Coding practice Problems and Solution:

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.

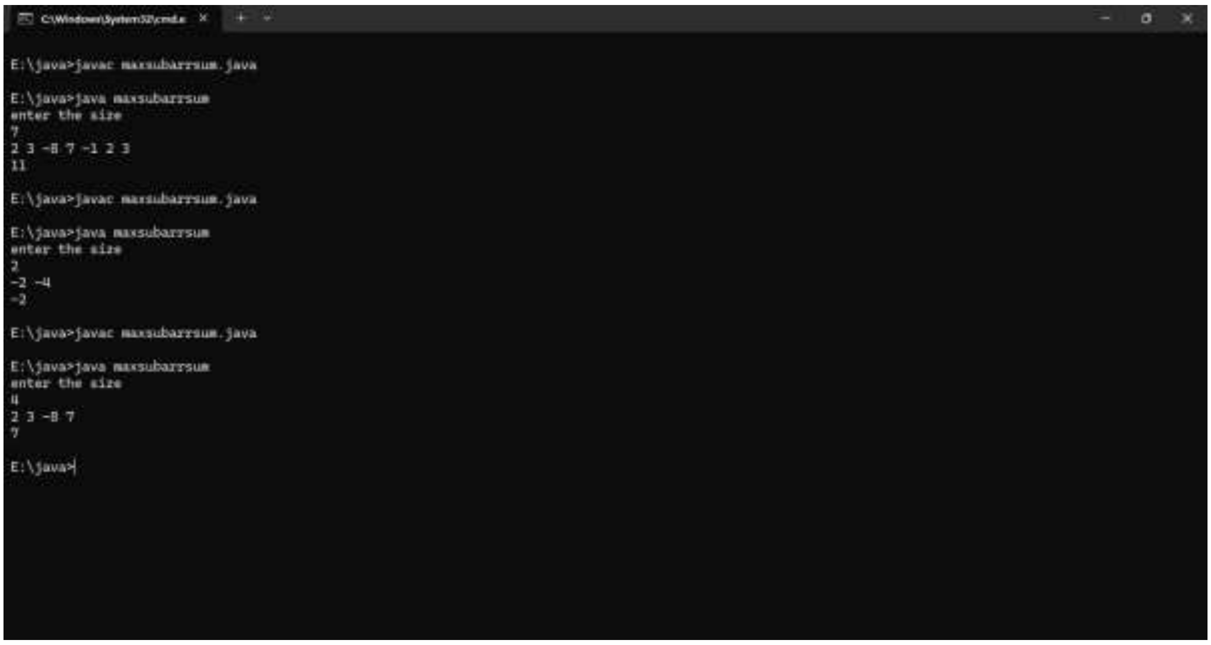
Input: arr[] = {-2, -4} Output: –2 Explanation: The subarray {-2} has the largest sum -2.

Input: arr[] = {5, 4, 1, 7, 8} Output: 25

```
import java.util.*;

public class maxsubarrsum {

    public static void main(String[] arr) {
        System.out.println("enter the size");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int maxi = Integer.MIN_VALUE;
        int sum = 0;
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
            nums[i] = sc.nextInt();
        }
        for (int i = 0; i < nums.length; i++) {
            sum += nums[i];
            if (sum > maxi) {
                maxi = sum;
            }
            if (sum < 0) {
                sum = 0;
            }
        }
        System.out.println(maxi);
    }

}
```

Output:

Tc=o(n)
Ts=o(1)



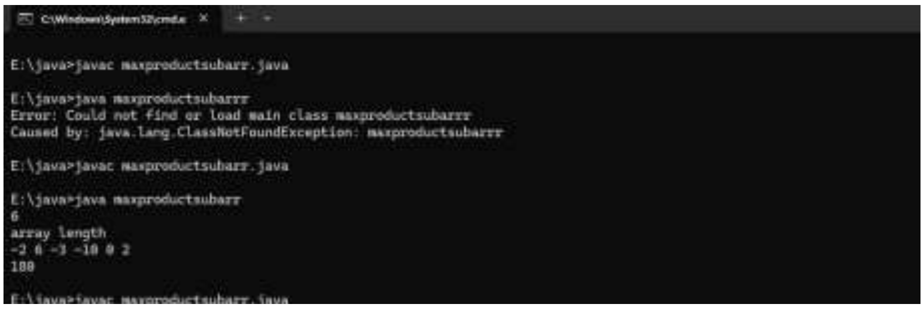
```
C:\Windows\System32\cmd.e

E:\java>javac maxsubarrsum.java

E:\java>java maxsubarrsum
enter the size
7
2 3 -8 7 -1 2 3
11

E:\java>javac maxsubarrsum.java

E:\java>java maxsubarrsum
enter the size
2
-2 -4
-2

E:\java>javac maxsubarrsum.java

E:\java>java maxsubarrsum
enter the size
4
2 3 -8 7
7

E:\java>
```

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 :

```
import java.util.*;
public class maxproductsubarr{
    public static void main(String[] hello){
        Scanner sc = new Scanner(System.in);
        int n= sc.nextInt();
        System.out.println("array length");
        int nums[] = new int[n];
        for(int i=0;i<n;i++){
            nums[i]= sc.nextInt();
        }
        int res=nums[0];
        int minproduct = nums[0];
        int maxproduct = nums[0];

        for(int i=1;i<nums.length;i++){
            if(nums[i] < 0){
                int temp = maxproduct;
                maxproduct = minproduct;
                minproduct=temp;
            }
            maxproduct = Math.max(nums[i],maxproduct*nums[i]);
            minproduct = Math.min(nums[i], minproduct*nums[i]);
            res = Math.max(res,maxproduct);
        }
        System.out.println(res);

    }

}
```

Output:

TC=o(n)



```
E:\java>javac maxproductsubarr.java

E:\java>java maxproductsubarrr
Error: Could not find or load main class maxproductsubarrr
Caused by: java.lang.ClassNotFoundException: maxproductsubarrr

E:\java>javac maxproductsubarr.java

E:\java>java maxproductsubarr
6
array length
-2 6 -3 -10 0 2
180

E:\java>javac maxproductsubarr.java
```

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.

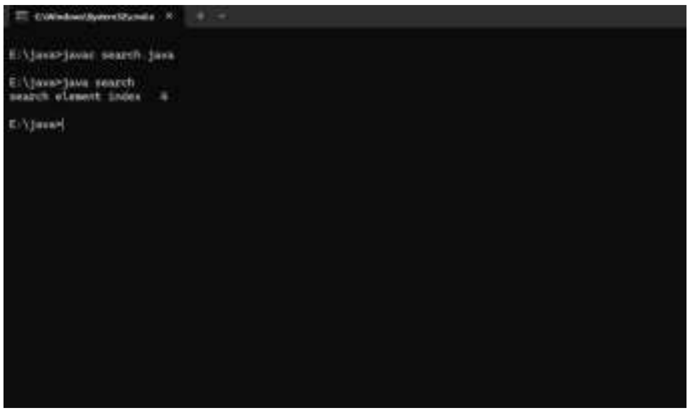
Input : arr[] = {4, 5, 6, 7, 0, 1, 2}, key = 0 Output : 4 Input : arr[] = { 4, 5, 6, 7, 0, 1, 2 }, key = 3 Output : -1
Input : arr[] = {50, 10, 20, 30, 40}, key = 10 Output : 1

Code:

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online
public class search{
    private int searchdata(int[] num, int k){
        int n= num.length;
        int low=0;
        int high =n-1;
        while(low <= high){
            int mid = (low+high)/2;
            if(num[mid] == k){
                return mid;
            }
            else if(num[low]<= num[mid]){
                if(num[low] <= k && k < num[mid]){
                    high = mid-1;
                }else{
                    low=mid+1;
                }
            }else{
                if(num[mid] < k && k <= num[high]){
                    low= mid+1;
                }else{
                    high = mid-1;
                }
            }


        }
          return -1;
    }
    public static void main(String[] str){
        search  s = new search();
        int nums[] = {4, 5, 6, 7, 0, 1, 2};
        int k=0;
        int res = s.searchdata(nums,k);
        if(res != -1){
            System.out.println("search element index   "+  res);
        }else{
            System.out.println(-1);
        }
    }
}
```

Output:

TC: O(logn)



```
C:\Windows\System32\cmd.e

E:\java>javac search.java

E:\java>java search
search element index   4

E:\java>
```

4) Container with Most Water

 Input: arr = [1, 5, 4, 3] Output: 6 Explanation: 5 and 3 are distance 2 apart. So the size of the base = 2. Height of container = min(5, 3) = 3. So total area = 3 * 2 = 6

Input: arr = [3, 1, 2, 4, 5] Output: 12 Explanation: 5 and 3 are distance 4 apart. So the size of the base = 4. Height of container = min(5, 3) = 3. So total area = 4 * 3 = 12

Code:

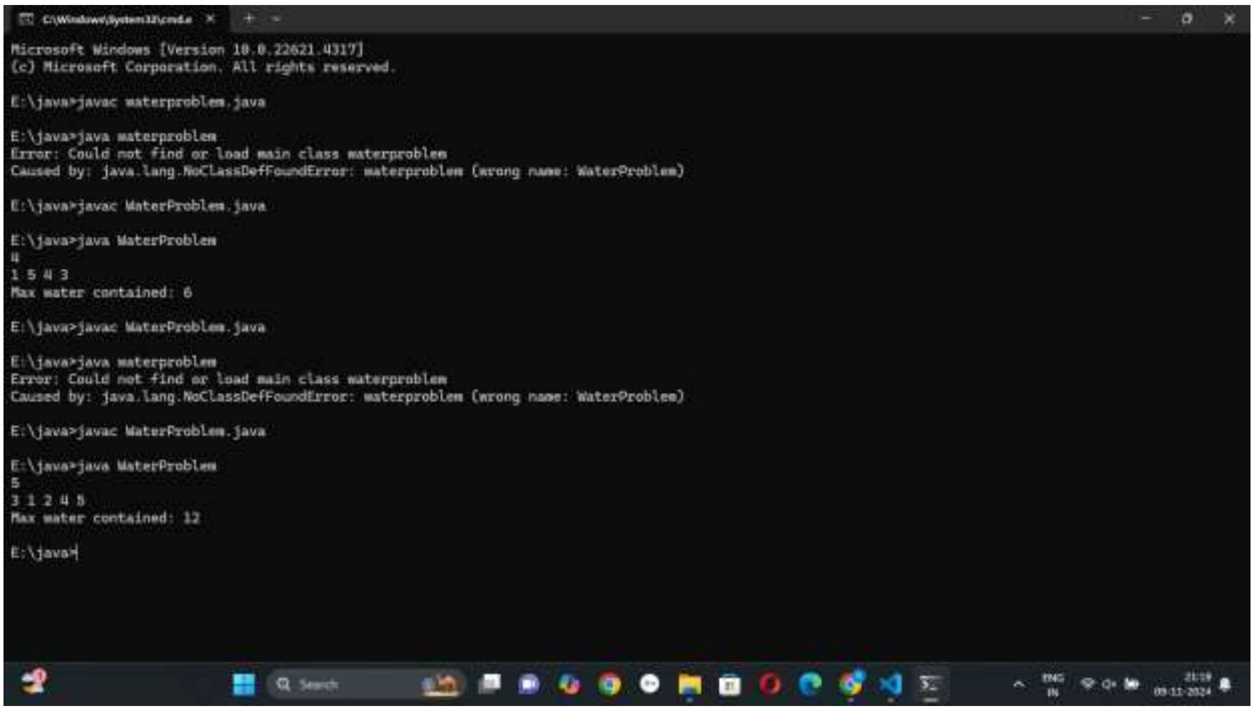
```
import java.util.*;


public class WaterProblem {
```

```
public static int maxwater(int[] height) {
    int maxarea = 0;
    int start = 0;
    int end = height.length - 1;

    while (start < end) {
        int curr = Math.min(height[start], height[end]);
        maxarea = Math.max(maxarea, curr * (end - start));
        if (height[start] < height[end]) {
            start++;
        } else {
            end--;
        }
    }
    return maxarea;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n=sc.nextInt();
    int num[] =new int[n];
    for(int i=0;i<n;i++){
        num[i]=sc.nextInt();
    }

    System.out.println("Max water contained: " + maxwater(num));
}
```

}

Output:

Tc: O(n)



```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

E:\java>javac waterproblem.java

E:\java>java waterproblem
Error: Could not find or load main class waterproblem
Caused by: java.lang.NoClassDefFoundError: waterproblem (wrong name: WaterProblem)

E:\java>javac WaterProblem.java

E:\java>java WaterProblem
4
1 5 4 3
Max water contained: 6

E:\java>javac WaterProblem.java

E:\java>java waterproblem
Error: Could not find or load main class waterproblem
Caused by: java.lang.NoClassDefFoundError: waterproblem (wrong name: WaterProblem)

E:\java>javac WaterProblem.java

E:\java>java WaterProblem
5
3 1 2 4 5
Max water contained: 12

E:\java>
```

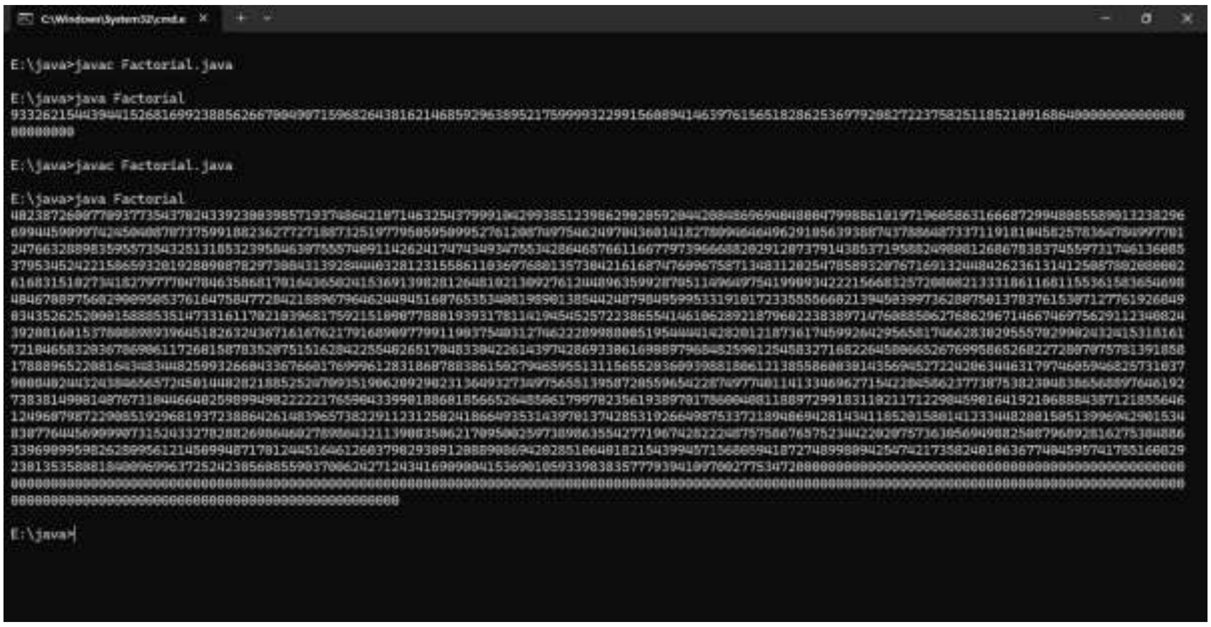
5) Find the Factorial of a large number

Input: 100 Output:
93326215443944152681699238856266700490715968264381621468592
963895217599993229915608941463976156518286253697920827223758251185210916864000
000000000000000000 00

Input: 50 Output:
30414093201713378043612608166064768844377641568960512000000
000000

Code:
```
import java.math.BigInteger;
class Factorial{
    public static void main(String[] args) {
        int n=100;
        BigInteger fact = new BigInteger("1");
        for(int i=1;i<n;i++){
            fact =fact.multiply(BigInteger.valueOf(i));
        }
        System.out.println(fact);
    }
}
```

Output
TC: O(n2·logn·loglogn)



