart) {

                    curr[1] = Math.max(currEnd, nextEnd);

                } else {

                    curr = i;

                    merged.add(curr);

                }

            }

            System.out.println("Merged intervals:");

            for (int[] m : merged) {

                System.out.println(Arrays.toString(m));

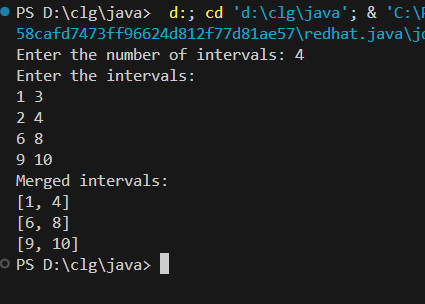
            }

        }

    }

}

**Output**

****

**Time Complexity :** O(nlogn)

9.

**Code**

import java.util.\*;

public class ModifyMatrix {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the Number of Rows:");

        int r = sc.nextInt();

        System.out.print("Enter the Number of Columns:");

        int c = sc.nextInt();

        int[][] matrix = new int[r][c];

        System.out.println("Enter the Matrix Values");

        for (int i = 0; i < r; i++) {

            for (int j = 0; j < c; j++) {

                matrix[i][j] = sc.nextInt();

            }

        }

        boolean[] rowm = new boolean[r];

        boolean[] colm = new boolean[c];

        for (int i = 0; i < r; i++) {

            for (int j = 0; j < c; j++) {

                if (matrix[i][j] == 1) {

                    rowm[i] = true;

                    colm[j] = true;

                }

            }

        }

        for (int i = 0; i < r; i++) {

            for (int j = 0; j < c; j++) {

                if (rowm[i] || colm[j]) {

                    matrix[i][j] = 1;

                }

            }

        }

        System.out.println("Modified Matrix");

        for (int i = 0; i < r; i++) {

            for (int j = 0; j < c; j++) {

                System.out.print(matrix[i][j] + " ");

            }

            System.out.println();

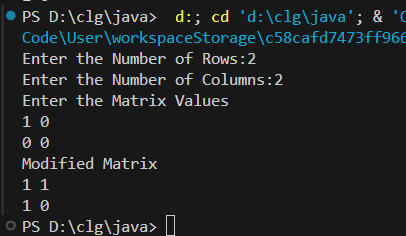
        }

        sc.close();

    }

}

**Output**

****

**Time Complexity :** O(n \* m)

**10. Check if given Parentheses expression is balanced or not**

**Code**

import java.util.\*;

public class SpiralForm {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of rows:");

        int rows = sc.nextInt();

        System.out.print("Enter the number of columns:");

        int cols = sc.nextInt();

        int[][] matrix = new int[rows][cols];

        System.out.println("Enter the matrix values:");

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

            for (int j = 0; j < cols; j++) {

                matrix[i][j] = sc.nextInt();

            }

        }

        ArrayList<Integer> result = new ArrayList<>();

        int top = 0, left = 0, bottom = rows - 1, right = cols - 1;

        while (top <= bottom && left <= right) {

            for (int i = left; i <= right; i++)

                result.add(matrix[top][i]);

            top++;

            for (int i = top; i <= bottom; i++)

                result.add(matrix[i][right]);

            right--;

            if (top <= bottom) {

                for (int i = right; i >= left; i--)

                    result.add(matrix[bottom][i]);

                bottom--;

            }

            if (left <= right) {

                for (int i = bottom; i >= top; i--)

                    result.add(matrix[i][left]);

                left++;

            }

        }

        for (int num : result) {

            System.out.print(num + " ");

