Coding practice Problems:

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 25.25.

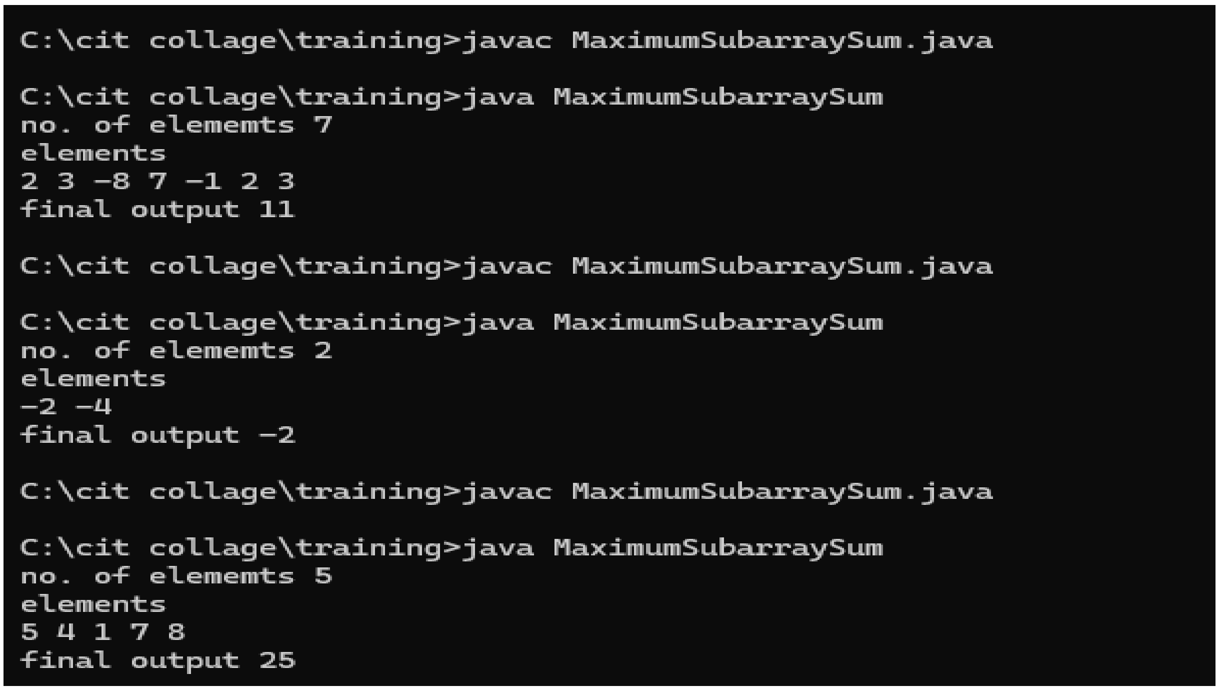
Output

```
import java.util.Scanner;

class MaximumSubarraySum {
  static int maxSubarraySum(int[] arr) {
    int total = arr[0];
    int result = arr[0];

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

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

```
        System.out.print("no. of elememts ");
        int n = scanner.nextInt();

        int[] arr = new int[n];

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

        System.out.println("final output " + maxSubarraySum(arr));
        scanner.close();
    }
}
```

Time complexity : O(n)

Space complexity : O(1)



```
C:\cit collage\training>javac MaximumSubarraySum.java

C:\cit collage\training>java MaximumSubarraySum
no. of elememts 7
elements
2 3 -8 7 -1 2 3
final output 11

C:\cit collage\training>javac MaximumSubarraySum.java

C:\cit collage\training>java MaximumSubarraySum
no. of elememts 2
elements
-2 -4
final output -2

