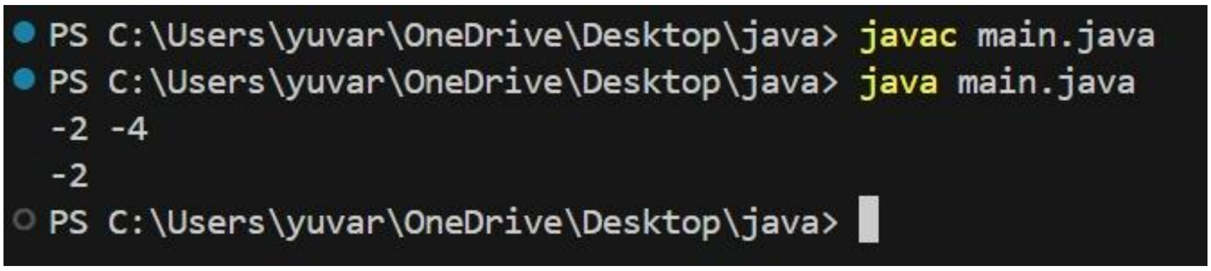
1. Maximum Subarray Sum – Kadane‟s Algorithm: Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.
   Input: arr[] = {2, 3, -8, 7, -1, 2, 3}
   Output: 11
   Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.
   Input: arr[] = {-2, -4}
   Output: –2
   Explanation: The subarray {-2} has the largest sum -2.

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

   Output: 25

   Explanation: The subarray {5, 4, 1, 7, 8} has the largest sum 25.

```
import java.util.Scanner;


public class Main {
 public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   String[] input = sc.nextLine().split(" ");
   int[] y = new int[input.length];
   for (int i = 0; i < input.length; i++) {
      y[i] = Integer.parseInt(input[i]);
    }
   int u = Integer.MIN_VALUE;
   int sum = 0;
    for (int i = 0; i < y.length; i++) {
      sum += y[i];
      if (sum > u) {
        u = sum;
      }
      if (sum<0) {
        sum=0;
      }
    }
    System.out.println(u);
  }
}
```



```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac main.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java main.java
-2 -4
-2
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

Time Complexity:O(n)

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

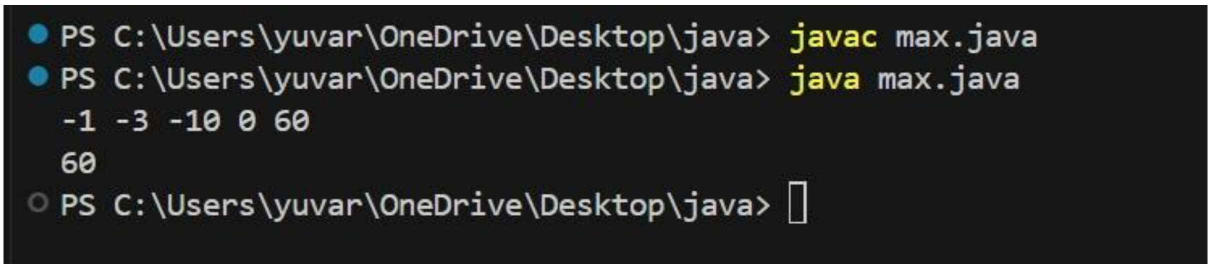
   Input: arr[] = {-1, -3, -10, 0, 60}

   Output: 60

   Explanation: The subarray with maximum product is {60}.

```
import java.util.Scanner;


public class max {
  public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  String[] input = sc.nextLine().split(" ");
  int[] y = new int[input.length];
  for (int i = 0; i < input.length; i++) {
    y[i] = Integer.parseInt(input[i]);
   }
  int b=y[0];
  for (int i=0;i<y.length;i++){
    int s=1;
    for(int j=i;j<y.length;j++){
      s*=y[j];
```

