DATE: 09-11-2024

Coding practice Problems

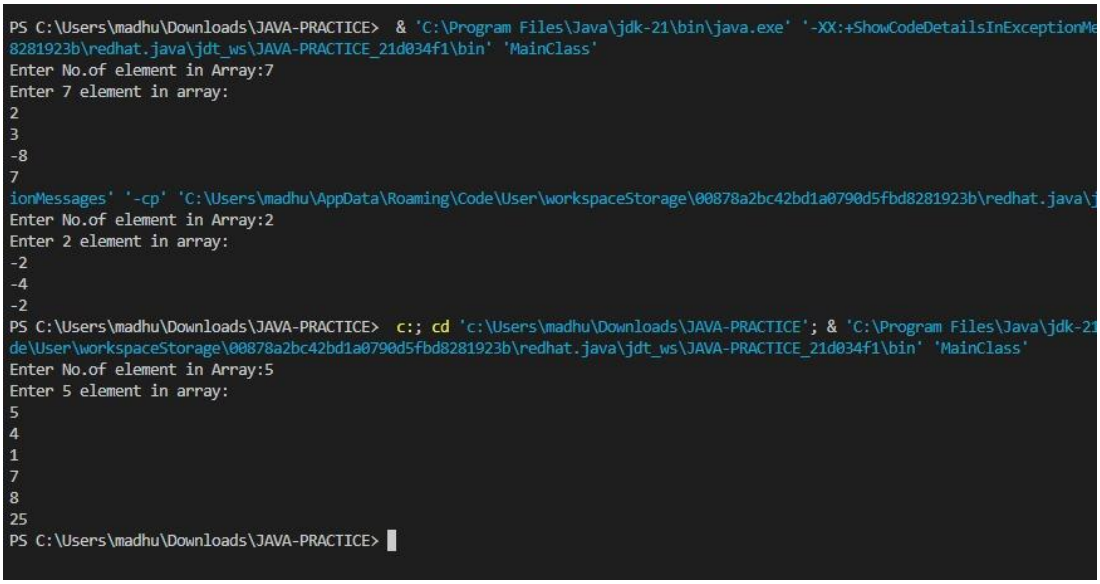
1. Maximum Subarray Sum – Kadane's Algorithm

```
import java.util.*;

public class MainClass {
    static int maximumSubarray(int[] arr) {
        int res = arr[0];
        int maxSum = arr[0];

        for (int i = 1; i < arr.length; i++) {
            maxSum = Math.max(maxSum + arr[i], arr[i]);
            res = Math.max(res, maxSum);
        }
        return res;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter No.of element in Array:");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter " + n + " element in array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        System.out.println(maximumSubarray(arr));
        scanner.close();
    }
}
```

Time Complexity: O()

Reason: We are traversing the array only one time

OUTPUT:



2. Maximum Product Subarray

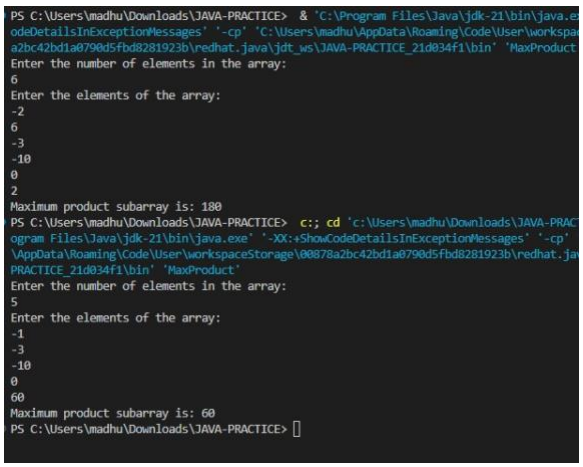
```
import java.util.*;

public class MaxProduct {
    static int maxProduct(int[] arr) {
        int n = arr.length;
        int result = arr[0];
        for (int i = 0; i < n; i++) {
            int mul = 1;

            for (int j = i; j < n; j++) {
                mul *= arr[j];
                result = Math.max(result, mul);
            }
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("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("Maximum product subarray is: " +
maxProduct(arr));
```

```
    }
}
```