C:\cit collage\training>javac MaximumSubarraySum.java

C:\cit collage\training>java MaximumSubarraySum
no. of elememts 5
elements
5 4 1 7 8
final output 25
```

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}

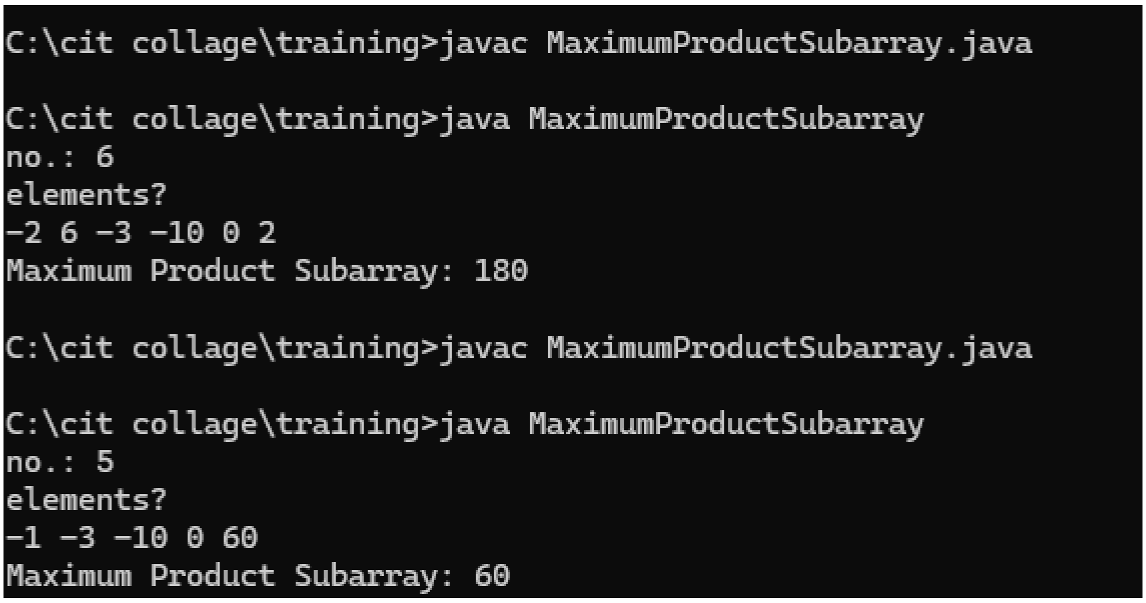
Output: 60

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

```
import java.util.Scanner;

public class MaximumProductSubarray {

  public static int maxProduct(int[] arr) {
    int maxp= arr[0], minp= arr[0],result= arr[0];

    for (int i= 1;i<arr.length; i++) {
      if (arr[i] < 0) {
        int temp = maxp;
        maxp = minp;
        minp = temp;
      }
      maxp = Math.max(arr[i], maxp * arr[i]);
      minp = Math.min(arr[i], minp * arr[i]);
      result = Math.max(result, maxp);
    }

    return result;
  }

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


    System.out.print("no.: ");
    int n = scanner.nextInt();
    int[] arr = new int[n];
    System.out.println("elements?");
    for (int i = 0; i < n; i++) {
      arr[i] = scanner.nextInt();
    }
    System.out.println("Maximum Product Subarray: " + maxProduct(arr));
    scanner.close();
  }
}
```

Time complexity : O(n)

Space complexity : O(1)



```
C:\cit collage\training>javac MaximumProductSubarray.java

C:\cit collage\training>java MaximumProductSubarray
no.: 6
elements?
-2 6 -3 -10 0 2
Maximum Product Subarray: 180