```
        b=Math.max(b,s);
      }
    }
    System.out.println(b);
  }
}
```

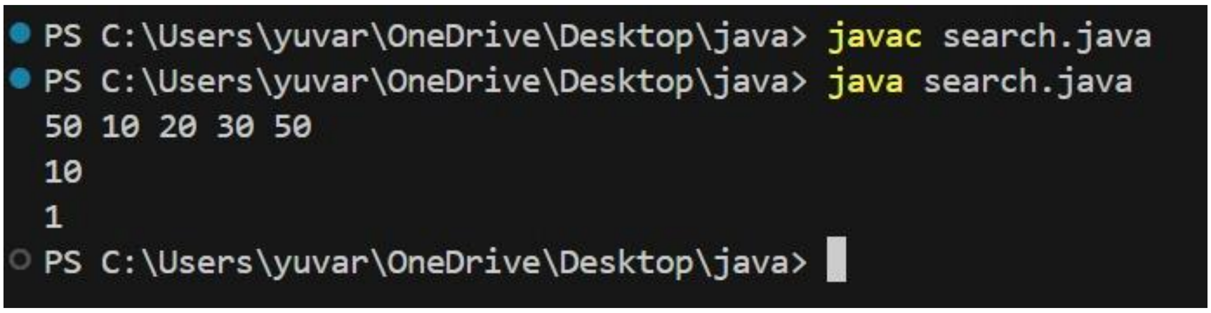


```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac max.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java max.java
-1 -3 -10 0 60
60
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

Time Complexity:O(n)

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

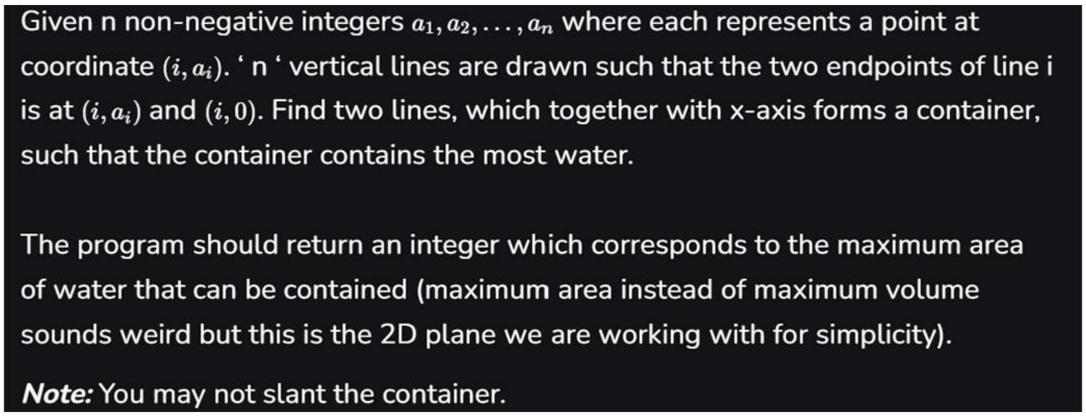
```
import java.util.Scanner;


public class search {
  public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  String[] input = sc.nextLine().split(" ");
  int[] y = new int[input.length];
  for (int i = 0; i < input.length; i++) {
    y[i] = Integer.parseInt(input[i]);
   }
   int u = sc.nextInt();
   for(int i=0;i<y.length;i++){
    if (y[i]==u){
            System.out.println(i);
            return;
          }
        }
        System.out.println(-1);
      }

      }
```



```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac search.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java search.java
50 10 20 30 50
10
1
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

Time Complexity:O(n)

4. Container with Most Water



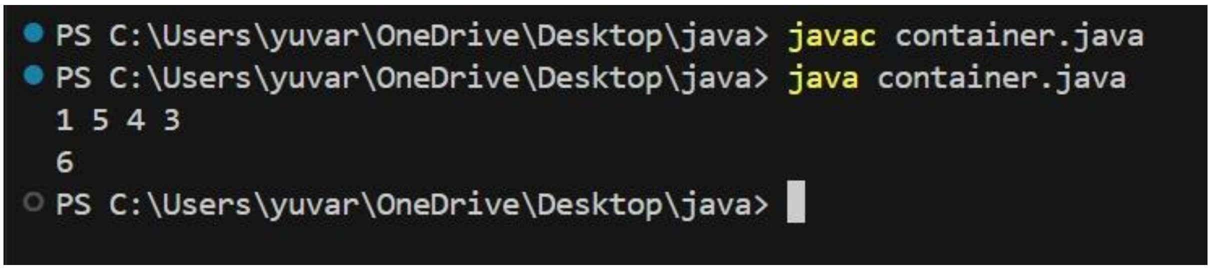
Given n non-negative integers $a_1, a_2, \ldots, a_n$ where each represents a point at coordinate (i, a_i). ‘ n ‘ vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and $(i, 0)$. Find two lines, which together with x-axis forms a container, such that the container contains the most water.

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

Note: You may not slant the container.

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