        }

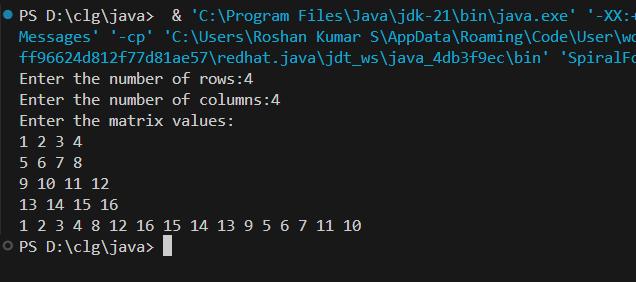
        System.out.println();

        sc.close();

    }

}

**Output**



**Time Complexity :** O(n\*m)

**13 . Check if given Parentheses expression is balanced or not**

**Code**

import java.util.\*;

public class CheckBalanceParentheses {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the expression:");

        String expression = sc.nextLine();

        boolean result = true;

        int balance = 0;

        for (int i = 0; i < expression.length(); i++) {

            if (expression.charAt(i) == '(') {

                balance++;

            } else {

                balance--;

            }

            if (balance < 0) {

                result = false;

                break;

            }

        }

        if (balance != 0) {

            result = false;

        }

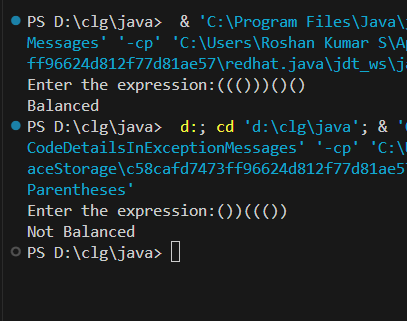
        System.out.println(result ? "Balanced" : "Not Balanced");

        sc.close();

    }

}

**Output**



**Time Complexity :** O(n)

**14.** **Check if two Strings are Anagrams of each other**

**Code**

import java.util.\*;

public class AnagaramStr {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the first string:");

        String str1 = sc.nextLine();

        System.out.print("Enter the second string:");

        String str2 = sc.nextLine();

        HashMap<Character, Integer> CountM = new HashMap<>();

        for (char i : str1.toCharArray())  CountM.put(i, CountM.getOrDefault(i, 0) + 1);

        for (char i : str2.toCharArray())  CountM.put(i, CountM.getOrDefault(i, 0) - 1);

        boolean flag = true;

        for (var val : CountM.entrySet()) {

            if (val.getValue() != 0) {

                flag = false;

                break;

            }

        }

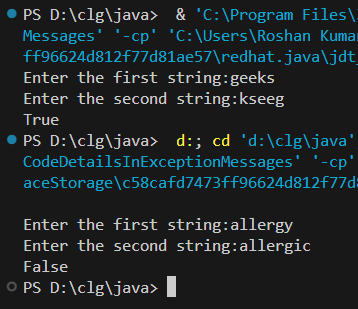
        System.out.println(flag ? "True" : "False");

        sc.close();

    }

}

**Output**



**Time Complexity :** O(n+m)

**15.** **Longest Palindromic Substring**

**Code**

import java.util.\*;

public class LogestPalinrome {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a string:");

        String s = sc.nextLine();

        int len = s.length();

        boolean[][] dp = new boolean[len][len];

        int maxLen = 1;

        int startIdx = 0;

        for (int i = 0; i < len; ++i)

            dp[i][i] = true;

        for (int i = 0; i < len - 1; ++i) {

            if (s.charAt(i) == s.charAt(i + 1)) {

                dp[i][i + 1] = true;

                startIdx = i;

                maxLen = 2;

            }

        }

        for (int k = 3; k <= len; ++k) {

            for (int i = 0; i < len - k + 1; ++i) {

                int j = i + k - 1;

                if (dp[i + 1][j - 1] && s.charAt(i) == s.charAt(j)) {

                    dp[i][j] = true;

                    if (k > maxLen) {

                        startIdx = i;

                        maxLen = k;

                    }

                }

            }

        }

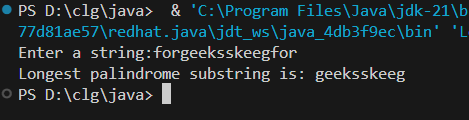
        System.out.println("Longest palindrome substring is: " + s.substring(startIdx, startIdx + maxLen));

        sc.close();