Time Complexity: O(n^2)

Reason: Every pair of starting and ending indices of subarrays is considered in the nested loops

OUTPUT:



3. Search in a sorted and rotated Array

```
import java.util.Scanner;

class SearchArray {
    static int search(int arr[], int l, int h, int key) {
        if (l > h)
            return -1;
        int mid = (l + h) / 2;
        if (arr[mid] == key)
            return mid;
        if (arr[l] <= arr[mid]) {
            if (key >= arr[l] && key <= arr[mid])
                return search(arr, l, mid - 1, key);
            return search(arr, mid + 1, h, key);
        }
        if (key >= arr[mid] && key <= arr[h])
            return search(arr, mid + 1, h, key);
        return search(arr, l, mid - 1, key);
    }
    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("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("Enter the key to be searched:");
        int key = scanner.nextInt();
        int i = search(arr, 0, n - 1, key);
        if (i != -1)
            System.out.println("Index: " + i);
        else
            System.out.println("-1");
    }
}
```

Time Complexity: O(log n)

Reason: Binary Search requires log n comparisons to find the element.

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+Sho
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'SearchArray'
Enter the number of elements in the array:
7
Enter the elements of the array:
4
5
6
7
0
1
2
Enter the key to be searched:
0
Index: 4
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1
Enter the number of elements in the array:
7
Enter the elements of the array:
4
5
1
2
Enter the key to be searched:
3
-1
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1
Enter the number of elements in the array:
5
Enter the elements of the array:
50
10
20
30
40
Enter the key to be searched:
10
Index: 1
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

4. Container with Most Water

```
import java.util.Scanner;

class MostWater {
    static int maxArea(int[] arr) {
        int n = arr.length;
```

```
        int left = 0;
        int right = n - 1;
        int area = 0;
        while (left < right) {
            area = Math.max(area, Math.min(arr[left], arr[right]) *
(right - left));
            if (arr[left] < arr[right])
                left += 1;
            else
                right -= 1;
        }
        return area;
    }

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

        System.out.println("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("Maximum area of water that can be
contained: " + maxArea(arr));
    }
}
```

Time Complexity: O(n)

Reason: Only one traversal of the array is required

OUTPUT:

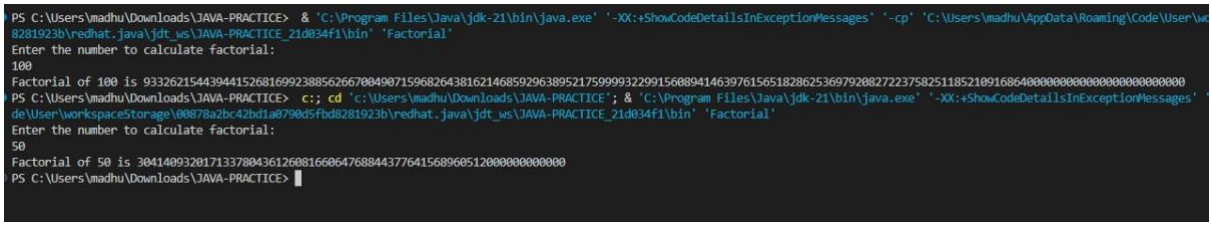
```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCod
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'MostWater'
Enter the number of elements in the array:
4
Enter the elements of the array:
1
5
4
3
Maximum area of water that can be contained: 6
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\Prog
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin
Enter the number of elements in the array:
5
Enter the elements of the array:
3
1
2
4
5
Maximum area of water that can be contained: 12
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

5. Find the Factorial of a large number

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

public class Factorial {
    static BigInteger factorial(int N) {
        BigInteger res = BigInteger.ONE;
        for (int i = 2; i <= N; i++)
            res = res.multiply(BigInteger.valueOf(i));
        return res;
    }

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

        System.out.println("Enter the number to calculate factorial:");
        int N = scanner.nextInt();
        System.out.println("Factorial of " + N + " is " +
factorial(N));
    }
}
```