C:\cit collage\training>javac MaximumProductSubarray.java

C:\cit collage\training>java MaximumProductSubarray
no.: 5
elements?
-1 -3 -10 0 60
Maximum Product Subarray: 60
```

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

OUTPUT:

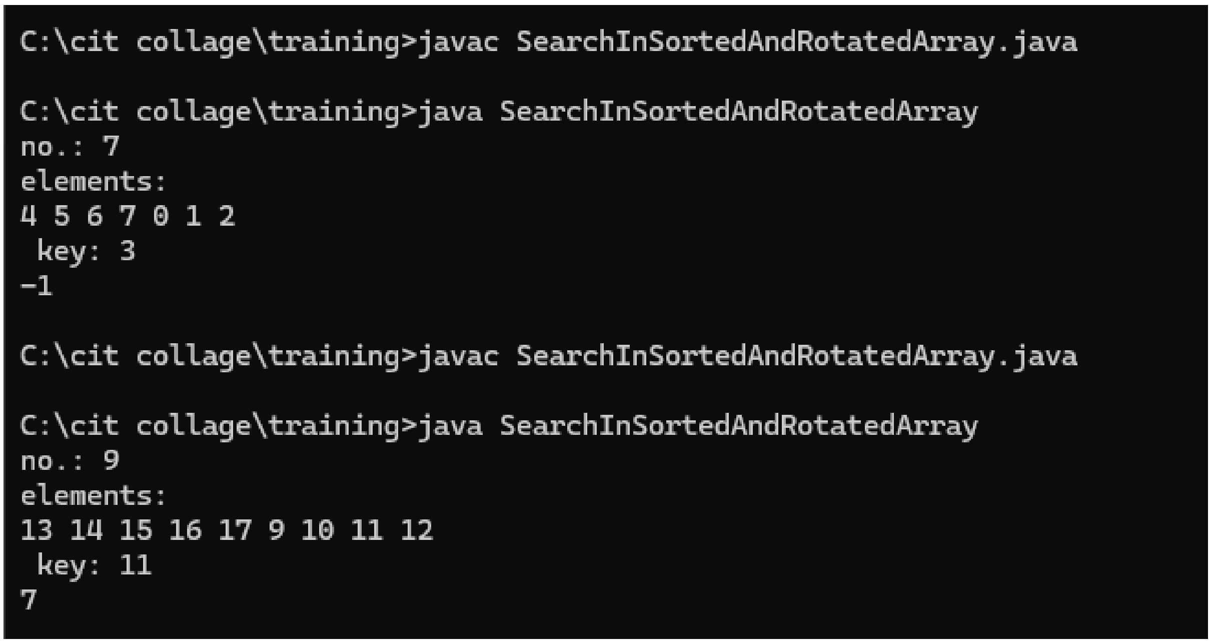
```
import java.util.Scanner;

public class SearchInSortedAndRotatedArray {

    public static int search(int[] arr, int key) {
        int L = 0, R = arr.length - 1;

        while (L <= R) {
            int mid = L + (R - L) / 2;

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

            if (arr[L] <= arr[mid]) {
                if (key >= arr[L] && key < arr[mid]) {
                    R = mid - 1;
                } else {
                    L = mid + 1;
                }
            } else {
```

```
            if (key > arr[mid] && key <= arr[R]) {
                L = mid + 1;
            } else {
                R = mid - 1;
            }
        }
    }

    return -1;
}

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

    System.out.print("no.: ");
    int n = scanner.nextInt();

    int[] arr = new int[n];

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

    System.out.print(" key: ");
    int key = scanner.nextInt();

    int result = search(arr, key);
    System.out.println(result);

    scanner.close();
```

```
   }
}
```

Time complexity : O(n)

Space complexity : O(1)



```
C:\cit collage\training>javac SearchInSortedAndRotatedArray.java

C:\cit collage\training>java SearchInSortedAndRotatedArray
no.: 7
elements:
4 5 6 7 0 1 2
 key: 3
-1

