PRACTICE – 1

1.Maximum Subarray Sum – Kadane's Algorithm: Given an array arr[], the task is to find the subarraythat has the maximum sum and return its sum.

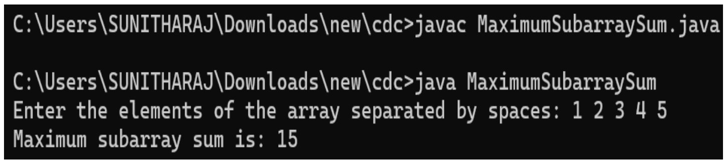
Solution:

```
import java.util.Scanner;
public class MaximumSubarraySum{
  public static int maxSubArraySum(int[] arr) {
    int maxSoFar = arr[0];
    int maxEndingHere = arr[0];

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

    return maxSoFar;
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the elements of the array separated by spaces: ");
    String inputLine = scanner.nextLine();
    String[] inputStrings = inputLine.split(" ");
    int[] arr = new int[inputStrings.length];
    for (int i = 0; i < inputStrings.length; i++) {
      arr[i] = Integer.parseInt(inputStrings[i]);
    }
    int maxSum = maxSubArraySum(arr);
    System.out.println("Maximum subarray sum is: " + maxSum);
    scanner.close();
  }
}
```

TEST CASE :

Test Case	Input	Output
1	1 2 3 4 5	15
2	-2 -3 -4 -1 -2	-1

OUTPUT :



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac MaximumSubarraySum.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java MaximumSubarraySum
Enter the elements of the array separated by spaces: 1 2 3 4 5
Maximum subarray sum is: 15
```

TIME COMPLEXITY : O(n)

2.Maximum Product Subarray Given an integer array, the task is to find the maximum product of anysubarray.

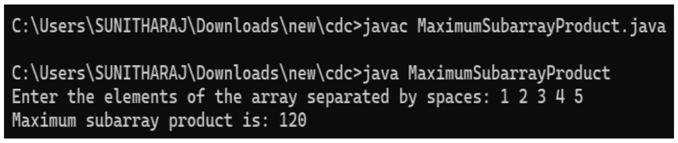
Solution:

```
import java.util.Scanner;
public class MaximumSubarrayProduct{
     public static int maxSubArrayProduct(int[] arr) {
        int maxProduct = arr[0];
        int minProduct = arr[0];
        int result = arr[0];
       for (int i = 1; i < arr.length; i++) {
           if (arr[i] < 0) {
               int temp = maxProduct;
               maxProduct = minProduct;
               minProduct = temp;
           }
           maxProduct = Math.max(arr[i], maxProduct * arr[i]);
           minProduct = Math.min(arr[i], minProduct * arr[i]);
           result = Math.max(result, maxProduct);
        }

        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the elements of the array separated by spaces: ");
        String inputLine = scanner.nextLine();
        String[] inputStrings = inputLine.split(" ");
        int[] arr = new int[inputStrings.length];
        for (int i = 0; i < inputStrings.length; i++) {
           arr[i] = Integer.parseInt(inputStrings[i]);
        }
        int maxSum = maxSubArrayProduct(arr);
        System.out.println("Maximum subarray product is: " + maxSum);
       scanner.close();
    }
}
```

TEST CASE :

Test Case	Input	Output
1	1 2 3 4 5	120
2	-3 -2 4 -1 -2	48

OUTPUT :



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac MaximumSubarrayProduct.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java MaximumSubarrayProduct
Enter the elements of the array separated by spaces: 1 2 3 4 5
Maximum subarray product is: 120
```

TIME COMPLEXITY : O(n)

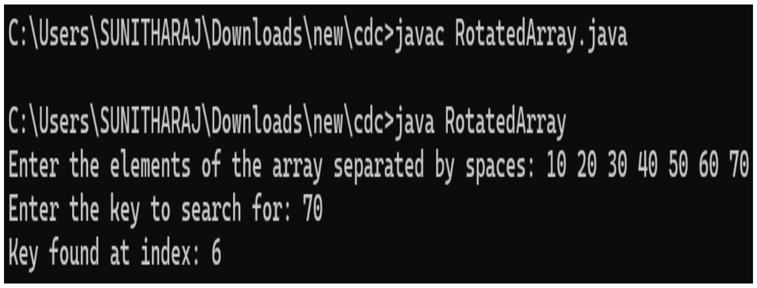
3.Search in a sorted and rotated Array Given a sorted and rotated array arr[] of n distinct elements, thetask is to find the index of given key in the array. If the key is not present in the array, return -1.

Solution:

```
import java.util.Scanner;
public class RotatedArray {
    public static int linearSearch(int[] arr, int key) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the elements of the array separated by spaces: ");
        String inputLine = scanner.nextLine();
        String[] inputStrings = inputLine.split(" ");
        int[] arr = new int[inputStrings.length];
        for (int i = 0; i < inputStrings.length; i++) {
            arr[i] = Integer.parseInt(inputStrings[i]);
        }
       System.out.print("Enter the key to search for: ");
        int key = scanner.nextInt();
        int index = linearSearch(arr, key);
        if (index != -1) {
            System.out.println("Key found at index: " + index);
        } else {
            System.out.println("-1");
        }
        scanner.close();
    }
}
```

Test Case 1:
Elements: 4 5 6 7 8 9 1 2 3
Key: 7
Output: Key found at index: 3

Test Case 2:
Elements:10 20 30 40 50 60 70 80 90
Key: 60
Output: Key found at index: 5



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac RotatedArray.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java RotatedArray
Enter the elements of the array separated by spaces: 10 20 30 40 50 60 70
Enter the key to search for: 70
Key found at index: 6
```

TIME COMPLEXITY : O(n)

4.Container with Most Water

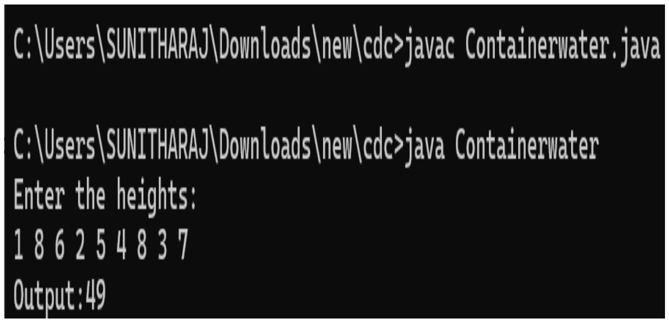
Solution:

```
import java.util.Scanner;
public class Containerwater{
    public static int maxArea(int[] arr) {
        int maxArea = 0;
        int left = 0;
        int right = arr.length - 1;
        while (left < right) {
            int height=Math.min(arr[left], arr[right]);
            int width=right - left;
            int area=height * width;
            maxArea=Math.max(maxArea, area);

            if (arr[left]<arr[right]) {
                left++;
            } else {
                right--;
            }
        }
        return maxArea;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the heights:");
        String inputLine = scanner.nextLine();
        String[] inputStrings = inputLine.split(" ");
        int[] arr = new int[inputStrings.length];
        for (int i = 0; i < inputStrings.length; i++) {
            arr[i] = Integer.parseInt(inputStrings[i]);
        }
        System.out.println("Output:"+maxArea(arr));
        scanner.close();

    }
```

TEST CASES :

Test Case 1:
Input: 1 8 6 2 5 4 8 3 7
Output: 49

Test Case 2:
Input: 1 1
Output: 1



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac Containerwater.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Containerwater
Enter the heights:
1 8 6 2 5 4 8 3 7
Output:49
```

TIME COMPLEXITY : O:(n)

5.Find the Factorial of a large number

Solution:

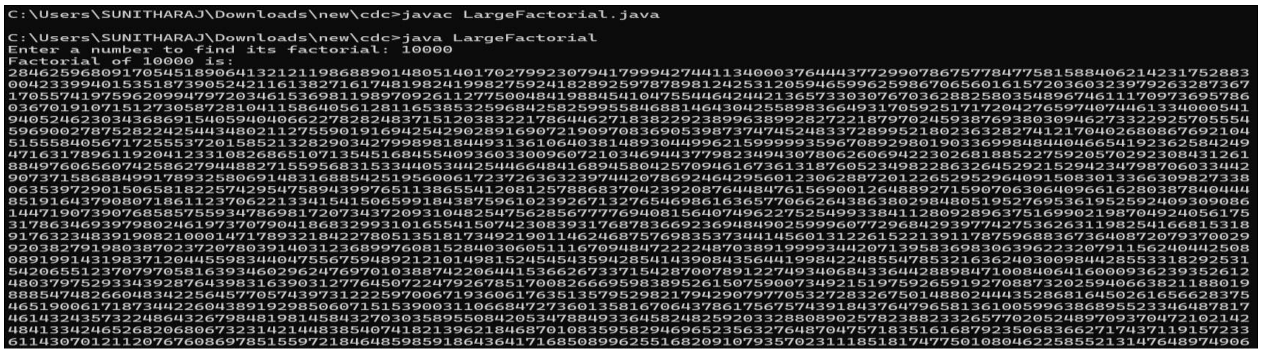
```
import java.math.BigInteger;
import java.util.Scanner;

public class LargeFactorial {

    public static BigInteger factorial(int n) {
        BigInteger result = BigInteger.ONE;

        for (int i = 1; i <= n; i++) {
            result = result.multiply(BigInteger.valueOf(i));
        }

        return result;
    }

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

        System.out.print("Enter a number to find its factorial: ");
        int number = scanner.nextInt();

        BigInteger factorialResult = factorial(number);

        System.out.println("Factorial of " + number + " is: ");
        System.out.println(factorialResult);

        scanner.close();

    }
}
```

Test Case : Input 1000

Input: 10000

Expected Output:2846259680917054431293015991698430074 52……

OUTPUT:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac LargeFactorial.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java LargeFactorial
Enter a number to find its factorial: 10000
Factorial of 10000 is:
284625968091705451890641321211986889014805140170279923079417999427441134000376444377299078675778477581588406214231752883
004233994015351873905242116138271617481982419982759241828925978789812425312059465996259867065601615720360323979263287367
170557419759620994797203461536981198970926112775004841988454104755446424421365733030767036288258035489674611170973695786
036701910715127305872810411586405612811653853259684258259955846881464304255898366493170592517172042765974074461334000541
940524623034368691540594040662278282483715120383221786446271838229238996389928272218797024593876938030946273322925705554
596900278752822425443480211275590191694254290289169072190970836905398737474524833728995218023632827412170402680867692104
515558405671725553720158521328290342799898184493136106403814893044996215999993596708929801903369984844046654192362584249
471631789611920412331082686510713545168455409360330096072103469443779823494307806260694223026818852275920570292308431261
884976065607425862794488271559568315334405344254466484168945804257094616736131876052349822863264529215294234798706033442
907371586884991789325806914831688542519560061723726363239744207869246429560123062887201226529529640915083013366309827338
063539729015065818225742954758943997651138655412081257886837042392087644847615690012648892715907063064096616280387840444
851916437908071861123706221334154150659918438759610239267132765469861636577066264386380298480519527695361952592409309086
144719073907685857559347869817207343720931048254756285677776940815640749622752549933841128092896375169902198704924056175
317863469397980246197370790418683299310165541507423083931768783669236948490259996077296842939774275362631198254166815318
917632348391908210001471789321842278051351817349219011462468757698353734414560131226152213911787596883673640872079370029
920382791980387023720780391403123689976081528403060511167094847222248703891999934420713958369830639622320791156240442508
089199143198371204455983440475567594892121014981524545435942854143908435644199842248554785321636240300984428553318292531
542065512370797058163934602962476970103887422064415366267337154287007891227493406843364428898471008406416000936239352612
480379752933439287643983163903127764507224792678517008266695983895261507590073492151975926591927088732025940663821188019
888547482660483422564577057439731222597006719360617635135795298217942907977053272832675014880244435286816450261656628375
465190061718734422604389192985060715153900311066847273601358167064378617567574391843764796581361005996386895523346487817
461432435732248643267984819814584327030358955084205347884933645824825920332880890257823882332657702052489709370472102142
484133424652682068067323142144838540741821396218468701083595829469652356327648704757183516168792350683662717437119157233
611430701211207676086978515597218464859859186436417168508996255168209107935702311185181747750108046225855213147648974906
```

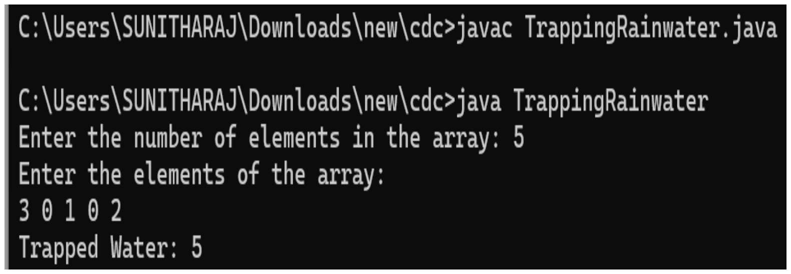
TIME COMPLEXITY : O(n)

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

Solution:

```
import java.util.Scanner;
public class TrappingRainwater {
   public static int trapWater(int[] arr) {
      int n = arr.length;

      if (n == 0) {
         return 0;
      }

      int[] leftMax = new int[n];
      int[] rightMax = new int[n];

      leftMax[0] = arr[0];
      for (int i = 1; i < n; i++) {
         leftMax[i] = Math.max(leftMax[i - 1], arr[i]);
      }

      rightMax[n - 1] = arr[n - 1];
      for (int i = n - 2; i >= 0; i--) {
         rightMax[i] = Math.max(rightMax[i + 1], arr[i]);
      }

      int totalWater = 0;
      for (int i = 0; i < n; i++) {
         totalWater += Math.min(leftMax[i], rightMax[i]) - arr[i];
      }

      return totalWater;
   }

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the number of elements in the array: ");
      int n = scanner.nextInt();

      int[] arr = new int[n];

      System.out.println("Enter the elements of the array: ");
      for (int i = 0; i < n; i++) {
         arr[i] = scanner.nextInt();
      }

      System.out.println("Trapped Water: " + trapWater(arr));
```

```
        scanner.close();


    }
```



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac TrappingRainwater.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java TrappingRainwater
Enter the number of elements in the array: 5
Enter the elements of the array:
3 0 1 0 2
Trapped Water: 5
```

Test Case 1:
Input: 7
3 0 1 0 4 0 2
Expected Output: 10

Hidden Test Case 2:
Input: 6
10 0 2 5 0 7
Expected Output: 12

TIME COMPLEXITY : O(n)

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.

Solution:

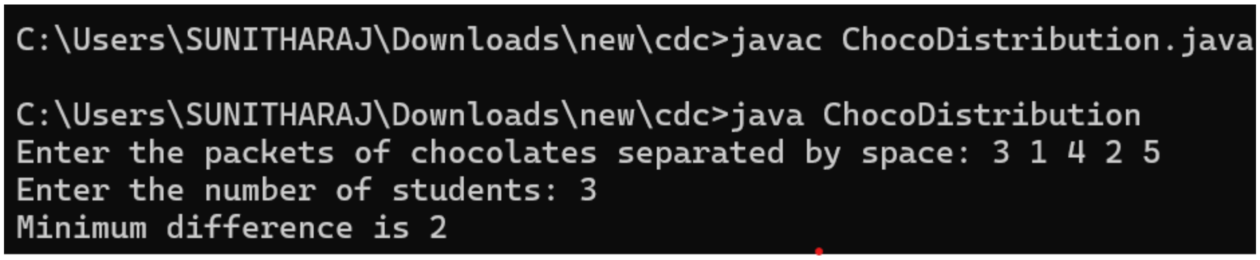
```
import java.util.Arrays;
import java.util.Scanner;

public class ChocoDistribution{
      public static int findminDiff(int[] arr,int n,int m){
          if(m==0 || n==0){
             return 0;


}
          Arrays.sort(arr);
          if(n<m){
             return -1;
          }
          int minDiff = Integer.MAX_VALUE;
          for(int i =0;i+m-1 < n;i++){
             int diff = arr[i+m-1] - arr[i];
             minDiff = Math.min(minDiff,diff);
          }
          return minDiff;
      }
```

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


        System.out.print("Enter the packets of chocolates separated by space: ");
        String[] input = scanner.nextLine().split(" ");
        int[] arr = new int[input.length];
        for (int i = 0; i < input.length; i++) {
            arr[i] = Integer.parseInt(input[i]);
        }


        System.out.print("Enter the number of students: ");
        int m = scanner.nextInt();


        int result = findminDiff(arr, arr.length, m);
        if (result == -1) {
            System.out.println("Not enough packets for each student.");
        } else {
            System.out.println("Minimum difference is " + result);
        }

        scanner.close();
    }
}
```

Test Case 1:
Input: Packets: 12 4 7 9 5 3 8 6
Number of students: 3
Expected Output: 2

Test Case 2:
Input: Packets: 20 25 30 35 40 45
Number of students: 3
Expected Output: 10

OUTPUT:


```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac ChocoDistribution.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java ChocoDistribution
Enter the packets of chocolates separated by space: 3 1 4 2 5
Enter the number of students: 3
Minimum difference is 2
```

TIME COMPLEXITY : O(n logn)

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.

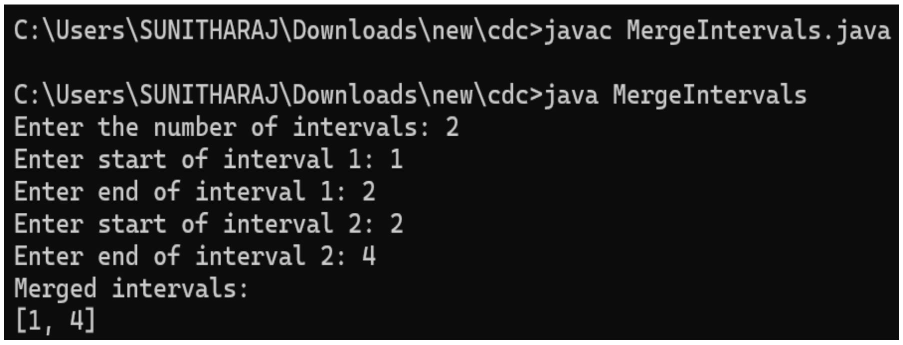
Solution:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class MergeIntervals {

    public static int[][] merge(int[][] intervals) {
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
        List<int[]> merged = new ArrayList<>();

        for (int[] interval : intervals) {
            if (merged.isEmpty() || merged.get(merged.size() - 1)[1] < interval[0]) {
                merged.add(interval);
            } else {
                merged.get(merged.size() - 1)[1] = Math.max(merged.get(merged.size() - 1)[1], interval[1]);
            }
        }

        return merged.toArray(new int[merged.size()][]);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of intervals: ");
        int n = scanner.nextInt();

        int[][] intervals = new int[n][2];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter start of interval " + (i + 1) + ": ");
            intervals[i][0] = scanner.nextInt();
            System.out.print("Enter end of interval " + (i + 1) + ": ");
            intervals[i][1] = scanner.nextInt();
        }

        scanner.close();
        int[][] mergedIntervals = merge(intervals);

        System.out.println("Merged intervals:");
        for (int[] interval : mergedIntervals) {
            System.out.println(Arrays.toString(interval));
        }
    }
}
```

Test Case 1:
Input:
Number of intervals: 4
Interval 1: 1 3
Interval 2: 2 6
Interval 3: 8 10
Interval 4: 15 18
Expected Output:
Merged intervals:
[1, 6]
[8, 10]
[15, 18]

Test Case 2:
Input:
Number of intervals: 3
Interval 1: 1 4
Interval 2: 4 5
Interval 3: 7 9
Expected Output:
Merged intervals:
[1, 5]
[7, 9]

OUTPUT:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac MergeIntervals.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java MergeIntervals
Enter the number of intervals: 2
Enter start of interval 1: 1
Enter end of interval 1: 2
Enter start of interval 2: 2
Enter end of interval 2: 4
Merged intervals:
[1, 4]
```

TIME COMPLEXITY : O(n logn)

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.

Solution:

```
import java.util.Scanner;

public class BooleanMatrix{
```

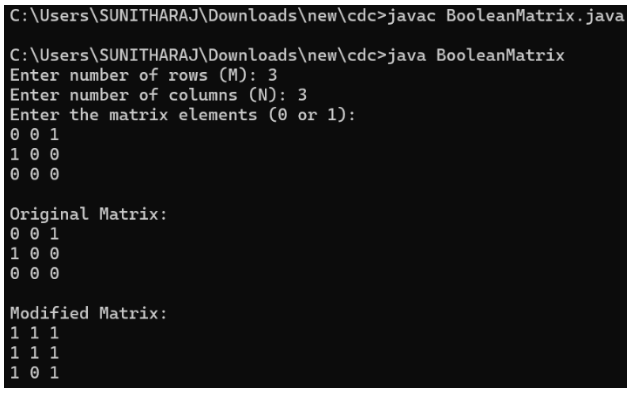
```
public static void modifyMatrix(int[][] mat) {
    int M = mat.length;
    int N = mat[0].length;

    boolean[] rows = new boolean[M];
    boolean[] cols = new boolean[N];

    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            if (mat[i][j] == 1) {
                rows[i] = true;
                cols[j] = true;
            }
        }
    }

    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            if (rows[i] || cols[j]) {
                mat[i][j] = 1;
            }
        }
    }
}

public static void printMatrix(int[][] mat) {
            for (int[] row:mat) {for (int cell : row) {
            System.out.print(cell + " ");
        }
        System.out.println();
    }
}

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

    System.out.print("Enter number of rows (M): ");
    int M = scanner.nextInt();

    System.out.print("Enter number of columns (N): ");
    int N = scanner.nextInt();

    int[][] mat = new int[M][N];

    System.out.println("Enter the matrix elements (0 or 1):");
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            mat[i][j] = scanner.nextInt();
        }
    }
```

```
        System.out.println("\nOriginal Matrix:");
        printMatrix(mat);

        modifyMatrix(mat);

        System.out.println("\nModified Matrix:");
        printMatrix(mat);

        scanner.close();
    }
}
```

Test Case:
Input:
Enter number of rows (M): 3
Enter number of columns (N): 3
Enter the matrix elements (0 or 1):
0 1 0
0 0 0
1 0 0
Output:
Original Matrix:
0 1 0
0 0 0
1 0 0

Modified Matrix:
1 1 1
1 1 1
1 1 1

OUTPUT :



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac BooleanMatrix.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java BooleanMatrix
Enter number of rows (M): 3
Enter number of columns (N): 3
Enter the matrix elements (0 or 1):
0 0 1
1 0 0
0 0 0

Original Matrix:
0 0 1
1 0 0
0 0 0

Modified Matrix:
1 1 1
1 1 1
1 0 1
```

TIME COMPLEXITY : O(M * N)

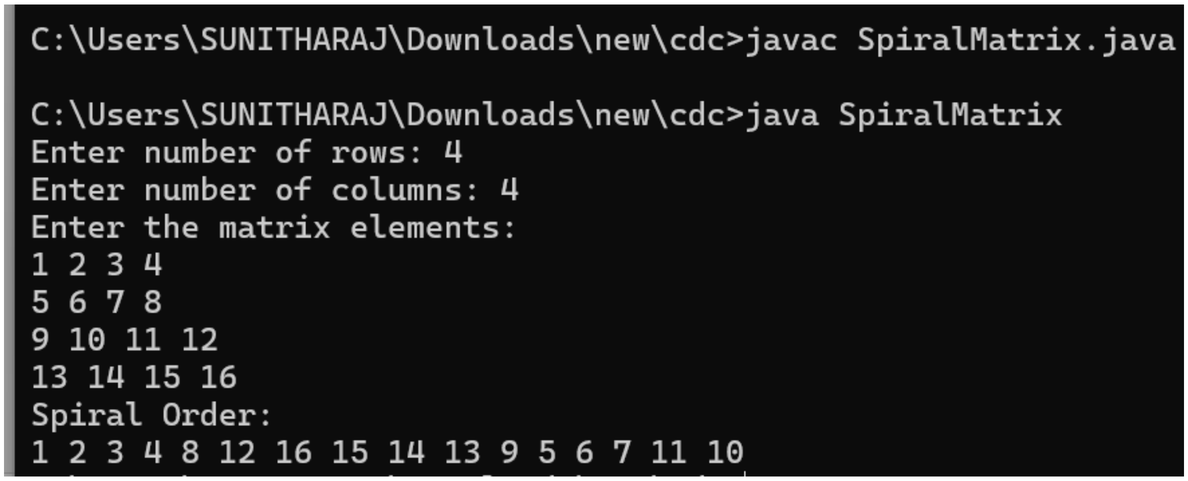
10.Print a given matrix in spiral form Given an m x n matrix, the task is to print all elements of the matrixin spiral form.