```
E:\java>javac Factorial.java

E:\java>java Factorial
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000
00000000

E:\java>javac Factorial.java

E:\java>java Factorial
4023872600770937735437024339230039857193748642107146325437999104299385123986290205920442084869694048004799886101971960586316668729948085589013238296
6994459099742450408707375991882362772718873251977950595099527612087497546249704360141827809464649629105639388743788648733711918104582578364784997701
2476632889835955735432513185323958463075557409114262417474349347553428646576611667797396668820291207379143853719588249808126867838374559731746136085
3795345242215865932019280908782973084313928444032812315586110369768013573042161687476096758713483120254785893207671691324484262361314125087802080002
6168315102734182797770478463586817016436502415369139828126481021309276124489635992870511496497541990934222156683257208082133318611681155361583654698
4046708975602900950537616475847728421889679646244945160765353408198901385442487984959953319101723355556602139450399736280750137837615307127761926849
0343526252000158885351473316117021039681759215109077880193931781141945452572238655414610628921879602238389714760885062768629671466746975629112340824
3920816015378088989396451826324367161676217916890977991190375403127462228998800519544441428201218736174599264295658174662830295557029902432415318161
7210465832036786906117260158783520751516284225540265170483304226143974286933061690897968482590125458327168226458066526769958652682272807075781391858
1788896522081643483448259932660433676601769996128318607883861502794659551311565520360939881806121385586003014356945272242063446317974605946825731037
900084024432438465657245014402821885252470935190620929023136493273497565513958720559654228749774011413346962715422845862377387538230483865688976461 92
7383814900140767310446640259899490222221765904339901886018566526485061799702356193897017860040811889729918311021171229845901641921068884387121855646
1249607987229085192968193723886426148396573822911231250241866493531439701374285319266498753372189406942814341185201580141233448280150513996942901534
8307764456909907315243327828826986460278986432113908350621709500259738986355427719674282224875758676575234422020757363056949882508796892816275384886
3396909959826280956121450994871701244516461260379029309120889086942028510640182154399457156805941872748998094254742173582401063677404595741785160829
2301353580818400969963725242305608559037006242712434169090041536901059339838357779394109700277534720000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000

E:\java>
```

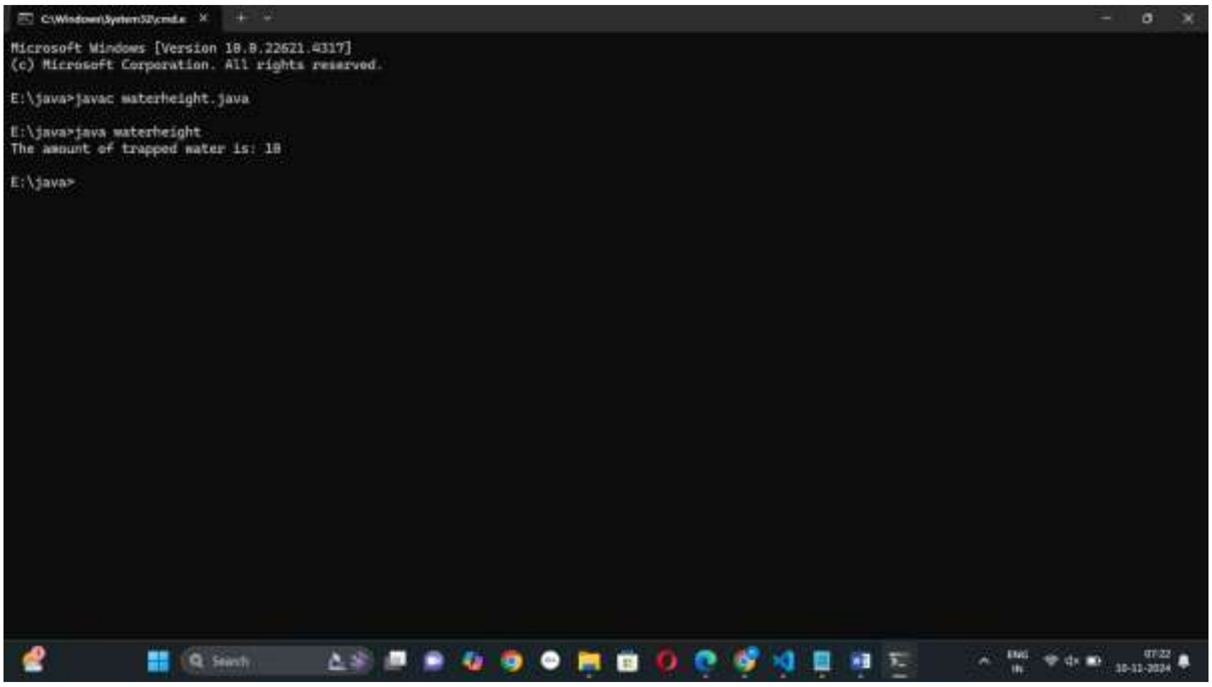
6) Trapping Rainwater Problem states that given an array of n non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.

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

```
class waterheight{
    public int trap(int[] height) {
        int left = 0;
        int right = height.length - 1;
        int leftMax = height[left];
        int rightMax = height[right];
        int water = 0;

        while (left < right) {
            if (leftMax < rightMax) {
                left++;
                leftMax = Math.max(leftMax, height[left]);
                water += leftMax - height[left];
            } else {
                right--;
                rightMax = Math.max(rightMax, height[right]);
                water += rightMax - height[right];
            }
        }

        return water;
    }

    public static void main(String[] args) {
        waterheight solution = new waterheight();

        int[] height = {3, 0, 1, 0, 4, 0, 2};

        int result = solution.trap(height);

        System.out.println("The amount of trapped water is: " + result);
    }
}
```

OutPut:

Time complexity:O(n)