C:\cit collage\training>javac SearchInSortedAndRotatedArray.java

C:\cit collage\training>java SearchInSortedAndRotatedArray
no.: 9
elements:
13 14 15 16 17 9 10 11 12
 key: 11
7
```

5. Find the Factorial of a large number

Input: 100

Output:

93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000

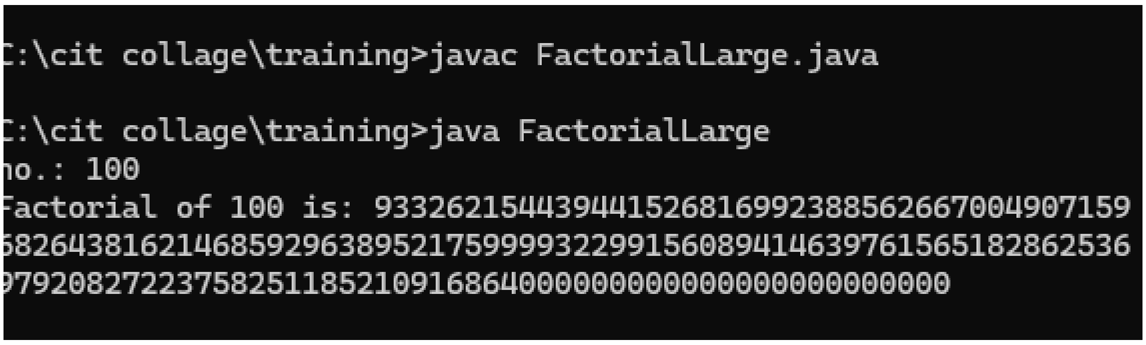
Input: 50

Output: 30414093201713378043612608166064768844377641568960512000000000000

Program:
```
import java.math.BigInteger;

import java.util.Scanner;


public class FactorialLarge {


   public static BigInteger factorial(int n) {

      BigInteger ans = BigInteger.ONE;

      for (int i= 2; i <= n; i++) {

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

      }

      return ans;

   }


   public static void main(String[] args) {

      Scanner scanner=new Scanner(System.in);

      System.out.print("no.: ");

      int n=scanner.nextInt();

      if (n< 0) {

         System.out.println("error.");

      } else {

         System.out.println("Factorial of " + n + " is: " + factorial(n));

      }


      scanner.close();

   }

}
```

time complexity : O(n)

Space complexity :O(logn)



```
:\cit collage\training>javac FactorialLarge.java

:\cit collage\training>java FactorialLarge
no.: 100
actorial of 100 is: 93326215443944152681699238856266700490715
682643816214685929638952175999932299156089414639761565182862536
979208272237582511852109168640000000000000000000000000
```

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

program :

```
import java.util.Scanner;


public class TrappedRainwater {


  public int trap(int[] height) {
    int n = height.length;
    int[] rtl = new int[n];
```

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

        int max = -1, sum = 0;
        for (int i = 0; i < n; i++) {
            if (height[i] > max) max = height[i];
            sum += Math.min(max, rtl[i]) - height[i];
        }

        return sum;
    }

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

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

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

        TrappedRainwater solution = new TrappedRainwater();
        int result = solution.trap(height);

        System.out.println("Total trapped rainwater: " + result);
        scanner.close();
    }
}
```

Output:

```
Enter the number of elements: 7
Enter the heights of the bars:
3 0 1 0 4 0 2
Total trapped rainwater: 10
```

Time Complexity : O(n)

Space 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.

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

public class ChocolateDistribution {

    public static int FindminD(int a[], int n, int m) {
        if (n < m) {
            return -1;
        }

        Arrays.sort(a);
        int minD = Integer.MAX_VALUE;
        for (int i = 0; i <= n - m; i++) {
            int d = a[i + m - 1] - a[i];
            minD = Math.min(minD, d);
        }

        return minD;
    }

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

        System.out.print("number of chocolate packets: ");
        int n = sc.nextInt();

        int[] a = new int[n];
        System.out.println("number of chocolates in each packet:");
        for (int i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }

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

        int result = FindminD(a, n, m);

        System.out.println("The minimum difference is: " + result);

        sc.close();
    }
}
```

```
output:

number of chocolate packets: 7

number of chocolates in each packet:

7 3 2 4 9 12 56

number of students: 3
```

time and space complexity :O(n log n) , O(n)

8. Merge Overlapping Intervals Given an array of time intervals where arr[i] = [starti, endi], the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals. Input: arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]] Output: [[1, 4], [6, 8], [9, 10]] Explanation: In the given intervals, we have only two overlapping intervals [1, 3] and [2, 4]. Therefore, we will merge these two and return [[1, 4}], [6, 8], [9, 10]]. Input: arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]] Output: [[1, 6], [7, 8]] Explanation: We will merge the overlapping intervals [[1, 5], [2, 4], [4, 6]] into a single interval [1, 6].