```
import java.util.Scanner;

public class container {
  public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  String[] input = sc.nextLine().split(" ");
  int[] y = new int[input.length];
  for (int i = 0; i < input.length; i++) {
    y[i] = Integer.parseInt(input[i]);
   }
  int u = 0;
  int v = y.length - 1;
  int area = 0;

  while (u < v) {
    int a = Math.min(y[u], y[v]) * (v- u);
    area = Math.max(area, a);

    if (y[u] < y[v]) {
      u++;
    } else {
      v--;
    }
  }
    System.out.println(area);

}
}
```

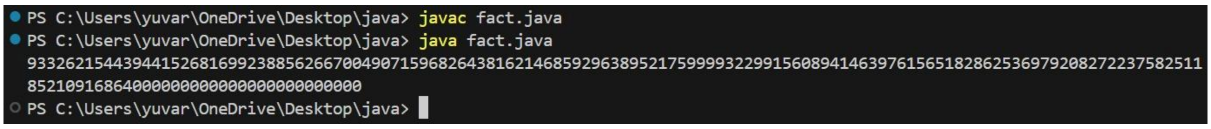


```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac container.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java container.java
1 5 4 3
6
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

Time Complexity:O(n)

5. Find the Factorial of a large number
   Input: 100
   Output:
   93326215443944152681699238856266700490715968264381621468592963895217599993
   2299
   15608941463976156518286253697920827223758251185210916864000000000000000000
   0000 00
   Input: 50
   Output: 30414093201713378043612608166064768844377641568960512000000000000

```
import java.math.BigInteger;

public class fact {

  static BigInteger factorial(int y)

  {

    BigInteger v= new BigInteger("1");

    for (int i = 2; i <= y; i++)

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



    return v;

  }

  public static void main(String args[]) throws Exception

  {

    int y = 100;

    System.out.println(factorial(y));

  }

}
```

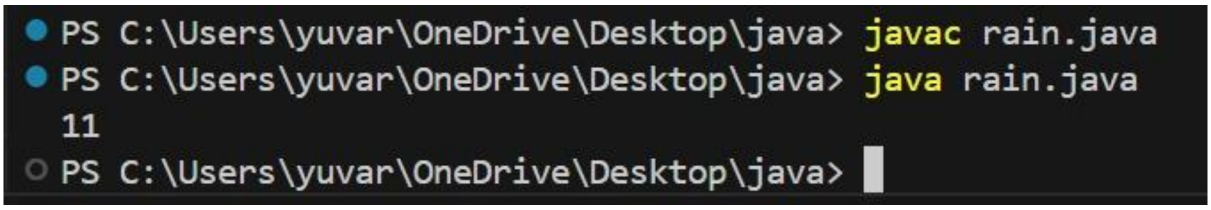


```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac fact.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java fact.java
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511
852109168640000000000000000000000
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

Time Complexity:O(n)

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 Solution {
    public static int rain(int heights[]){

        int leftmax[] = new int[heights.length];
        leftmax[0] = heights[0];
        for(int i=1; i<heights.length; i++){
            leftmax[i] = Math.max(heights[i], leftmax[i-1]);
        }


        int rightmax[] = new int[heights.length];
        rightmax[heights.length-1] = heights[heights.length-1];
        for(int i=heights.length-2; i>=0; i--){
            rightmax[i] = Math.max(heights[i], rightmax[i+1]);
        }


        int trappedWater = 0;
        for(int i=0; i<heights.length; i++){
            int waterlevel = Math.min(leftmax[i], rightmax[i]);
            trappedWater += waterlevel - heights[i];
        }
        return trappedWater;
    }

    public static void main(String args[]){
        int heights[] = {4,2,0,6,3,2,5};
        System.out.println(rain(heights));
    }
}
```

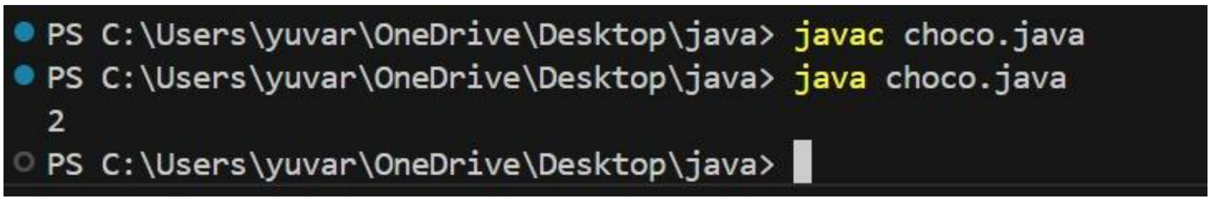


```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac rain.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java rain.java
11
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

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. Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 3 Output: 2 Explanation: If we distribute chocolate packets {3, 2, 4}, we will get the minimum difference, that is 2. Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 5 Output: 7 Explanation: If we distribute chocolate packets {3, 2, 4, 9, 7}, we will get the minimum difference, that is 9 – 2 = 7.