```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

E:\java>javac waterheight.java

E:\java>java waterheight
The amount of trapped water is: 10

E:\java>
```

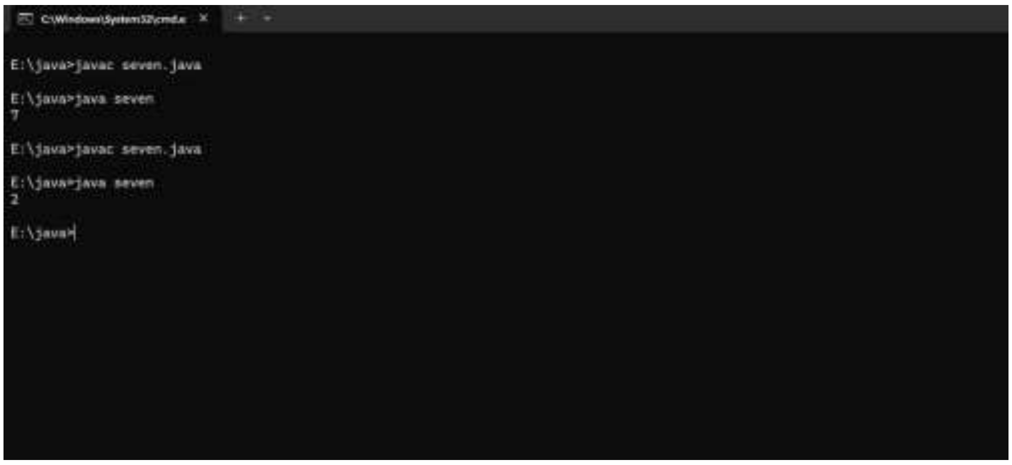
7) Chocolate Distribution Problem Given an array arr[] of n integers where arr[i] represents the number of chocolates in ith packet. Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that: Each student gets exactly one packet.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

Code:

```
import java.util.*;

public class seven {
    public static void main(String[] args) {
        int[] nums = {7, 3, 2, 4, 9, 12, 56};
        int m = 5;
        Arrays.sort(nums);
        int min = Integer.MAX_VALUE;
        int n = nums.length;
        int j = m-1; int i = 0;
        while(j<n) {
            int diff = nums[j] - nums[i];
            min = Math.min(diff,min);
            i++; j++;
        }
        System.out.println(min);
    }
}
```

Output:

TC: O(nlogn)



```
E:\java>javac seven.java

E:\java>java seven
7

E:\java>javac seven.java

E:\java>java seven
2

E:\java>
```

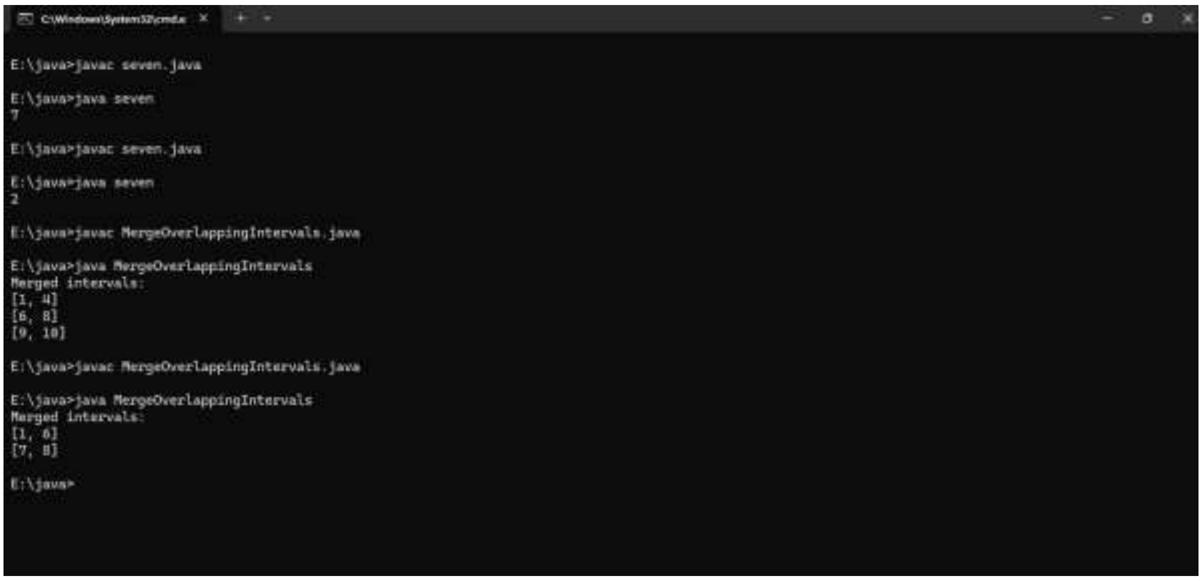
8) . Merge Overlapping Intervals Given an array of time intervals where arr[i] = [starti, endi], the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals.
Input: arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]] Output: [[1, 4], [6, 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:

```
import java.util.*;

public class MergeOverlappingIntervals {
    public int[][] merge(int[][] num) {
        if (num.length == 0) {
            return new int[0][];
        }



        Arrays.sort(num, new Comparator<int[]>() {
            public int compare(int[] a, int[] b) {
                return a[0] - b[0];
            }
        });

        List<int[]> ans = new ArrayList<>();
        int[] currin = num[0];
        ans.add(currin);

        for (int[] interval : num) {
            int currend = currin[1];
            if (interval[0] <= currend) {
                currin[1] = Math.max(currend, interval[1]);
            } else {
                currin = interval;
                ans.add(currin);
            }
        }
        return ans.toArray(new int[ans.size()][]);
    }

    public static void main(String[] args) {
        int[][] num = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};


        MergeOverlappingIntervals mergeIntervals = new
MergeOverlappingIntervals();
        int[][] merged = mergeIntervals.merge(num);

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

Output:
TC: O(nlogn)



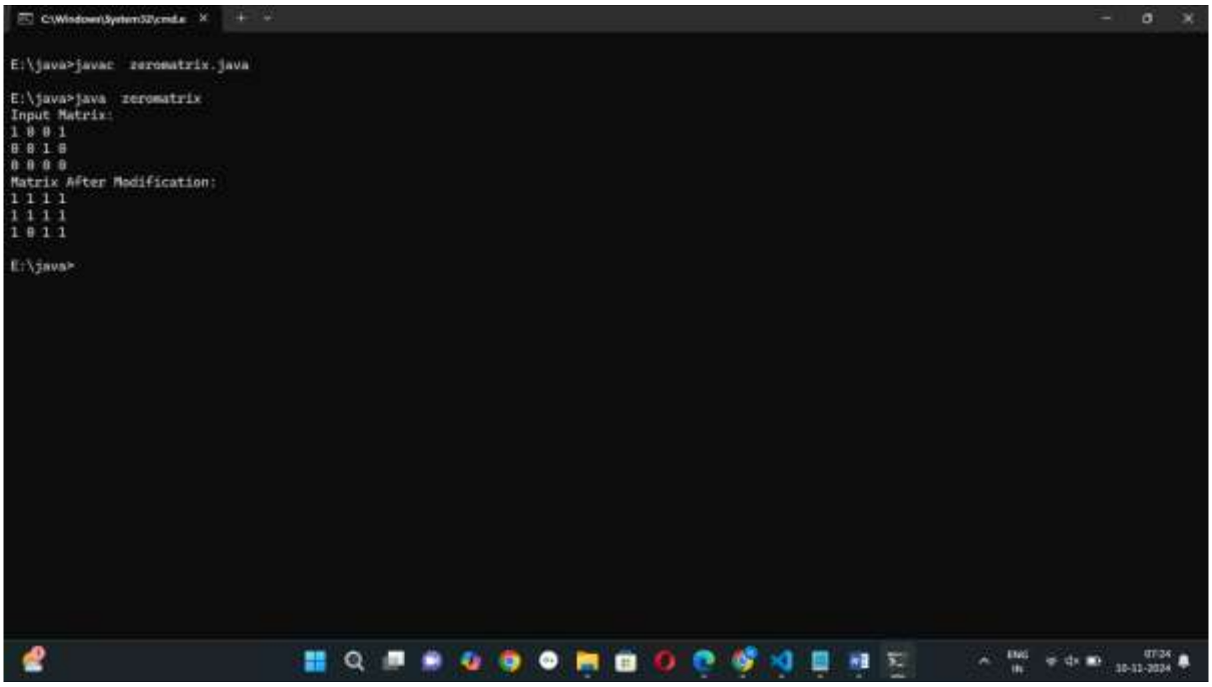
```
E:\java>javac seven.java

E:\java>java seven
7

E:\java>javac seven.java

E:\java>java seven
2

E:\java>javac MergeOverlappingIntervals.java

E:\java>java MergeOverlappingIntervals
Merged intervals:
[1, 4]
[6, 8]
[9, 10]

E:\java>javac MergeOverlappingIntervals.java

E:\java>java MergeOverlappingIntervals
Merged intervals:
[1, 6]
[7, 8]

E:\java>
```