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

Solution:

```
import java.util.Scanner;

public class SpiralMatrix {
    public static void printSpiral(int[][] matrix) {
        int m = matrix.length;
        int n = matrix[0].length;
        int top = 0, bottom = m - 1, left = 0, right = n - 1;

        while (top <= bottom && left <= right) {
            for (int i = left; i <= right; i++) {
                System.out.print(matrix[top][i] + " ");
            }
            top++;

            for (int i = top; i <= bottom; i++) {
                System.out.print(matrix[i][right] + " ");
            }
            right--;

            if (top <= bottom) {
                for (int i = right; i >= left; i--) {
                    System.out.print(matrix[bottom][i] + " ");

                }
                bottom--;
            }

            if (left <= right) {
                for (int i = bottom; i >= top; i--) {
                    System.out.print(matrix[i][left] + " ");
                }
                left++;
            }
        }
    }

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

        System.out.print("Enter number of rows: ");
        int m = scanner.nextInt();
        System.out.print("Enter number of columns: ");
        int n = scanner.nextInt();

        int[][] matrix = new int[m][n];
        System.out.println("Enter the matrix elements:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }

        scanner.close();

        System.out.println("Spiral Order:");
        printSpiral(matrix);
    }
}
```

Output:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac SpiralMatrix.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java SpiralMatrix
Enter number of rows: 4
Enter number of columns: 4
Enter the matrix elements:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Spiral Order:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

TIME COMPLEXITY : O(M * N)

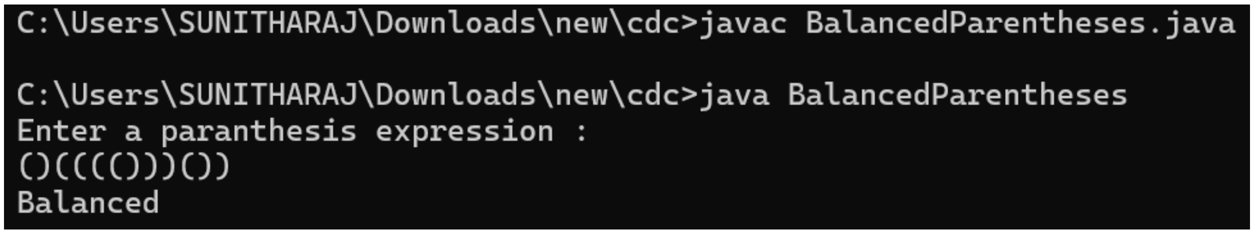
13.Check if given Parentheses expression is balanced or not Given a string str of length N, consisting of ((and) only, the task is to check whether it is balanced or not. Input: str = “((()))()()” Output: Balanced Input: str = “())((())” Output: Not Balanced

Solution:

```
import java.util.Scanner;
import java.util.Stack;
public class BalancedParentheses{
      public static String isBalanced(String expression){
         Stack<Character> stack=new Stack<>();
         for(char ch:expression.toCharArray()){
            if(ch=='('){
                stack.push(ch);
            }
```

```
              else if(ch == ')'){
                   if(stack.isEmpty()){
                        return "Not Balanced";
                   }
                   stack.pop();
              }
           }
           return stack.isEmpty()? "Balanced" : "Not Balanced";
       }
       public static void main(String args[]){
            Scanner scanner = new Scanner(System.in);
            System.out.println("Enter a paranthesis expression : ");
            String expression =scanner.nextLine();
            System.out.println(isBalanced(expression));
       }
  }
```

Output :



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac BalancedParentheses.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java BalancedParentheses
Enter a paranthesis expression :
()((()))())
Balanced
```

Time complexity : O (n)

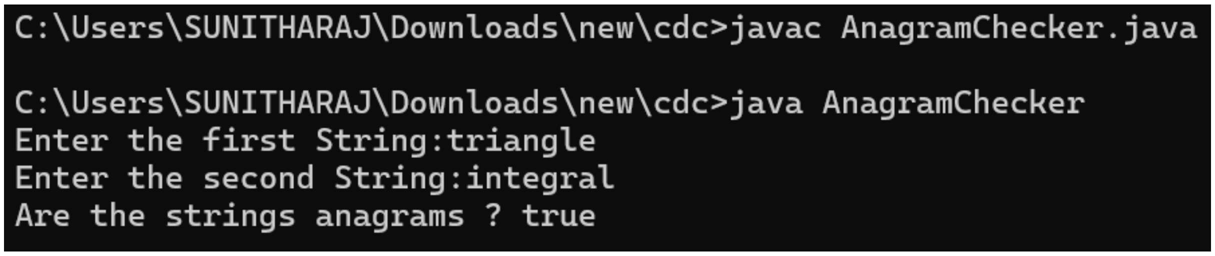
14.Check if two Strings are Anagrams of each other Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters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

Solution:

```
import java.util.Scanner;
import java.util.Arrays;
public class AnagramChecker{
       public static boolean isAnagram(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[]){

              Scanner scanner=new Scanner(System.in);

              System.out.print("Enter the first String:");
              String s1=scanner.nextLine();

              System.out.print("Enter the second String:");
              String s2=scanner.nextLine();