```
import java.util.Arrays;


public class choco {
  public static int dis(int[] y, int u) {
    int n = y.length;
    if (n < u) return -1;
    Arrays.sort(y);
    int mdiff = Integer.MAX_VALUE;
    for (int i = 0; i <= n - u; i++) {
      int diff = y[i + u - 1] - y[i];
      mdiff = Math.min(mdiff, diff);
    }
    return mdiff;
}
public static void main(String[] args) {
int[] y = {7, 3, 2, 4, 9, 12, 56};
int u = 3;


System.out.println( dis(y, u));
}
}
```

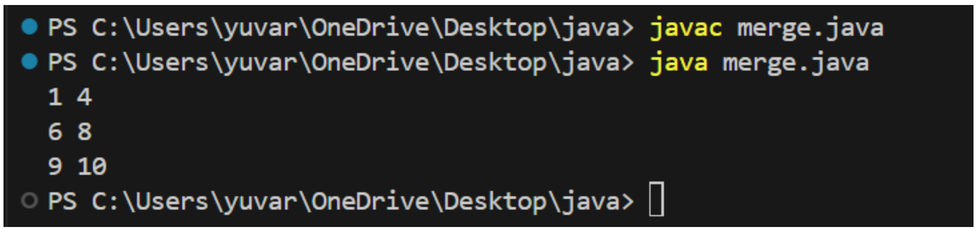


```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac choco.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java choco.java
2
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

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].

```
import java.util.*;


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


    if (y.length == 0) {
      System.out.println(-1);
      return;
    }
    Arrays.sort(y, (a, b) -> a[0] - b[0]);
    LinkedList<int[]> u = new LinkedList<>();


    for (int[] i : y) {
      if (u.isEmpty() || u.getLast()[1] < i[0]) {
        u.add(i);
      } else {
        u.getLast()[1] = Math.max(u.getLast()[1], i[1]);
      }
    }
    int[][] arr = u.toArray(new int[u.size()][]);
    for (int i = 0; i < arr.length; i++) {
      for (int j = 0; j < arr[i].length; j++) {
        System.out.print(arr[i][j] + " ");
      }
      System.out.println();
```

```
        }
    }
}
```

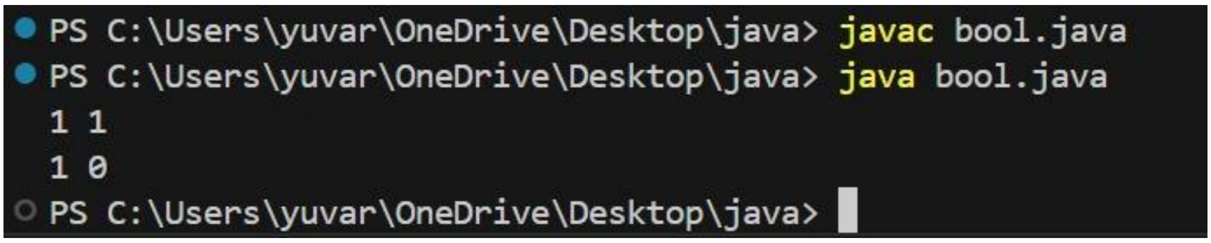


```
● PS C:\Users\yuvar\OneDrive\Desktop\java> javac merge.java
● PS C:\Users\yuvar\OneDrive\Desktop\java> java merge.java
  1 4
  6 8
  9 10
○ PS C:\Users\yuvar\OneDrive\Desktop\java> 
```

Time Complexity:O(n)

9. A Boolean Matrix Question Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1. Input: {{1, 0}, {0, 0}} Output: {{1, 1} {1, 0}} Input: {{0, 0, 0}, {0, 0, 1}} Output: {{0, 0, 1}, {1, 1, 1}} Input: {{1, 0, 0, 1}, {0, 0, 1, 0}, {0, 0, 0, 0}} Output: {{1, 1, 1, 1}, {1, 1, 1, 1}, {1, 0, 1, 1}}