Time Complexity: O(N)

OUTPUT:



6. Trapping Rain Water

```
import java.util.Scanner;

class TrappedWater {
    static int maxWater(int[] arr) {
        int res = 0;

        for (int i = 1; i < arr.length - 1; i++) {
            int left = arr[i];
            for (int j = 0; j < i; j++)
                left = Math.max(left, arr[j]);
            int right = arr[i];
            for (int j = i + 1; j < arr.length; j++)
                right = Math.max(right, arr[j]);

            res += Math.min(left, right) - arr[i];
        }

        return res;
    }

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

        System.out.println("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("Maximum water that can be stored is: " +
maxWater(arr));
    }
}
```

Time Complexity: O(n)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetail
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'TrappedWater'
Enter the number of elements in the array:
7
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\Program Fil
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'Trap
Enter the number of elements in the array:
5
Enter the elements of the array:
3
0
2
0
4
Maximum water that can be stored is: 7
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\Program Fil
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'Trap
Enter the number of elements in the array:
4
Enter the elements of the array:
1
2
3
4
Maximum water that can be stored is: 0
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\Program Fil
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'Trap
Enter the number of elements in the array:
4
Enter the elements of the array:
10
9
0
5
Maximum water that can be stored is: 5
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

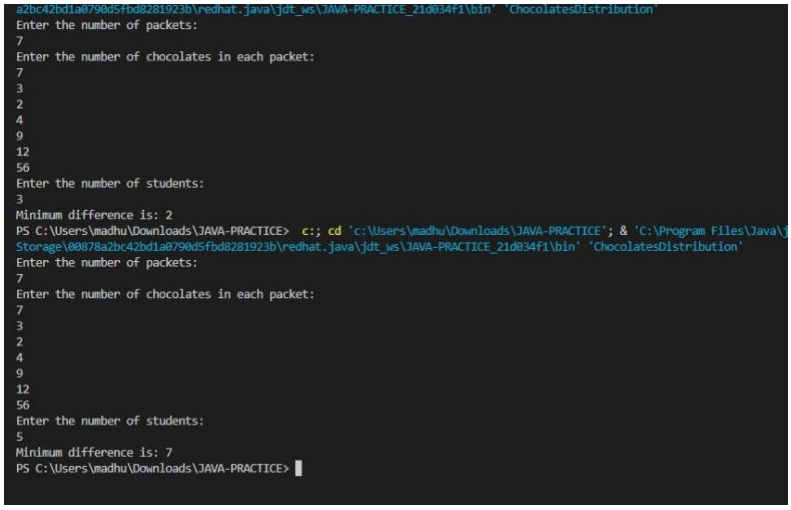
7. Chocolate Distribution Problem

```java
import java.util.Arrays;
import java.util.Scanner;
class ChocolatesDistribution {
    static int findMinDiff(int[] arr, int m) {
        int n = arr.length;
        Arrays.sort(arr);
        int minDiff = Integer.MAX_VALUE;

        for (int i = 0; i + m - 1 < n; i++) {
            int diff = arr[i + m - 1] - arr[i];
            if (diff < minDiff)
                minDiff = diff;
        }
        return minDiff;
```

```
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of packets:");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the number of chocolates in each
packet:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        System.out.println("Enter the number of students:");
        int m = scanner.nextInt();
        System.out.println("Minimum difference is: " + findMinDiff(arr,
m));
    }
}
```

Time Complexity: n log(n)

OUTPUT:



8. Merge Overlapping Intervals

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

class MergeOverlapping {
    static List<int[]> mergeOverlap(int[][] arr) {
        Arrays.sort(arr, (a, b) -> Integer.compare(a[0], b[0]));
        List<int[]> res = new ArrayList<>();
        res.add(new int[] { arr[0][0], arr[0][1] });
        for (int i = 1; i < arr.length; i++) {
            int[] last = res.get(res.size() - 1);
            int[] curr = arr[i];
```