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:

```
import java.util.*;
class zeromatrix{
    public static void modifyMatrix(int mat[][]) {
        boolean row_flag = false;
        boolean col_flag = false;

        for (int i = 0; i < mat.length; i++) {
            for (int j = 0; j < mat[0].length; j++) {
                if (i == 0 && mat[i][j] == 1) row_flag = true;
                if (j == 0 && mat[i][j] == 1) col_flag = true;
                if (mat[i][j] == 1) {
                    mat[0][j] = 1;
                    mat[i][0] = 1;
                }
            }
        }

        for (int i = 1; i < mat.length; i++)
            for (int j = 1; j < mat[0].length; j++)
                if (mat[0][j] == 1 || mat[i][0] == 1)
                    mat[i][j] = 1;

        if (row_flag)
            for (int i = 0; i < mat[0].length; i++)
                mat[0][i] = 1;

        if (col_flag)
            for (int i = 0; i < mat.length; i++)
                mat[i][0] = 1;
    }

    public static void printMatrix(int mat[][]) {
        for (int i = 0; i < mat.length; i++) {
            for (int j = 0; j < mat[0].length; j++)
                System.out.print(mat[i][j] + " ");
            System.out.println();
        }
    }
    public static void main(String args[]) {
        int mat[][] = { { 1, 0, 0, 1 }, { 0, 0, 1, 0 }, { 0, 0, 0, 0 } };
        System.out.println("Input Matrix:");
        printMatrix(mat);
        modifyMatrix(mat);
        System.out.println("Matrix After Modification:");
        printMatrix(mat);
    }
}
```

Output:
Time complexity: O(n)



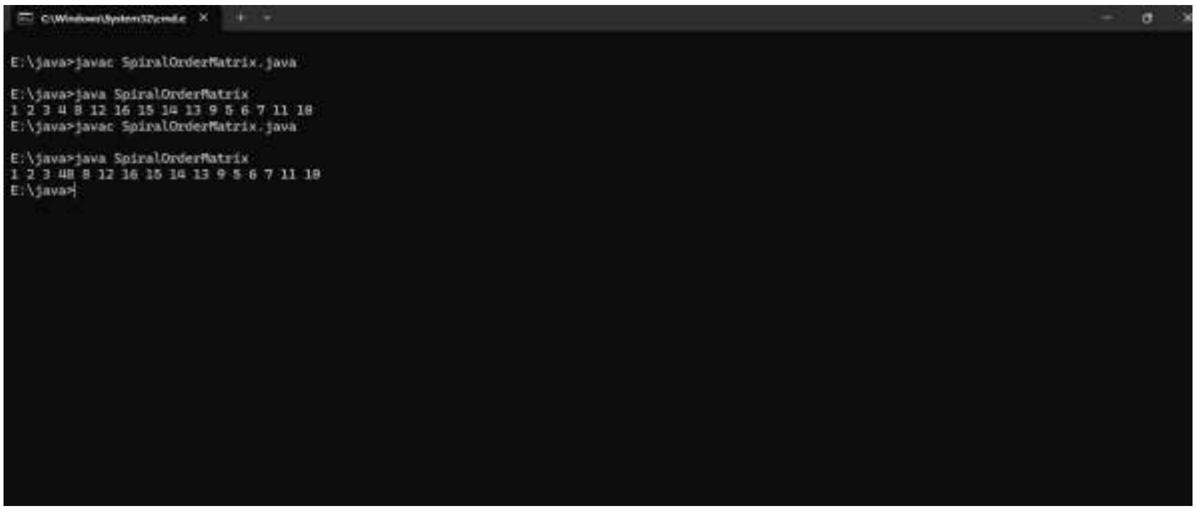
```
C:\Windows\System32\cmd.e

E:\java>javac  zeromatrix.java

E:\java>java  zeromatrix
Input Matrix:
1 0 0 1
0 0 1 0
0 0 0 0
Matrix After Modification:
1 1 1 1
1 1 1 1
1 0 1 1

E:\java>
```

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:

```
import java.util.*;

public class SpiralOrderMatrix {

    public static List<Integer> spiralOrder(int[][] matrix) {
        List<Integer> result = new ArrayList<>();
        if (matrix == null || matrix.length == 0) {
            return result;
        }

        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++) {
                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++;
            }
        }

        return result;
    }

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

        List<Integer> result = spiralOrder(matrix);

        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}
```

Output:
TC: O(m×n)



```
E:\java>javac SpiralOrderMatrix.java

E:\java>java SpiralOrderMatrix
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
E:\java>javac SpiralOrderMatrix.java

E:\java>java SpiralOrderMatrix
1 2 3 48 8 12 16 15 14 13 9 5 6 7 11 10
E:\java>
```