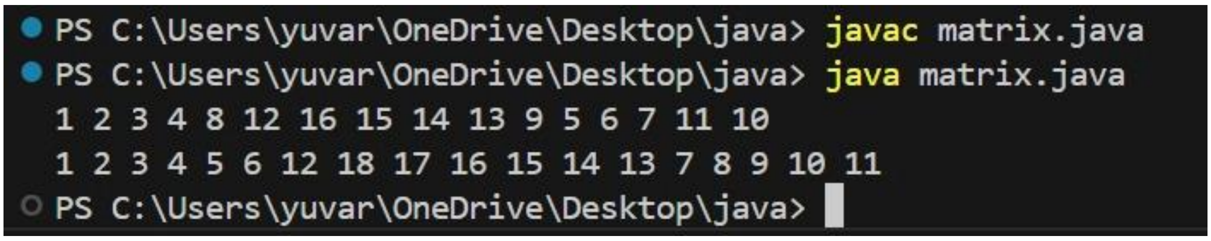
```
import java.util.*;


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


        ArrayList<Integer> row = new ArrayList<>();
        ArrayList<Integer> col = new ArrayList<>();


        int r = y.length;
        int c = y[0].length;
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                if (y[i][j] == 1) {
                    row.add(i);
                    col.add(j);
                }
            }
        }
        for (int i = 0; i < row.size(); i++) {
            int q = row.get(i);
            for (int j = 0; j < c; j++) {
                y[q][j] = 1;
            }
        }
        for (int j = 0; j < col.size(); j++) {
            int p = col.get(j);
            for (int i = 0; i < r; i++) {
                y[i][p] = 1;
            }
        }
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                System.out.print(y[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```



```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac bool.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java bool.java
1 1
1 0
PS C:\Users\yuvar\OneDrive\Desktop\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.

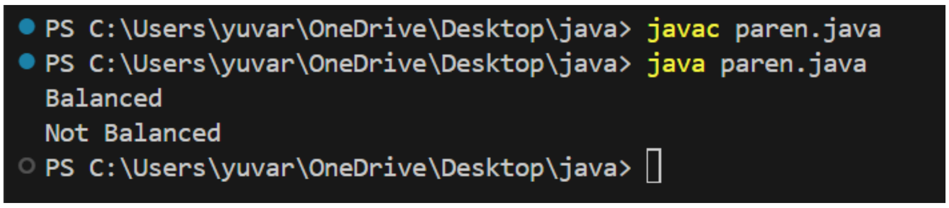
```
public class matrix {
    public static void y(int[][] u) {
        int m = u.length;
    int v = u[0].length;

    int t = 0, b = m - 1, l = 0, r = v - 1;

    while (t <= b && l <= r) {
        for (int i = l; i <= r; i++) {
            System.out.print(u[t][i] + " ");
        }
        t++;
        for (int i = t; i <= b; i++) {
            System.out.print(u[i][r] + " ");
        }
        r--;

        if (t <= b) {
            for (int i = r; i >= l; i--) {
                System.out.print(u[b][i] + " ");
            }
            b--;
        }

        if (l <= r) {
            for (int i = b; i >= t; i--) {
                System.out.print(u[i][l] + " ");
            }
            l++;
        }
    }
}

public static void main(String[] args) {
        int[][] u1 = {

           {1, 2, 3, 4},

           {5, 6, 7, 8},

           {9, 10, 11, 12},

           {13, 14, 15, 16}

        };


        int[][] u2 = {

           {1, 2, 3, 4, 5, 6},

           {7, 8, 9, 10, 11, 12},

           {13, 14, 15, 16, 17, 18}

        };

        y(u1);

        System.out.println();

        y(u2);

    }

}
```



```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac matrix.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java matrix.java
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
PS C:\Users\yuvar\OneDrive\Desktop\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

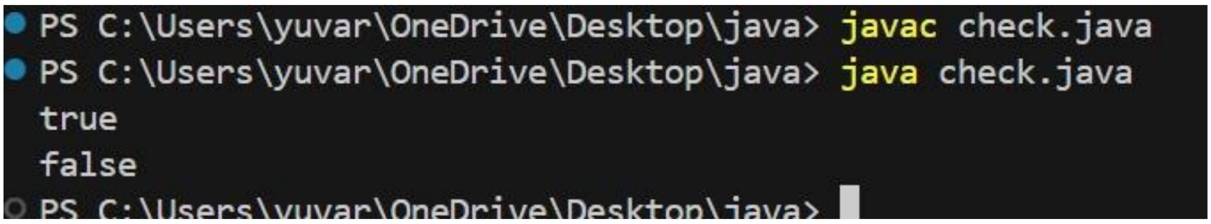
```
public class paren {


    public static String y(String str) {

        int c = 0;
        for (int i = 0; i < str.length(); i++) {
            if (str.charAt(i) == '(') {
                c++;
            } else if (str.charAt(i) == ')') {
                c--;
            }
            if (c < 0) {
                return "Not Balanced";
            }
        }
        return (c == 0) ? "Balanced" : "Not Balanced";
    }