```
            if (curr[0] <= last[1])
                last[1] = Math.max(last[1], curr[1]);
            else
                res.add(new int[] { curr[0], curr[1] });
        }

        return res;
    }

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

        System.out.println("Enter the number of intervals:");
        int n = scanner.nextInt();
        int[][] arr = new int[n][2];
        System.out.println("Enter the intervals:");
        for (int i = 0; i < n; i++) {
            arr[i][0] = scanner.nextInt();
            arr[i][1] = scanner.nextInt();
        }
        List<int[]> res = mergeOverlap(arr);
        System.out.println("Merged Intervals:");
        for (int[] interval : res)
            System.out.println(interval[0] + " " + interval[1]);
    }
}
```

Time Complexity: O(n log(n))

OUTPUT:

```
A-PRACTICE_21d034f1\bin' 'MergeOverlapping'
Enter the number of intervals:
4
Enter the intervals:
1 3
2 4
6 8
9 10
Merged Intervals:
1 4
6 8
9 10
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\Program Files\Java\jdk-2
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'MergeOverlapping'
Enter the number of intervals:
4
Enter the intervals:
7 8
1 5
2 4
4 6
Merged Intervals:
1 6
7 8
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

9. A Boolean Matrix Question

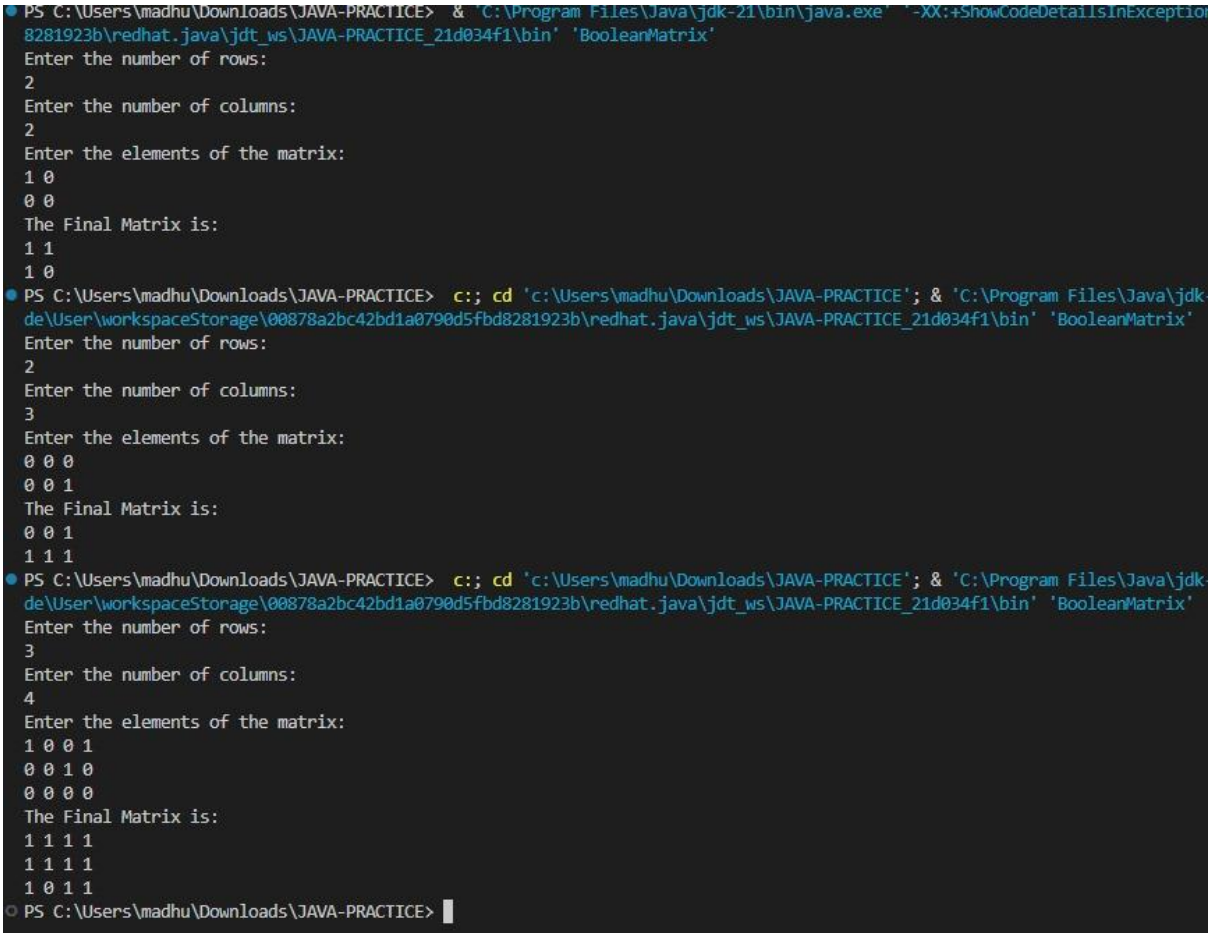
```
import java.util.Scanner;

class BooleanMatrix {
    static void setZeroes(int[][] matrix) {

        int rows = matrix.length;
        int columns = matrix[0].length;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (matrix[i][j] == 1) {
                    int index = i - 1;
                    while (index >= 0) {
                        if (matrix[index][j] != 1) {
                            matrix[index][j] = -1;
                        }
                        index--;
                    }
                    index = i + 1;
                    while (index < rows) {
                        if (matrix[index][j] != 1) {
                            matrix[index][j] = -1;
                        }
                        index++;
                    }
                    index = j - 1;
                    while (index >= 0) {
                        if (matrix[i][index] != 1) {
                            matrix[i][index] = -1;
                        }
```

```
                        index--;
                    }
                    index = j + 1;
                    while (index < columns) {
                        if (matrix[i][index] != 1) {
                            matrix[i][index] = -1;
                        }
                        index++;
                    }
                }
            }
        }
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (matrix[i][j] < 0) {
                    matrix[i][j] = 1;
                }
            }
        }
    }

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

        System.out.println("Enter the number of rows:");
        int rows = scanner.nextInt();
        System.out.println("Enter the number of columns:");
        int columns = scanner.nextInt();
        int[][] arr = new int[rows][columns];
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                arr[i][j] = scanner.nextInt();
            }
        }
        setZeroes(arr);
        System.out.println("The Final Matrix is:");
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr[0].length; j++) {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Time Complexity: O(NM)(N+M)

Reason:O(NM) for traversing through each element and (N+M)for traversing to row and column of elements having value 1.

OUTPUT:



10. Print a given matrix in spiral form