```
import java.util.*;


public class M {


  public static int[][] m(int[][] I) {
    if (I == null || I.length == 0) return I;


    Arrays.sort(I, (a, b) -> a[0] - b[0]);


    Stack<int[]> S = new Stack<>();


    for (int[] i : I) {
      if (S.isEmpty() || S.peek()[1] < i[0]) {
        S.push(i);
      } else {
        int[] t = S.pop();
        S.push(new int[]{t[0], Math.max(t[1], i[1])});
```

```
            }
        }

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

    public static void main(String[] args) {
        int[][] I1 = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
        int[][] I2 = {{7, 8}, {1, 5}, {2, 4}, {4, 6}};

        System.out.println("Merged intervals 1: " + Arrays.deepToString(m(I1)));
        System.out.println("Merged intervals 2: " + Arrays.deepToString(m(I2)));
    }
}
```

output

```
Merged intervals 1: [[1, 4], [6, 8], [9, 10]]
Merged intervals 2: [[1, 6], [7, 8]]
```

time complexity :O(n log n)

Space complexity :O(n)

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

```
public class BooleanMatrix {

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

        boolean rowFlag = false, colFlag = false;
        for (int j = 0; j < N; j++) if (mat[0][j] == 1) rowFlag = true;
        for (int i = 0; i < M; i++) if (mat[i][0] == 1) colFlag = true;

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

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

        if (rowFlag) for (int j = 0; j < N; j++) mat[0][j] = 1;
        if (colFlag) for (int i = 0; i < M; i++) mat[i][0] = 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) {
        int[][] mat = {{1, 0}, {0, 0}};
        modifyMatrix(mat);
        printMatrix(mat);
    }
}
```

output:

```
{{1, 0}, {0, 0}}

1 1
1 0
```

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 SpiralMatrix {


    public static void printSpiral(int[][] matrix) {
        if (matrix == null || matrix.length == 0) return;


        int m = matrix.length, n = matrix[0].length;
        int[] dirs = {0, 1, 1, 0, 0, -1, -1, 0};
        int top = 0, bottom = m - 1, left = 0, right = n - 1;
        int dirIndex = 0;


        while (top <= bottom && left <= right) {
            System.out.print(matrix[top][left] + " ");
            int newTop = top + dirs[dirIndex % 8], newLeft = left + dirs[(dirIndex + 1) % 8];
            if (newTop <= bottom && newLeft <= right) {
                top = newTop;
                left = newLeft;
                dirIndex++;
            } else {
                top++;
                left--;
            }
        }
    }

    public static void main(String[] args) {
        int[][] matrix1 = {
            {1, 2, 3, 4},
            {5, 6, 7, 8},
            {9, 10, 11, 12},
            {13, 14, 15, 16}
        };

        printSpiral(matrix1);
    }
}
```

output:

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

time complexity : O(M * N)

Space complexity : O(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

```
public class ParenthesesBalance {


    public static String checkBalance(String str) {
        int openCount = 0;

        for (char ch : str.toCharArray()) {
            if (ch == '(') {
                openCount++;
            } else if (ch == ')') {
                openCount--;
            }

            if (openCount < 0) {
                return "Not Balanced";
            }
        }

        if (openCount != 0) {
            return "Not Balanced";
        }

        return "Balanced";
    }

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

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

output:

str = "((()))()()"

Balanced

str = "())((())"

Not Balanced

Time and space complexity : O(N) , O(1)

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.

```
public class AnagramCheck {


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


    int[] count = new int[26];


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


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


    return true;
  }


  public static void main(String[] args) {
    String s1 = "geeks";
```

```
        String s2 = "kseeg";
        System.out.println(anagrams(s1, s2));

        String s1a = "allergy";
        String s2a = "allergic";
        System.out.println(anagrams(s1a, s2a));

        String s1b = "g";
        String s2b = "g";
        System.out.println(anagrams(s1b, s2b));
    }
}
```

Output:\

```
s1 = "geeks", s2 = "kseeg"
true
s1 = "allergy", s2 = "allergic"
false
```

time and space :

O(N) , O(1)

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: “”