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

        System.out.println(y(str1));
        System.out.println(y(str2));
    }
}
```



```
● PS C:\Users\yuvar\OneDrive\Desktop\java> javac paren.java
● PS C:\Users\yuvar\OneDrive\Desktop\java> java paren.java
  Balanced
  Not Balanced
○ PS C:\Users\yuvar\OneDrive\Desktop\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.

```
import java.util.Arrays;

public class check {

    public static boolean y(String s1, String s2) {
        if (s1.length() != s2.length()) {
            return false;
        }
        char[] str1 = s1.toCharArray();
        char[] str2 = s2.toCharArray();
        Arrays.sort(str1);
        Arrays.sort(str2);
        return Arrays.equals(str1, str2);
    }

    public static void main(String[] args) {
        String s1 = "geeks";
        String s2 = "kseeg";
        System.out.println(y(s1, s2));
        String s3 = "allergy";
        String s4 = "allergic";
        System.out.println(y(s3, s4));
    }
}
```

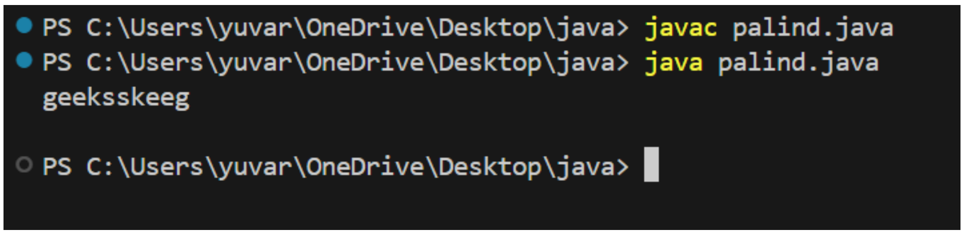


```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac check.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java check.java
true
false
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

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: “”

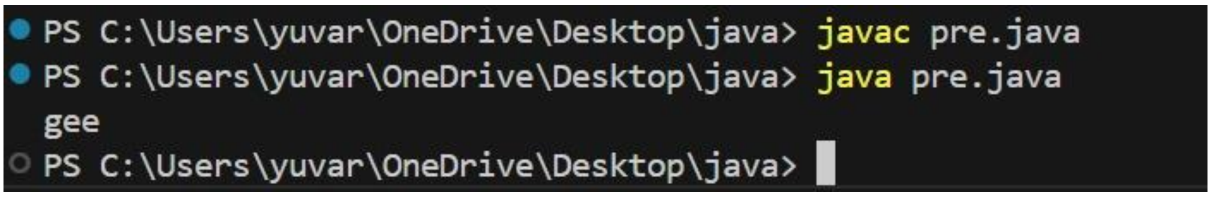
```
public class palind {

   public static String y(String str) {
      int u = str.length();
      if (u == 0) return "";
      int s = 0, m = 1;
      boolean[][] dp = new boolean[u][u];
      for (int i = 0; i < u; i++) {
         dp[i][i] = true;
      }
      for (int length = 2; length <= u; length++) {
         for (int i = 0; i < u - length + 1; i++) {
            int j = i + length - 1;
            if (str.charAt(i) == str.charAt(j)) {
               if (length == 2) {
                  dp[i][j] = true;
               } else {
                  dp[i][j] = dp[i + 1][j - 1];
               }
            }
            if (dp[i][j] && length > m) {
               s = i;
               m = length;
            }
         }
      }
      return str.substring(s, s + m);
   }
```



```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac palind.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java palind.java
geeksskeeg

PS C:\Users\yuvar\OneDrive\Desktop\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.