```
import java.util.Scanner;

public class SpiralPrintMatrix {
    public static void printSpiral(int r, int c, int[][] mat) {
        int top = 0, bottom = r - 1, left = 0, right = c - 1;
        while (top <= bottom && left <= right) {
            for (int i = left; i <= right; ++i) {
                System.out.print(mat[top][i] + " ");
            }
            top++;
            for (int i = top; i <= bottom; ++i) {
                System.out.print(mat[i][right] + " ");
            }
            right--;
            if (top <= bottom) {
                for (int i = right; i >= left; --i) {
```

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

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

        System.out.println("Enter the number of rows:");
        int r = scanner.nextInt();
        System.out.println("Enter the number of columns:");
        int c = scanner.nextInt();
        int[][] mat = new int[r][c];
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                mat[i][j] = scanner.nextInt();
            }
        }
        System.out.println("Spiral Order of the given matrix is:");
        printSpiral(r, c, mat);
    }
```

Time Complexity: O(mn)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'SpiralPrintMatrix'
Enter the number of rows:

4
Enter the number of columns:
5 6 7 8
9 10 11 12
13 14 15 16
Spiral Order of the given matrix is:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_
Enter the number of rows:
3
Enter the number of columns:
6
Enter the elements of the matrix:
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
Spiral Order of the given matrix is:
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