```
import java.util.Scanner;

public class LongestPalindromicSubstring {

    public static String longestPalindrome(String str) {
        if (str == null || str.length() == 0) return "";
        int n = str.length();
        boolean[][] dp = new boolean[n][n];
        int start = 0, maxLength = 1;

        for (int i = 0; i < n; i++) {
            dp[i][i] = true;
        }

        for (int len = 2; len <= n; len++) {
            for (int i = 0; i < n - len + 1; i++) {
                int j = i + len - 1;
                if (str.charAt(i) == str.charAt(j)) {
                    if (len == 2) {
                        dp[i][j] = true;
                    } else {
                        dp[i][j] = dp[i + 1][j - 1];
                    }

                    if (dp[i][j] && len > maxLength) {
                        maxLength = len;
                        start = i;
                    }
                }
            }
        }
        return str.substring(start, start + maxLength);
    }

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

```
        String str = sc.nextLine();
        System.out.println(longestPalindrome(str));
    }
}
```

output:

Inpur:

forgeeksskeegfor

output:

geeksskeeg

time and space : O(N2), O(N2)

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.

```
import java.util.Scanner;

public class LongestCommonPrefix {

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

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

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

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

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

input:

```
4
geeksforgeeks
geeks
geek
geezer
```

output:

```
gee
```

Time Complexity:
$O(N\log N + M)$

Space : $O(1)$

17. Delete middle element of a stack Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure. Input : Stack[] =

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

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

public class DeleteMiddleElement {

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

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

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

        for (int i = 0; i < n; i++) {
            stack.push(sc.nextInt());
        }

        if (n % 2 != 0) {
            deleteMiddle(stack, n, 0);
        }
        while (!stack.isEmpty()) {
            System.out.print(stack.pop() + " ");
        }
    }
}
```

output:

```
Input:
5
1 2 3 4 5

Output:
1 2 4 5
```

Time and space :

O(N) , O(N)

18. Next Greater Element (NGE) for every element in given Array Given an array, print the Next Greater Element (NGE) for every element. Note: The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1. Input: arr[] = [4 , 5 , 2 , 25] Output: 4 5 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.Scanner;
import java.util.Stack;

public class NextGreaterElement {

    public static void nextGreater(int[] arr) {
        Stack<Integer> stack = new Stack<>();
        int n = arr.length;
        int[] result = new int[n];
        for (int i = n - 1; i >= 0; i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }
            result[i] = stack.isEmpty() ? -1 : stack.peek();
            stack.push(arr[i]);
        }

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

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        nextGreater(arr);
    }
}
```

input:

```
4
4 5 2 25
```

Output:

```
5
25
25
-1
```

Time and space:

O(N) , O(N)

19.

```java
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

class Node {
    int data;
    Node left, right;

    public Node(int data) {
        this.data = data;
        left = right = null;
    }
}

public class RightViewBinaryTree {

    public static void printRightView(Node root) {
        if (root == null) {
            return;
        }

        Queue<Node> queue = new LinkedList<>();
        queue.add(root);
        while (!queue.isEmpty()) {
            int levelSize = queue.size();

            for (int i = 1; i <= levelSize; i++) {
                Node currentNode = queue.poll();

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

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

    public static void insert(Node root, int data) {
        Queue<Node> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            Node current = queue.poll();

            if (current.left == null) {
                current.left = new Node(data);
                return;
            } else {
                queue.add(current.left);
            }

            if (current.right == null) {
                current.right = new Node(data);
                return;
            } else {
                queue.add(current.right);
            }
        }
    }

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

        System.out.print("no. of nodes: ");
        int n = scanner.nextInt();
        if (n <= 0) {
            System.out.println(" tree must have at least one node.");
            return;
        }

        System.out.println("Enter the values of the nodes:");
        Node root = new Node(scanner.nextInt());

        for (int i = 1; i < n; i++) {
            insert(root, scanner.nextInt());
        }

        System.out.println("Right view of the binary tree is:");
        printRightView(root);
        scanner.close();
    }
}
```

Output:

```
Right view of the binary tree is:
1 3 7
```

Time and space :

O(N) , O(N)