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

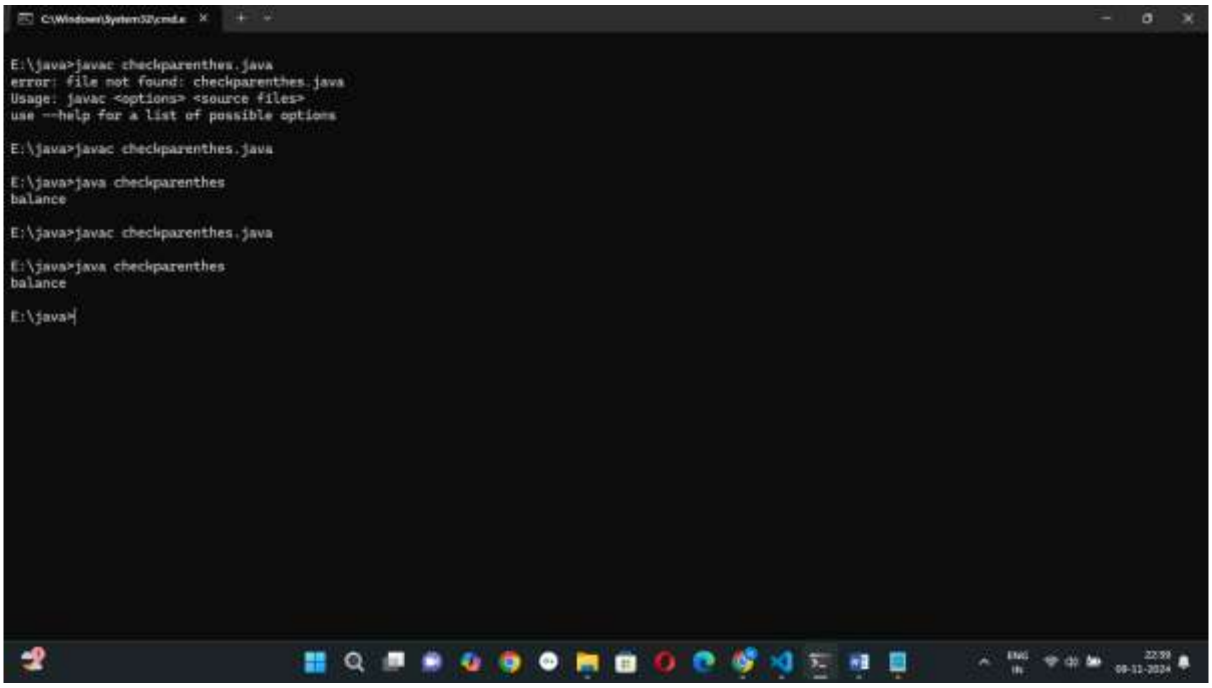
Input: str = “())((())” Output: Not Balanced

Code:

```
import java.util.*;
class checkparenthes{
    boolean check(String s){
        Stack<Character> st = new Stack<>();
        for(char ch : s.toCharArray()){
            if(ch == '('){
                st.push(ch);
            }else if (ch == ')') {
                if (st.isEmpty() || st.pop() != '(') {
                    return false;
                }
            }
        }
        return st.isEmpty();
    }
   public static void main(String[] arg){
      String s = "()";
      checkparenthes ch = new checkparenthes();
      if(ch.check(s)){
         System.out.println("balance");
        }else{
            System.out.println("no");
        }
    }
}
```

Output:

TC:O(n)



```
E:\java>javac checkparenthes.java
error: file not found: checkparenthes.java
Usage: javac <options> <source files>
use --help for a list of possible options

E:\java>javac checkparenthes.java

E:\java>java checkparenthes
balance

E:\java>javac checkparenthes.java

E:\java>java checkparenthes
balance

E:\java>
```

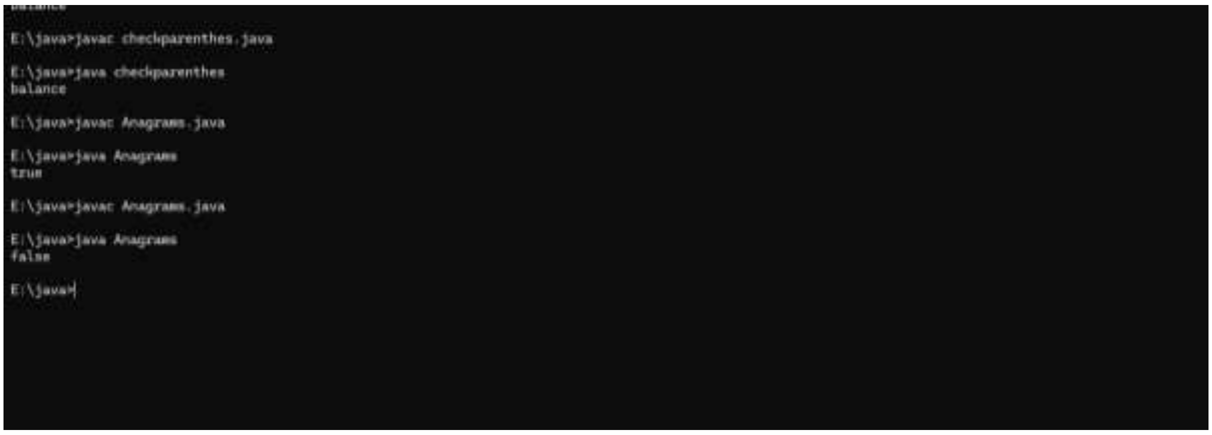
14. Check if two Strings are Anagrams of each other Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters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

```
class Anagrams {

    static final int MAX_CHAR = 26;

    static boolean areAnagrams(String s1, String s2) {
        if (s1.length() != s2.length()) {
            return false;
        }

        int[] freq = new int[MAX_CHAR];

        for (int i = 0; i < s1.length(); i++) {
            freq[s1.charAt(i) - 'a']++;
            freq[s2.charAt(i) - 'a']--;
        }

        for (int count : freq) {
            if (count != 0) {
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) {
        String s1 = "geek";
        String s2 = "kseeg";
        System.out.println(areAnagrams(s1, s2));
    }
}
```

Output:
Time Complexity: O(N)



```
E:\java>javac checkparenthes.java

E:\java>java checkparenthes
balance

E:\java>javac Anagrams.java

E:\java>java Anagrams
true

E:\java>javac Anagrams.java

E:\java>java Anagrams
false

E:\java>
```

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.

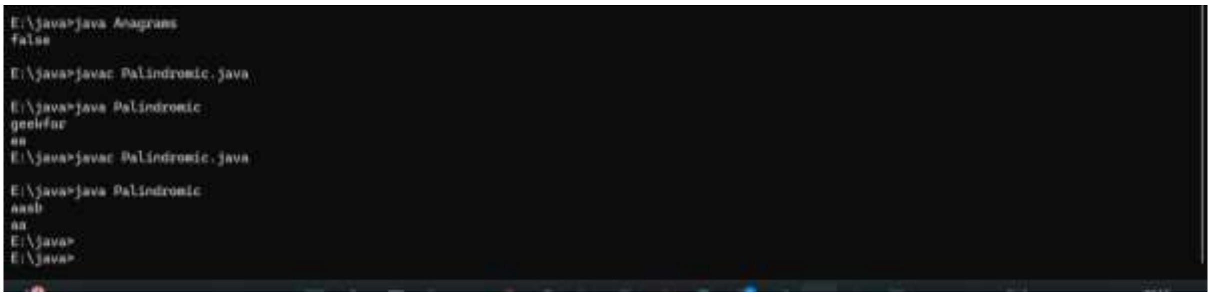
Input: str = “forgeeksskeegfor” Output: “geeksskeeg” Explanation: There are several possible palindromic substrings like “kssk”, “ss”, “eeksskee” etc. But the substring “geeksskeeg” is the longest among all. Input: str = “Geeks” Output: “ee” Input: str = “abc” Output: “a” Input: str = “” Output: “”

Code:

```
import java.util.*;

public class Palindromic{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String s = sc.nextLine();

        boolean flag=true;


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

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

                StringBuilder x = new StringBuilder();

                String a = s.substring(j, s.length() - i + j);
                x.append(a);
                String b = x.reverse().toString();

                if (a.equals(b) && flag==true) {
                    System.out.print(a);
                    flag=false;
                }
            }
        }
    }
}
```

Output:

Time Complexity:o(n*n)



```
E:\java>java Anagrams
false

E:\java>javac Palindromic.java

E:\java>java Palindromic
geekfor
ee
E:\java>javac Palindromic.java

E:\java>java Palindromic
aaab
aa
E:\java>
E:\java>
```

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.