              System.out.println("Are the strings anagrams ? " +isAnagram(s1,s2));
        }
 }
```

Ouput:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac AnagramChecker.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java AnagramChecker
Enter the first String:triangle
Enter the second String:integral
Are the strings anagrams ? true
```

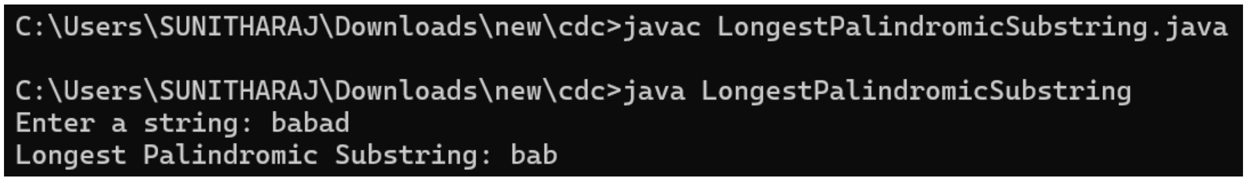
TIME COMPLEXITY : O(n logn)

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.

Solution:

```
import java.util.Scanner;

public class LongestPalindromicSubstring {

    public static String longestPalindrome(String s) {
        if (s == null || s.length() < 1) return "";

        String longest = "";

        for (int i = 0; i < s.length(); i++) {
            String oddPalindrome = expandAroundCenter(s, i, i);
            if (oddPalindrome.length() > longest.length()) {
                longest = oddPalindrome;
            }
            String evenPalindrome = expandAroundCenter(s, i, i + 1);
            if (evenPalindrome.length() > longest.length()) {
                longest = evenPalindrome;
            }
        }
        return longest;
    }

    private static String expandAroundCenter(String s, int left, int right) {
        while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {
            left--;
            right++;
        }
        return s.substring(left + 1, right);
    }

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

        System.out.print("Enter a string: ");
        String str = scanner.nextLine();

        System.out.println("Longest Palindromic Substring: " + longestPalindrome(str));
    }
}
```

Output:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac LongestPalindromicSubstring.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java LongestPalindromicSubstring
Enter a string: babad
Longest Palindromic Substring: bab
```

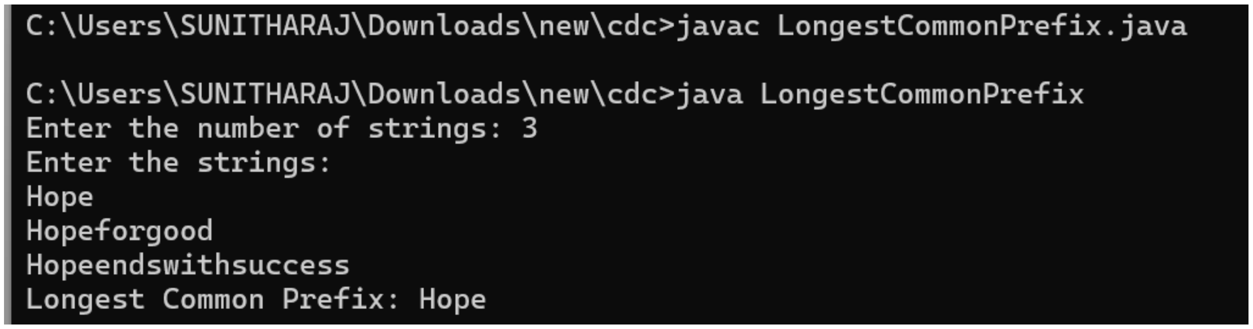
TIME COMPLEXITY : O(n^2)

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".

Solution:

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

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

        if (i == 0) {
            return "-1";
        }

        return first.substring(0, i);
    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of strings: ");
        int n = scanner.nextInt();
         scanner.nextLine();

        String[] arr = new String[n];
        System.out.println("Enter the strings:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextLine();
        }

        System.out.println("Longest Common Prefix: " + longestCommonPrefix(arr));
    }
}
```

Output:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac LongestCommonPrefix.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java LongestCommonPrefix
Enter the number of strings: 3
Enter the strings:
Hope
Hopeforgood
Hopeendswithsuccess
Longest Common Prefix: Hope
```

TIME COMPLEXITY : O(n * log(n) +m)

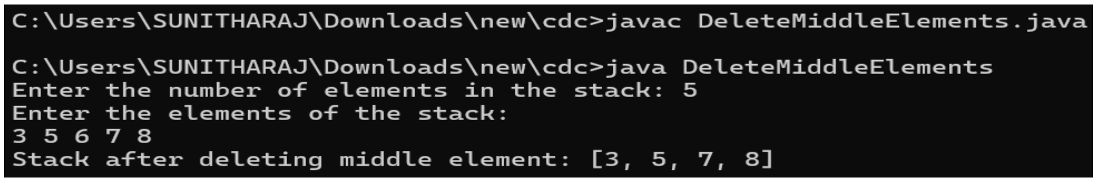
17.Delete middle element of a stack Given a stack with push(), pop(), and empty() operations, The taskis to delete the middle element of it without using any additional data structure.

Solution:

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

public class DeleteMiddleElements {

    public static void deleteMiddle(Stack<Integer> stack, int size, int current) {
        if (stack.isEmpty() || current == size / 2) {
            stack.pop();
            return;
        }

        int temp = stack.pop();
        deleteMiddle(stack, size, current + 1);
        stack.push(temp);
    }
```

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

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

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

        int size = stack.size();
        deleteMiddle(stack, size, 0);