11. Check if given Parentheses expression is balanced or not

```
import java.util.*;

public class ParenthesesExpression {
    public static int check(String s) {
        Stack<Character> st = new Stack<>();
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if (c == '(') {
                st.push('(');
            } else if (c == ')') {
                if (st.isEmpty()) {
                    return 0;
                } else {
                    char p = st.peek();
                    if (p == '(') {
                        st.pop();
                    } else {
                        return 0;
                    }
                }
            }
        }
        if (st.isEmpty()) {
            return 1;
        } else {
            return 0;
        }
    }

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

        System.out.println("Enter the string:");
        String s = sc.nextLine();
        if (check(s) == 0) {
            System.out.println("Invalid");
        } else {
            System.out.println("Valid");
        }
    }
}
```

Time Complexity: O(N)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'ParenthesesExpression'
Enter the string:
(((()))()()
Valid
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRA
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRA
Enter the string:
())((()))
Invalid
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

12. Check if two Strings are Anagrams of each other

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

class Anagram {
    static boolean areAnagrams(String s1, String s2) {
        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 sc = new Scanner(System.in);

        System.out.println("Enter the first string:");
        String s1 = sc.nextLine();
        System.out.println("Enter the second string:");
        String s2 = sc.nextLine();
        if (areAnagrams(s1, s2)) {
            System.out.println("The strings are anagrams.");
        } else {
            System.out.println("The strings are not anagrams.");
        }
    }
}
```

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

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetails
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'Anagram'
Enter the first string:
geeks
Enter the second string:
kseeg
The strings are anagrams.
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  c:; cd 'c:\Users\madhu\Downloads\JAVA-PRACTICE'; & 'C:\Program File
de\User\workspaceStorage\00878a2bc42bd1a0790d5fbd8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'Anagr
Enter the first string:
allergy
Enter the second string:
allergic
The strings are not anagrams.
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

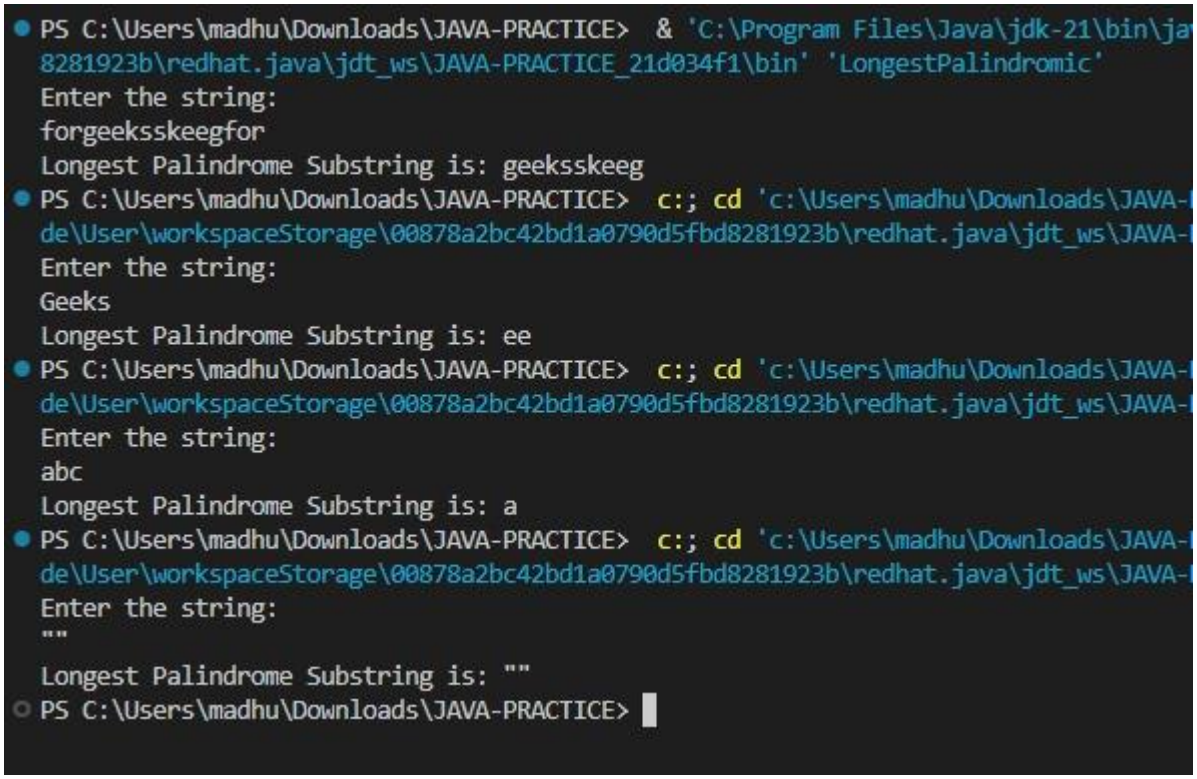
13. Longest Palindromic Substring