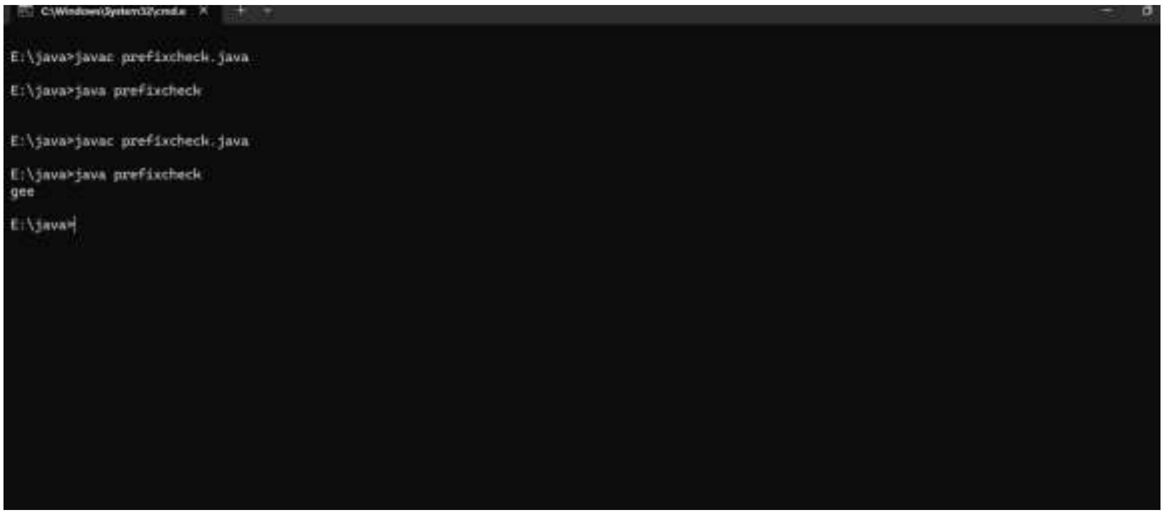
Input: arr[] = ["hello", "world"] Output: -1 Explanation: There"s no common prefix in the given strings

Code:

```
import java.util.*;
public class prefixcheck{
    public static String solve(String[] str){
        String ans="";
        Arrays.sort(str);
        String first = str[0];
        String second = str[str.length-1];
        int n= Math.min(first.length(),second.length());
        for(int i=0;i<n;i++){
            if(first.charAt(i) != second.charAt(i)){
                return ans;
            }
            ans+=first.charAt(i);


        }
        return ans.toString();
    }
    public static void main(String[] arg){
   // String arr[] = {"geeksforgeeks", "geeks", "geek", "geezer"};
    String arr[] = {"hello", "world"};
       String st = solve(arr);
       System.out.println(arr.length == 0? -1 : st );


    }
}
```

Output:

Time complexity:O(n)



```
E:\java>javac prefixcheck.java

E:\java>java prefixcheck


E:\java>javac prefixcheck.java

E:\java>java prefixcheck
gee

E:\java>
```

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.

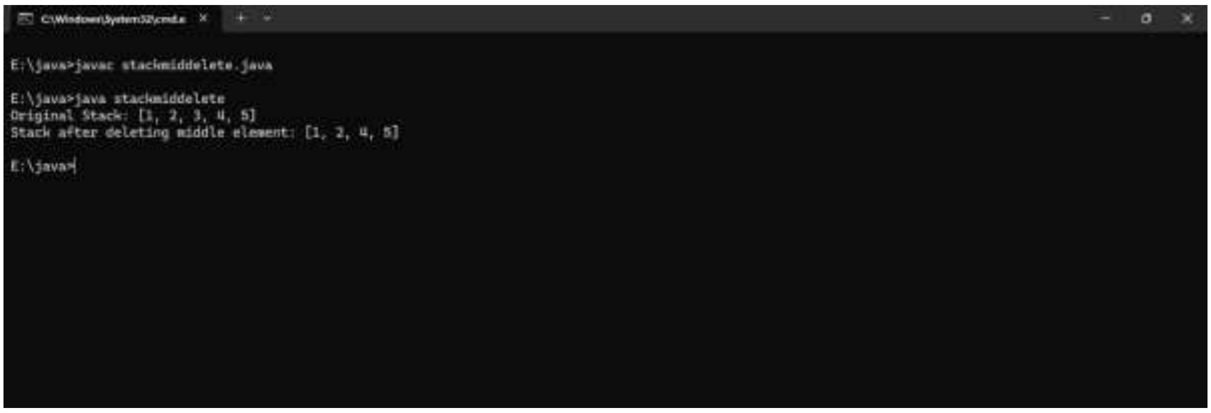
Input : Stack[] = [1, 2, 3, 4, 5] Output : Stack[] = [1, 2, 4, 5]

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

Code :

```
import java.util.*;
public class stackmiddelete{
    public static void delete(Stack<Integer> st, int size, int c){
        if(c == size/2){
            st.pop();
            return;
        }
        int top = st.pop();
        delete(st,size,c+1);
        st.push(top);
    }
    public static void main(String[] arg){
    Stack<Integer> stack = new Stack<>();



        stack.push(1);
        stack.push(2);
        stack.push(3);
        stack.push(4);
        stack.push(5);

        System.out.println("Original Stack: " + stack);



        delete(stack, stack.size(), 0);

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

    }
}
```

Output:O(n)



```
E:\java>javac stackmiddelete.java

E:\java>java stackmiddelete
Original Stack: [1, 2, 3, 4, 5]
Stack after deleting middle element: [1, 2, 4, 5]

E:\java>
```

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

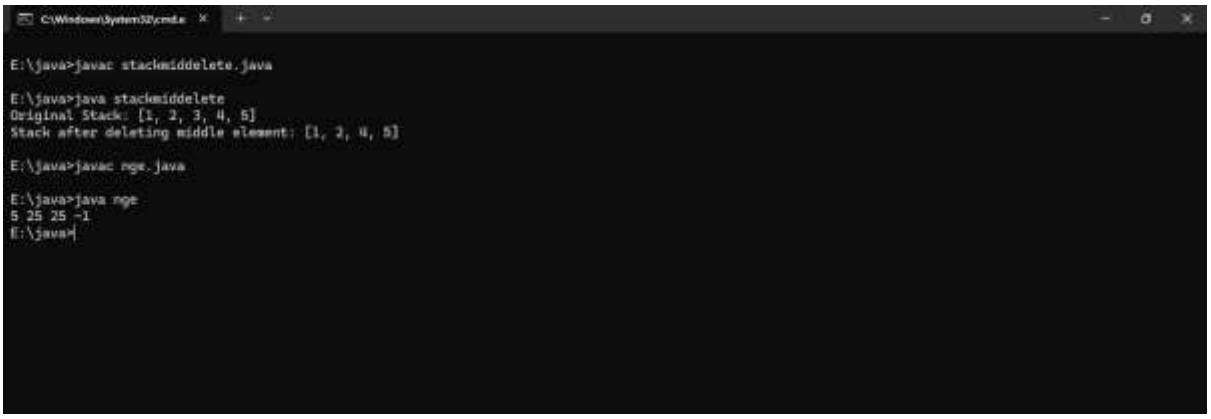
Input: arr[] = [13 , 7, 6 , 12] Output: 13 –> -1 7 –> 12 6 –> 12 12 –> -1 Explanation: 13 and 12 don‟t have any element greater than them present on the right side