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

OUTPUT:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac DeleteMiddleElements.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java DeleteMiddleElements
Enter the number of elements in the stack: 5
Enter the elements of the stack:
3 5 6 7 8
Stack after deleting middle element: [3, 5, 7, 8]
```

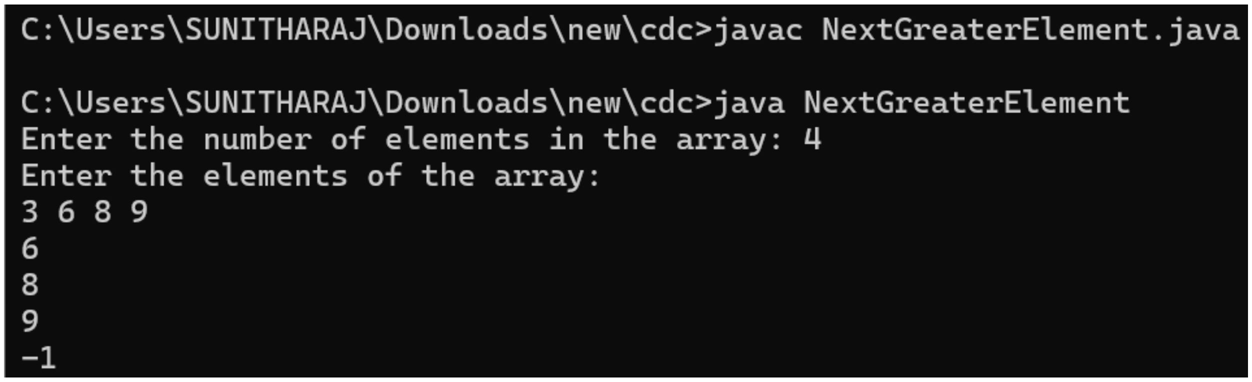
Time complexity : O(n)

18.Next Greater Element (NGE) for every element in given Array Given an array, print the Next Greater Element (NGE) for every element.

Solution:

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

public class NextGreaterElement {

    public static void nextGreaterElement(int[] arr) {
        Stack<Integer> stack = new Stack<>();
        int n = arr.length;
        int[] result = new int[n];

        for (int i = 0; i < n; i++) {
            result[i] = -1; // Initialize all elements as -1
            while (!stack.isEmpty() && arr[stack.peek()] < arr[i]) {
            result[stack.pop()] = arr[i];
            }

            stack.push(i);
        }


        for (int i = 0; i < n; i++) {
            System.out.println(result[i]);
        }
    }

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

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        nextGreaterElement(arr);
    }
}
```

Output:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac NextGreaterElement.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java NextGreaterElement
Enter the number of elements in the array: 4
Enter the elements of the array:
3 6 8 9
6
8
9
-1
```

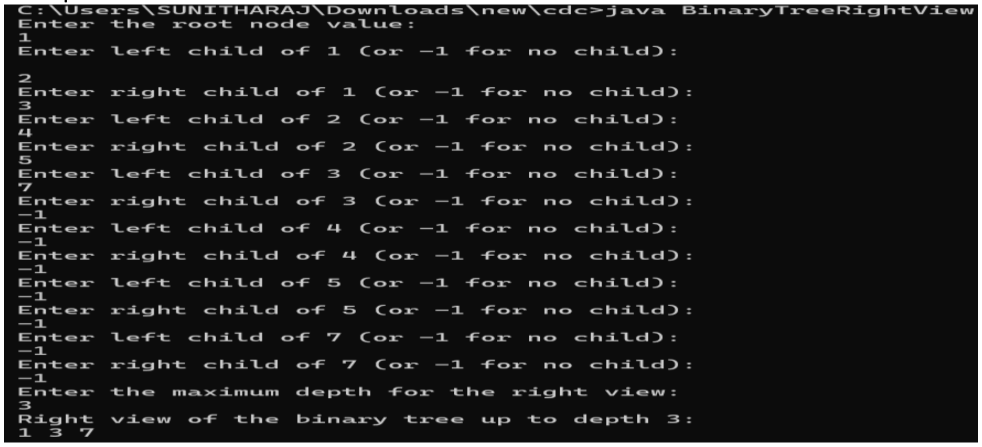
TIME COMPLEXITY : O(n)

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.

Solution:

```
import java.util.*;

class TreeNode {
    int val;
    TreeNode left, right;

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

public class BinaryTreeRightView {

    public void printRightView(TreeNode root, int maxDepth) {
        if (root == null) return;

        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);
        int currentDepth = 0;

        while (!queue.isEmpty() && currentDepth < maxDepth) {
            int levelSize = queue.size();
            currentDepth++;

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

                if (i == levelSize - 1) {
                    System.out.print(currentNode.val + " ");
                }

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

    public TreeNode createTreeFromInput() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the root node value:");
        int rootVal = scanner.nextInt();

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

        while (!queue.isEmpty()) {
            TreeNode currentNode = queue.poll();

            System.out.println("Enter left child of " + currentNode.val + " (or -1 for no child):");
            int leftVal = scanner.nextInt();
            if (leftVal != -1) {
                currentNode.left = new TreeNode(leftVal);
                queue.add(currentNode.left);
            }

            System.out.println("Enter right child of " + currentNode.val + " (or -1 for no child):");
            int rightVal = scanner.nextInt();
            if (rightVal != -1) {
                currentNode.right = new TreeNode(rightVal);
                queue.add(currentNode.right);
            }
        }
        return root;
    }

    public static void main(String[] args) {
        BinaryTreeRightView tree = new BinaryTreeRightView();
        TreeNode root = tree.createTreeFromInput();

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the maximum depth for the right view:");
        int maxDepth = scanner.nextInt();

        System.out.println("Right view of the binary tree up to depth " + maxDepth + ":");
        tree.printRightView(root, maxDepth);
    }
}
```

Output:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>java BinaryTreeRightView
Enter the root node value:
1
Enter left child of 1 (or -1 for no child):

2
Enter right child of 1 (or -1 for no child):
3
Enter left child of 2 (or -1 for no child):
4
Enter right child of 2 (or -1 for no child):
5
Enter left child of 3 (or -1 for no child):
7
Enter right child of 3 (or -1 for no child):
-1
Enter left child of 4 (or -1 for no child):
-1
Enter right child of 4 (or -1 for no child):
-1
Enter left child of 5 (or -1 for no child):
-1
Enter right child of 5 (or -1 for no child):
-1
Enter left child of 7 (or -1 for no child):
-1
Enter right child of 7 (or -1 for no child):
-1
Enter the maximum depth for the right view:
3
Right view of the binary tree up to depth 3:
1 3 7
```