```
import java.util.Scanner;

public class LongestPalindromic {
    static String longestPalSubstr(String str) {
        int n = str.length();
        if (n == 0)
            return "";

        int start = 0, maxLen = 1;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j <= 1; j++) {
                int low = i;
                int hi = i + j;
                while (low >= 0 && hi < n && str.charAt(low) ==
str.charAt(hi)) {
                    int currLen = hi - low + 1;
                    if (currLen > maxLen) {
                        start = low;
                        maxLen = currLen;
                    }
                    low--;
                    hi++;
                }
            }
        }
        return str.substring(start, start + maxLen);
    }

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

        System.out.println("Enter the string:");
        String str = sc.nextLine();
```

```
        System.out.println("Longest Palindrome Substring is: " +
longestPalSubstr(str));
    }
}
```

Time Complexity: O(n^2)

OUTPUT:



14. Longest Common Prefix using Sorting

```
public class LongestPalindrome {
    public static String findLongestPalindrome(String s) {
        if (s == null || s.length() == 0) {
            return "";
        }

        int len = s.length();
        int start = 0;
        int maxLen = 1;
        boolean[][] dp = new boolean[len][len];

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

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

```
            if (s.charAt(i) == s.charAt(i + 1)) {
                dp[i][i + 1] = true;
                start = i;
                maxLen = 2;
            }
        }

        for (int l = 3; l <= len; l++) {
            for (int i = 0; i < len - l + 1; i++) {
                int j = i + l - 1;

                if (s.charAt(i) == s.charAt(j) && dp[i + 1][j - 1]) {
                    dp[i][j] = true;
                    if (l > maxLen) {
                        start = i;
                        maxLen = l;
                    }
                }
            }
        }

        return s.substring(start, start + maxLen);
    }

    public static void main(String[] args) {
        String s1 = "forgeeksskeegfor";
        System.out.println("Longest Palindromic Substring: " +
findLongestPalindrome(s1));

        String s2 = "Geeks";
        System.out.println("Longest Palindromic Substring: " +
findLongestPalindrome(s2));

        String s3 = "abc";
        System.out.println("Longest Palindromic Substring: " +
findLongestPalindrome(s3));

        String s4 = "";
        System.out.println("Longest Palindromic Substring: " +
findLongestPalindrome(s4));
    }
}
```

Time Complexity: O(n)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'LongestPalindrome'
Longest Palindromic Substring: geeksskeeg
Longest Palindromic Substring: ee
Longest Palindromic Substring: a
Longest Palindromic Substring:
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