Code:

```
import java.util.*;
public class nge{
    public static int[] solve(int[] nums) {
        int n = nums.length;
        Stack<Integer> st = new Stack<>();
        for (int i=n-1;i>=0;i--) {
            st.push(nums[i]);
        }
        int ans[] = new int[n];
        for (int i=n-1;i>=0;i--) {
            while(!st.isEmpty() && st.peek()<=nums[i]) {
                st.pop();
            }
            ans[i] = st.isEmpty() ? -1 : st.peek();
            st.push(nums[i]);
        }
        return ans;
    }
    public static void main(String[] args) {
        int[] nums = {4,5,2,25};
        int n = nums.length;
        int[] ans = solve(nums);
        for (int i=0;i<n;i++) {
            System.out.print(ans[i] + " ");
        }
    }
}
```

Output:

Time complexity:O(n);



```
E:\java>javac stackmiddelete.java

E:\java>java stackmiddelete
Original Stack: [1, 2, 3, 4, 5]
Stack after deleting middle element: [1, 2, 4, 5]

E:\java>javac nge.java

E:\java>java nge
5 25 25 -1
E:\java>
```

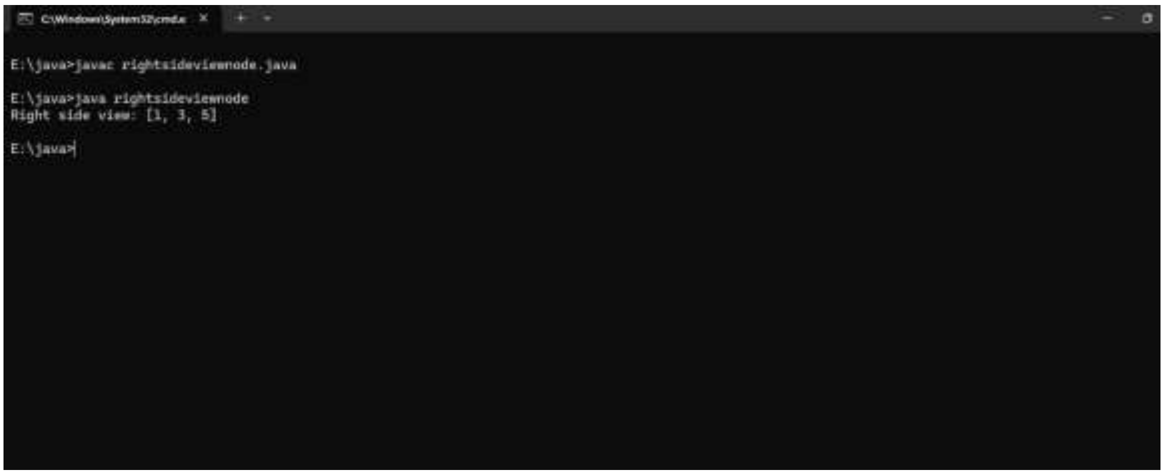
19) Print Right View of a Binary Tree Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

Code :

```
import java.util.ArrayList;
import java.util.List;
//tc=o(n)
//sp=0(H) //
class TreeNode {
    int data;
    TreeNode left;
    TreeNode right;

    TreeNode(int data) {
        this.data = data;
    }

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

public class rightsideviewnode {
    public List<Integer> rightview(TreeNode root) {
        List<Integer> ans = new ArrayList<>();
        level(root, ans, 0);
        return ans;
    }

    public void level(TreeNode root, List<Integer> ans, int c) {
        if (root == null) {
            return;
        }
        if (c == ans.size()) {
            ans.add(root.data);
        }
        level(root.right, ans, c + 1);
        level(root.left, ans, c + 1);
    }

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

        System.out.println("Right side view: " + tree.rightview(root));
    }
}
```

Output:

Timecomplxity:O(n)



```
E:\java>javac rightsideviewnode.java

E:\java>java rightsideviewnode
Right side view: [1, 3, 5]

E:\java>
```

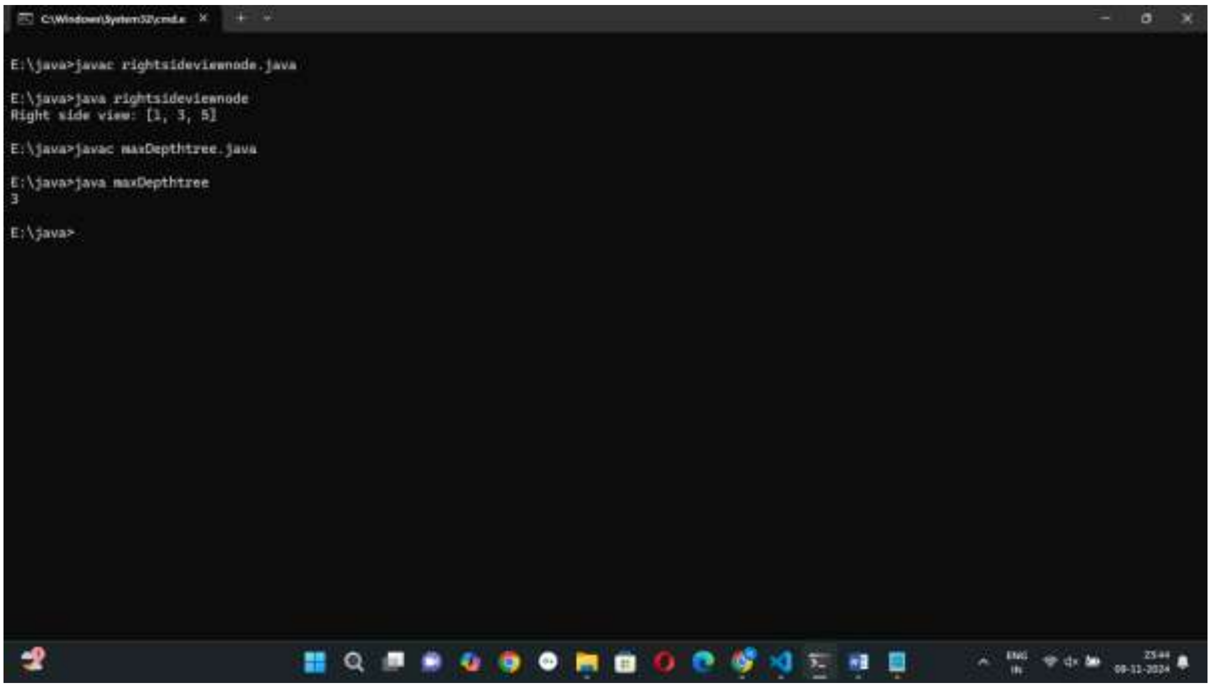
20) . Maximum Depth or Height of Binary Tree Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

Code:

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

public class maxDepthtree{
    private int height(TreeNode root){
        if(root == null){
            return 0;
        }
       int lh = height(root.left);
       int rh = height(root.right);
    return 1+Math.max(lh,rh);
    }
    public static void main(String[] arg){
         maxDepthtree tree = new maxDepthtree();
        TreeNode root =new TreeNode(12);
        root.left = new TreeNode(8);
        root.right = new TreeNode(18);
        root.left.left = new TreeNode(5);
        root.left.right = new TreeNode(11);
        System.out.println(tree.height(root));
    }
}
```

Output:

Timecomplxity:O(n)



```
E:\java>javac rightsideviewnode.java

E:\java>java rightsideviewnode
Right side view: [1, 3, 5]

E:\java>javac maxDepthtree.java

E:\java>java maxDepthtree
3

E:\java>
```