    }

}

**Output:**



**Time Complexity :** O(n2)

**16.** **. Longest Common Prefix using Sorting**

**Code**

import java.util.\*;

public class LongestPalindromeSubstring {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of strings:");

        int n = sc.nextInt();

        String[] arr = new String[n];

        sc.nextLine();

        System.out.println("Enter the strings:");

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextLine();

        }

        if (arr == null || arr.length == 0) {

            System.out.println("-1");

            return;

        }

        Arrays.sort(arr);

        String f = arr[0];

        String l = arr[arr.length - 1];

        int minLen = Math.min(f.length(), l.length());

        int i = 0;

        while (i < minLen && f.charAt(i) == l.charAt(i)) {

            i++;

        }

        if (i == 0) {

            System.out.println("-1");

        } else {

            System.out.println(f.substring(0, i));

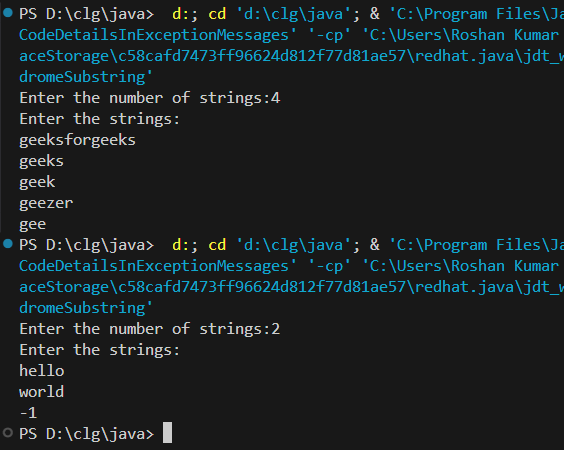
        }

        sc.close();

    }

}

**Output**

****

**Time Complexity :** O(n log n + m)

**17.** **Delete middle element of a stack**

**Code**

import java.util.\*;

public class DeleteMidElement {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        Stack<Character> s = new Stack<>();

        System.out.println("Enter the stack elements:");

        String[] e = sc.nextLine().split(" ");

        for (String x : e) {

            if (!x.isEmpty()) {

                s.push(x.charAt(0));

            }

        }

        Vector<Character> v = new Vector<>();

        while (!s.empty()) {

            v.add(s.pop());

        }

        int l = v.size();

        int m = l / 2;

        for (int i = 0; i < l; i++) {

            if (l % 2 == 0 && i == m || l % 2 != 0 && i == m) continue;

            s.push(v.get(i));

        }

        while (!s.empty()) {

            System.out.print(s.pop() + " ");

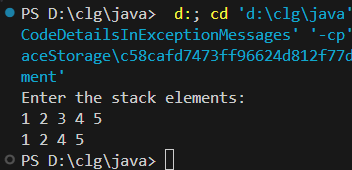
        }

        sc.close();

    }

}

**Output**

****

**Time Complexity :**O(n)

**18.** **Next Greater Element (NGE) for every element in given Array**

**Code**

import java.util.\*;

public class NextGreater {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of elements in the array: ");

        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter the elements of the array:");

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextInt();

        }

        int[] nge = new int[arr.length];

        Arrays.fill(nge, -1);

        Stack<Integer> stack = new Stack<>();

        for (int i = 0; i < arr.length; i++) {

            while (!stack.isEmpty() && arr[i] > arr[stack.peek()]) {

                int idx = stack.pop();

                nge[idx] = arr[i];

            }

            stack.push(i);

        }

        for (int i = 0; i < arr.length; i++) {

            System.out.println(arr[i] + " --> " + nge[i]);

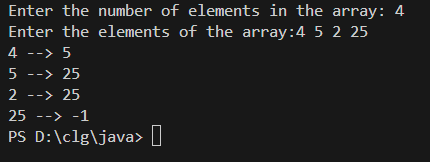
        }

        sc.close();