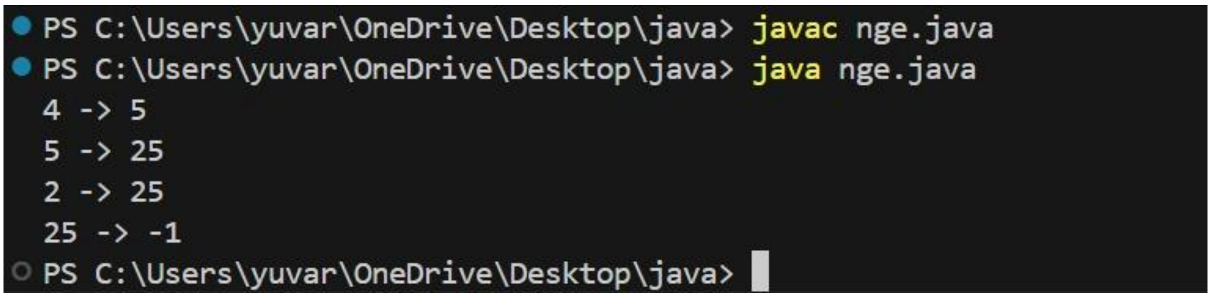
```
public class pre {


  public static String y(String[] u) {
    if (u == null || u.length == 0) return "-1";


    String v = u[0];
    for (int i = 1; i < u.length; i++) {
      while (u[i].indexOf(v) != 0) {
        v = v.substring(0, v.length() - 1);
        if (v.isEmpty()) return "-1";
      }
    }
    return v;
  }
  public static void main(String[] args) {
    String[] u = {"geeksforgeeks", "geeks", "geek", "geezer"};
    System.out.println(y(u));
}
}
```



```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac pre.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java pre.java
gee
PS C:\Users\yuvar\OneDrive\Desktop\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 2 –> 5 –> 25 –> 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 –> 7 -1 –> 12 6 12 –> 12 –> -1 Explanation: 13 and 12 don‟t have any element greater than them present on the right side

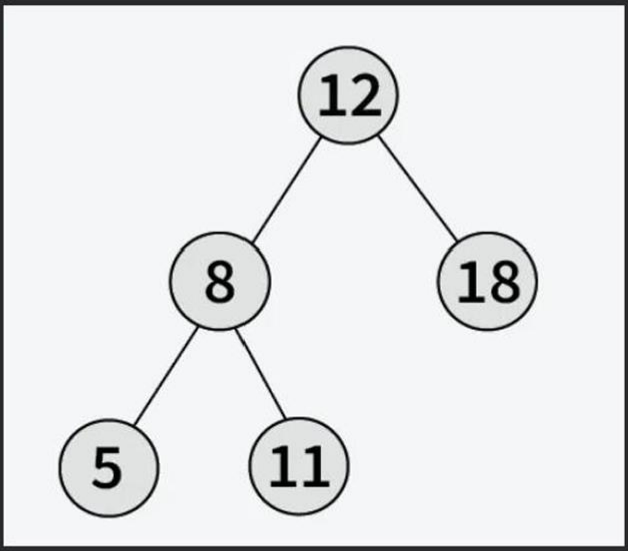
```
import java.util.Stack;


public class nge {


   public static void y(int[] u) {

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

      int[] r = new int[u.length];




      for (int i = u.length - 1; i >= 0; i--) {


         while (!stack.isEmpty() && stack.peek() <= u[i]) {

            stack.pop();

         }

         r[i] = stack.isEmpty() ? -1 : stack.peek();


         stack.push(u[i]);

      }




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

         System.out.println(u[i] + " -> " + r[i]);

      }

   }


   public static void main(String[] args) {
        int[] u1 = {4, 5, 2, 25};

        y(u1);