15. Delete middle element of a stack

```
import java.util.*;

public class MidDel {
    static void deleteMiddle(Stack<Integer> stack, int n, int curr) {
        if (stack.isEmpty() || curr == n) {
            return;
        }

        int x = stack.pop();
        deleteMiddle(stack, n, curr + 1);

        if (curr != n / 2) {
            stack.push(x);
        }
    }

    static void deleteMiddle(Stack<Integer> stack) {
        int n = stack.size();
        deleteMiddle(stack, n, 0);
    }

    public static void main(String[] args) {
        Stack<Integer> stack1 = new Stack<>();
        stack1.push(1);
        stack1.push(2);
        stack1.push(3);
        stack1.push(4);
        stack1.push(5);

        System.out.println("Original stack: " + stack1);
        deleteMiddle(stack1);
        System.out.println("Stack after deleting middle element: " +
stack1);

        Stack<Integer> stack2 = new Stack<>();
        stack2.push(1);
        stack2.push(2);
```

```
        stack2.push(3);
        stack2.push(4);
        stack2.push(5);
        stack2.push(6);

        System.out.println("Original stack: " + stack2);
        deleteMiddle(stack2);
        System.out.println("Stack after deleting middle element: " +
stack2);
    }
}
```

Time Complexity: O(n)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\j
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'MidDel'
Original stack: [1, 2, 3, 4, 5]
Stack after deleting middle element: [1, 2, 4, 5]
Original stack: [1, 2, 3, 4, 5, 6]
Stack after deleting middle element: [1, 2, 4, 5, 6]
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

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

```
import java.util.*;

public class NextGreat {
    static void findNextGreater(int[] arr) {
        int len = arr.length;
        int[] nge = new int[len];
        Stack<Integer> stack = new Stack<>();
        Arrays.fill(nge, -1);

        for (int i = len - 1; i >= 0; i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }

            if (!stack.isEmpty()) {
                nge[i] = stack.peek();
            }

            stack.push(arr[i]);
        }

        for (int i = 0; i < len; i++) {
            System.out.println(arr[i] + " --> " + nge[i]);
```

```
        }
    }

    public static void main(String[] args) {
        int[] arr1 = { 4, 5, 2, 25 };
        System.out.println("Next Greater Elements for the array " +
Arrays.toString(arr1) + ":");
        findNextGreater(arr1);

        System.out.println();

        int[] arr2 = { 13, 7, 6, 12 };
        System.out.println("Next Greater Elements for the array " +
Arrays.toString(arr2) + ":");
        findNextGreater(arr2);
    }
}
```

Time Complexity: O(n)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\ja
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'NextGreat'
Next Greater Elements for the array [4, 5, 2, 25]:
4 --> 5
5 --> 25
2 --> 25
25 --> -1

Next Greater Elements for the array [13, 7, 6, 12]:
13 --> -1
7 --> 12
6 --> 12
12 --> -1
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

17. Print Right View of a Binary Tree

```
import java.util.*;

class Node {
    int val;
    Node left, right;

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

```
}

public class BinaryTreeRight {
    public static List<Integer> rightView(Node root) {
        List<Integer> result = new ArrayList<>();
        if (root == null) {
            return result;
        }

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

        while (!queue.isEmpty()) {
            int count = queue.size();
            for (int i = 0; i < count; i++) {
                Node curr = queue.poll();
                if (i == count - 1) {
                    result.add(curr.val);
                }

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

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

        return result;
    }

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

        List<Integer> rightView = rightView(root);
        System.out.println("Right view of the binary tree: " +
rightView);
    }
}
```

Time Complexity: O(n)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.ex
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'BinaryTreeRight'
Right view of the binary tree: [1, 3, 4]
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```

18. Maximum Depth or Height of Binary Tree

```
import java.util.*;

class Node {
    int val;
    Node left, right;

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

public class TreeHeight {
    static int maxDepth(Node root) {
        if (root == null) {
            return 0;
        }

        int leftDepth = maxDepth(root.left);
        int rightDepth = maxDepth(root.right);
        return Math.max(leftDepth, rightDepth) + 1;
    }

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

        System.out.println("Maximum depth or height of the binary tree
is: " + maxDepth(root));
    }
}
```

Time Complexity: O(n)

OUTPUT:

```
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>  & 'C:\Program Files\Java\jdk-21\bin\java.
8281923b\redhat.java\jdt_ws\JAVA-PRACTICE_21d034f1\bin' 'TreeHeight'
Maximum depth or height of the binary tree is: 3
PS C:\Users\madhu\Downloads\JAVA-PRACTICE>
```