    }

}

**Output**

****

**Time Complexity :** O(n)

**19.Print Right View of a Binary Tree**

**Code:**

import java.util.ArrayList;

import java.util.Scanner;

class Element {

    int value;

    Element leftChild, rightChild;

    Element(int x) {

        value = x;

        leftChild = rightChild = null;

    }

}

public class MainProgram {

    public static void main(String[] args) {

        Element root = createTree();

        ArrayList<Integer> view = viewFromRight(root);

        System.out.print("Right view of the binary tree: ");

        displayList(view);

    }

    static void exploreRightView(Element root, int depth, int[] maxDepth, ArrayList<Integer> view) {

        if (root == null) return;

        if (depth > maxDepth[0]) {

            view.add(root.value);

            maxDepth[0] = depth;

        }

        exploreRightView(root.rightChild, depth + 1, maxDepth, view);

        exploreRightView(root.leftChild, depth + 1, maxDepth, view);

    }

    static ArrayList<Integer> viewFromRight(Element root) {

        ArrayList<Integer> view = new ArrayList<>();

        int[] maxDepth = new int[]{-1};

        exploreRightView(root, 0, maxDepth, view);

        return view;

    }

    static void displayList(ArrayList<Integer> list) {

        for (int num : list) {

            System.out.print(num + " ");

        }

        System.out.println();

    }

    static Element createTree() {

        Scanner input = new Scanner(System.in);

        System.out.println("Enter the number of elements:");

        int count = input.nextInt();

        if (count <= 0) {

            return null;

        }

        Element[] elements = new Element[count];

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

            elements[i] = new Element(i + 1);

        }

        System.out.println("Enter relations as (parent, child, L/R):");

        for (int i = 0; i < count - 1; i++) {

            int parentIdx = input.nextInt() - 1;

            int childIdx = input.nextInt() - 1;

            char direction = input.next().charAt(0);

            if (direction == 'L' || direction == 'l') {

                elements[parentIdx].leftChild = elements[childIdx];

            } else if (direction == 'R' || direction == 'r') {

                elements[parentIdx].rightChild = elements[childIdx];

            }

        }

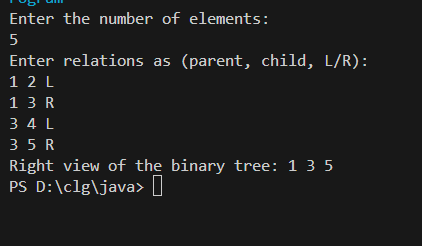
        input.close();

        return elements[0];

    }

}

**Output**

****

**Time Complexity : O(n)**

**20.Maximum Depth or Height of Binary Tree**

**Code**

class TreeNode {

    int val;

    TreeNode left, right;

    TreeNode(int x) {

        val = x;

        left = right = null;

    }

}

public class TreeDepthCalculator {

    public int maxDepth(TreeNode root) {

        if (root == null) {

            return 0;

        }

        int leftDepth = maxDepth(root.left);

        int rightDepth = maxDepth(root.right);

        return Math.max(leftDepth, rightDepth) + 1;

    }

    public static void main(String[] sasta) {

        TreeNode root = new TreeNode(1);

        root.left = new TreeNode(2);

        root.right = new TreeNode(3);

        root.left.left = new TreeNode(4);

        root.right.left = new TreeNode(5);

        root.right.right = new TreeNode(6);

        root.right.right.left = new TreeNode(7);

        TreeDepthCalculator calculator = new TreeDepthCalculator();

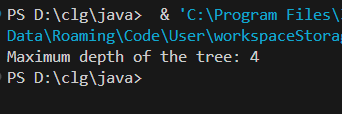
        int depth = calculator.maxDepth(root);

        System.out.println("Maximum depth of the tree: " + depth);

    }

}

**Output**

****