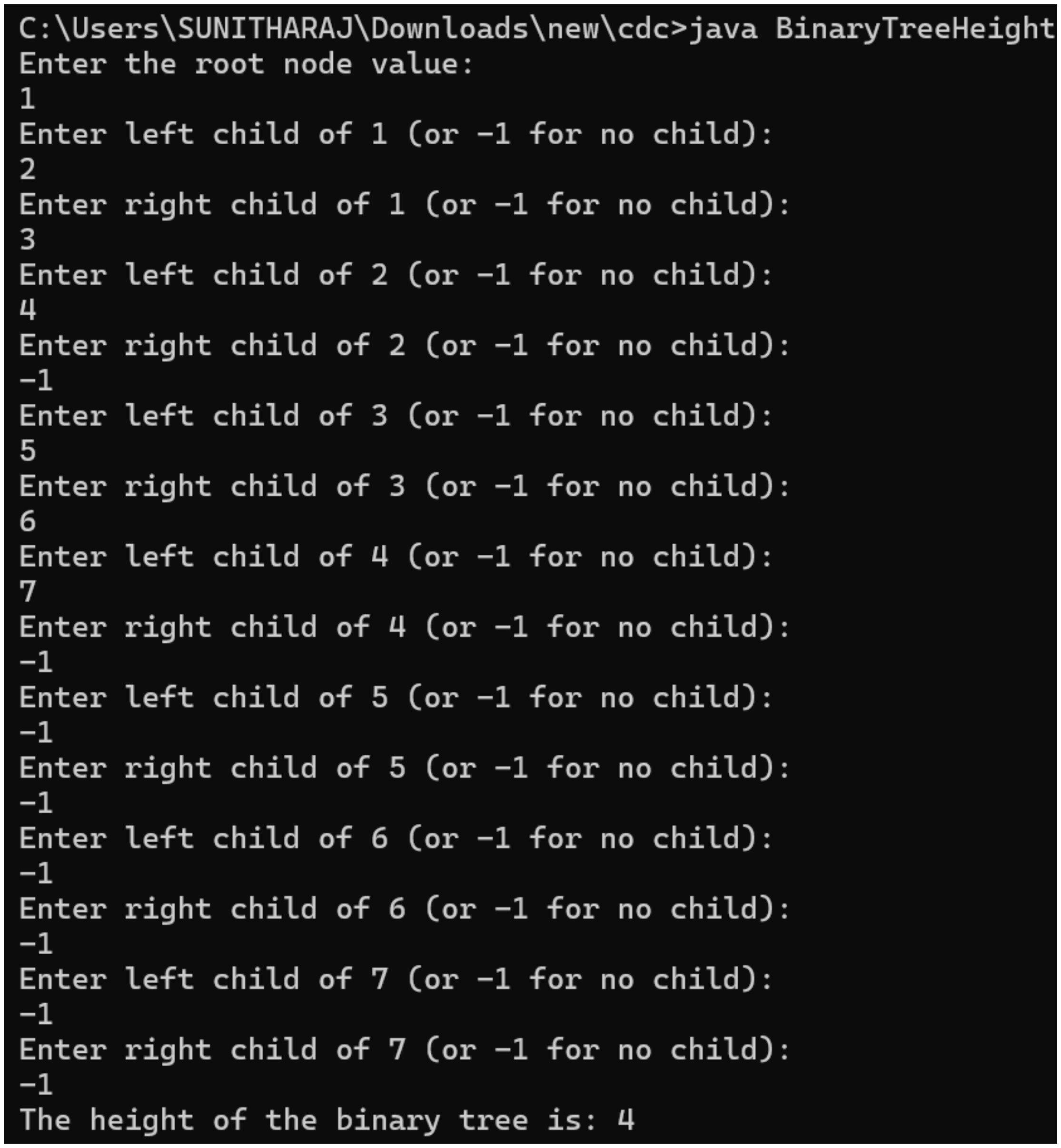
TIME COMPLEXITY : O(n)

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.

Solution:

```
import java.util.*;

class TreeNode {
    int val;
    TreeNode left, right;

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

public class BinaryTreeHeight {

    public int findHeight(TreeNode root) {
        if (root == null) return 0;
        int leftHeight = findHeight(root.left);
        int rightHeight = findHeight(root.right);
        return Math.max(leftHeight, rightHeight) + 1;
    }

    public TreeNode createTreeFromInput() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the root node value:");
        int rootVal = scanner.nextInt();

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

        while (!queue.isEmpty()) {
            TreeNode currentNode = queue.poll();

            System.out.println("Enter left child of " + currentNode.val + " (or -1 for no child):");
            int leftVal = scanner.nextInt();
            if (leftVal != -1) {
                currentNode.left = new TreeNode(leftVal);
                queue.add(currentNode.left);
            }

            System.out.println("Enter right child of " + currentNode.val + " (or -1 for no child):");
            int rightVal = scanner.nextInt();
            if (rightVal != -1) {
                currentNode.right = new TreeNode(rightVal);
                queue.add(currentNode.right);
            }
        }
        return root;
    }

    public static void main(String[] args) {
        BinaryTreeHeight tree = new BinaryTreeHeight();
        TreeNode root = tree.createTreeFromInput();
        int height = tree.findHeight(root);
        System.out.println("The height of the binary tree is: " + height);
    }
}
```

Output:



```
C:\Users\SUNITHARAJ\Downloads\new\cdc>java BinaryTreeHeight
Enter the root node value:
1
Enter left child of 1 (or -1 for no child):
2
Enter right child of 1 (or -1 for no child):
3
Enter left child of 2 (or -1 for no child):
4
Enter right child of 2 (or -1 for no child):
-1
Enter left child of 3 (or -1 for no child):
5
Enter right child of 3 (or -1 for no child):
6
Enter left child of 4 (or -1 for no child):
7
Enter right child of 4 (or -1 for no child):
-1
Enter left child of 5 (or -1 for no child):
-1
Enter right child of 5 (or -1 for no child):
-1
Enter left child of 6 (or -1 for no child):
-1
Enter right child of 6 (or -1 for no child):
-1
Enter left child of 7 (or -1 for no child):
-1
Enter right child of 7 (or -1 for no child):
-1
The height of the binary tree is: 4
```

TIME COMPLEXITY : O(n)