    }
}
```

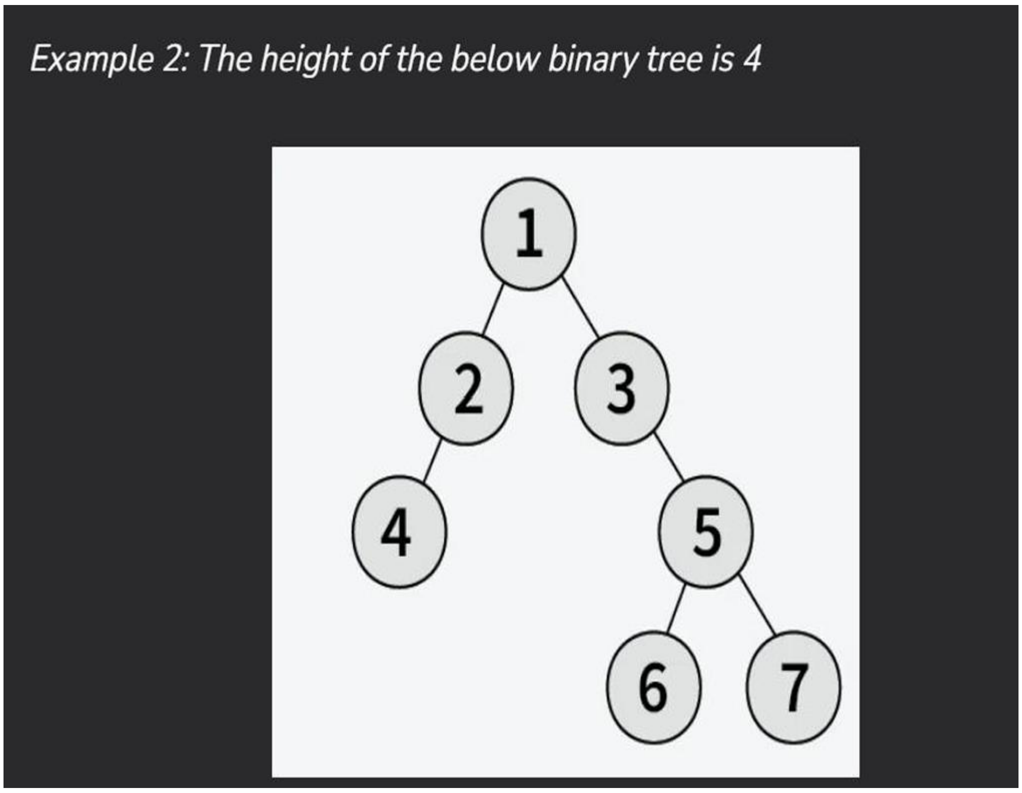


```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac nge.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java nge.java
4 -> 5
5 -> 25
2 -> 25
25 -> -1
PS C:\Users\yuvar\OneDrive\Desktop\java>
```

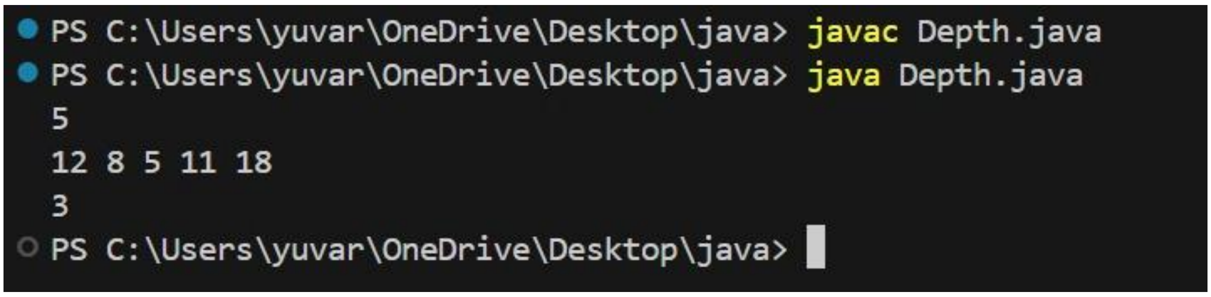
20. Maximum Depth or Height of Binary Tree Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

Example 1: The height of the below binary tree is 3.





```
import java.util.Scanner;
class Depth {
 public int maxDepth(TreeNode root) {
 if (root == null) {
 return 0;
 }
 int a = maxDepth(root.left);
 int b = maxDepth(root.right);
 return Math.max(a, b) + 1;
 }
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 int n = sc.nextInt();
 TreeNode root = null;
 Depth depth = new Depth();
 for (int i = 0; i < n; i++) {
 int value = sc.nextInt();
 root = insertNode(root, value);
 }
 System.out.println( depth.maxDepth(root));
 sc.close();
 }
 public static TreeNode insertNode(TreeNode root, int value) {
 if (root == null) {
 return new TreeNode(value);
 }
 if (value < root.val) {
 root.left = insertNode(root.left, value);
 } else {
 root.right = insertNode(root.right, value);
 }
 return root;
 }
}
class TreeNode {
 int val;
 TreeNode left;
 TreeNode right;
 TreeNode(int val) {
 this.val = val;
 }
}
```



```
PS C:\Users\yuvar\OneDrive\Desktop\java> javac Depth.java
PS C:\Users\yuvar\OneDrive\Desktop\java> java Depth.java
5
12 8 5 11 18
3
PS C:\Users\yuvar\OneDrive\Desktop\java>
```