Date: 13-11-2024 **PRACTICE SET – 3**

DSA

1. Anagram program

CODE:

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

public class Check {
  public static boolean isAnagram(String str1, String str2) {
    str1 = str1.replaceAll("\\s", "").toLowerCase();
    str2 = str2.replaceAll("\\s", "").toLowerCase();

    if (str1.length() != str2.length()) {
      return false;
    }

    char[] arr1 = str1.toCharArray();
    char[] arr2 = str2.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.print("Enter the first string: ");
    String str1 = sc.nextLine();

    System.out.print("Enter the second string: ");
    String str2 = sc.nextLine();

    if (isAnagram(str1, str2)) {
      System.out.println(str1 + " and " + str2 + " are anagrams.");
    } else {
      System.out.println(str1 + " and " + str2 + " are not anagrams.");
    }
```

```
    sc.close();
  }
}
```

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

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Check
Enter the first string: Triangle
Enter the second string: Integral
Triangle and Integral are anagrams.

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Check
Enter the first string: Hello
Enter the second string: World
Hello and World are not anagrams.
```

2. Row with max 1s'

CODE:

```
import java.util.Scanner;

public class Row {

  public static int rowWithMaxOnes(int[][] matrix) {
    int maxRow = -1;
    int maxCount = 0;

    for (int i = 0; i < matrix.length; i++) {
      int count = countOnes(matrix[i]);
      if (count > maxCount) {
        maxCount = count;
        maxRow = i;
      }
    }

    return maxRow;
  }

  private static int countOnes(int[] row) {
    int left = 0;
    int right = row.length - 1;

    while (left <= right) {
      int mid = left + (right - left) / 2;
      if (row[mid] == 1 && (mid == 0 || row[mid - 1] == 0)) {
        return row.length - mid; // Number of 1s in the row
```

```
        } else if (row[mid] == 1) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }

    return 0; // No 1s in the row
}

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

    System.out.print("Enter the number of rows: ");
    int rows = sc.nextInt();
    System.out.print("Enter the number of columns: ");
    int cols = sc.nextInt();

    int[][] matrix = new int[rows][cols];

    System.out.println("Enter the elements of the matrix row by row (0 or 1 only):");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = sc.nextInt();
            if (matrix[i][j] != 0 && matrix[i][j] != 1) {
                System.out.println("Invalid input! Only 0 and 1 are allowed.");
                return;
            }
        }
    }

    int maxRow = rowWithMaxOnes(matrix);
    System.out.println("Row with max 1s: " + maxRow);

    sc.close();
}
}
```

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

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Row
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements of the matrix row by row (0 or 1 only):
0 1
1 1
Row with max 1s: 1

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Row
Enter the number of rows: 4
Enter the number of columns: 4
Enter the elements of the matrix row by row (0 or 1 only):
0 0 0 1
1 1 1 1
0 0 1 0
1 1 0 0
Row with max 1s: 1
```

3. Longest consequtive subsequence

CODE:

```
import java.util.HashSet;
import java.util.Scanner;

public class Sequence {
  public static int findLongestConsecutiveSubsequence(int[] nums) {
    HashSet<Integer> set = new HashSet<>();
    int longestStreak = 0;

    for (int num : nums) {
      set.add(num);
    }

    for (int num : nums) {
      if (!set.contains(num - 1)) {
        int currentNum = num;
        int currentStreak = 1;

        while (set.contains(currentNum + 1)) {
          currentNum += 1;
          currentStreak += 1;
        }

        longestStreak = Math.max(longestStreak, currentStreak);
      }
    }

    return longestStreak;
```

```
  }

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

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

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

    int result = findLongestConsecutiveSubsequence(nums);
    System.out.println("Length of the longest consecutive subsequence: " + result);

    sc.close();
  }
}
```

```
C:\Users\SUNITHARAJ\Downloads\new\cdc>java Sequence
Enter the number of elements in the array: 6
Enter the elements of the array:
100
2
500
1
3
4
Length of the longest consecutive subsequence: 4
```

4. Longest palindrome in a string

CODE:

```
import java.util.Scanner;

public class Long {
  public static String longestPalindrome(String s) {
    if (s == null || s.length() < 1) return "";
    int start = 0, end = 0;

    for (int i = 0; i < s.length(); i++) {
      int len1 = expandAroundCenter(s, i, i);      // Odd-length palindromes
      int len2 = expandAroundCenter(s, i, i + 1);   // Even-length palindromes
      int len = Math.max(len1, len2);
```

```
            if (len > end - start) {
                start = i - (len - 1) / 2;
                end = i + len / 2;
            }
        }

        return s.substring(start, end + 1);
    }

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

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

        System.out.print("Enter a string: ");
        String s = sc.nextLine();

        System.out.println("Longest palindromic substring: " + longestPalindrome(s));

        sc.close();
    }
}
```

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

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Long
Enter a string: baba
Longest palindromic substring: aba

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Long
Enter a string: babbbaaa
Longest palindromic substring: abbba
```

5. Rat in a maze problem

CODE:

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

```
public class maze {
  private static final int N = 4;

  private static void printSolution(int[][] solution) {
    for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
        System.out.print(solution[i][j] + " ");
      }
      System.out.println();
    }
  }

  private static boolean isSafe(int[][] maze, int x, int y) {
    return (x >= 0 && x < N && y >= 0 && y < N && maze[x][y] == 1);
  }

  private static boolean solveMaze(int[][] maze) {
    int[][] solution = new int[N][N];
    if (!solveMazeUtil(maze, 0, 0, solution)) {
      System.out.println("Solution doesn't exist");
      return false;
    }
    printSolution(solution);
    return true;
  }

  private static boolean solveMazeUtil(int[][] maze, int x, int y, int[][] solution) {
    if (x == N - 1 && y == N - 1 && maze[x][y] == 1) {
      solution[x][y] = 1;
      return true;
    }
    if (isSafe(maze, x, y)) {
      solution[x][y] = 1;
      if (solveMazeUtil(maze, x + 1, y, solution)) {
        return true;
      }
      if (solveMazeUtil(maze, x, y + 1, solution)) {
        return true;
      }
      solution[x][y] = 0;
      return false;
    }
    return false;
  }
```

```
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int[][] maze = new int[N][N];
    System.out.println("Enter the maze matrix (4x4) with 1s and 0s:");
    for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
        maze[i][j] = sc.nextInt();
      }
    }
    solveMaze(maze);
    sc.close();
  }
}
```

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

C:\Users\SUNITHARAJ\Downloads\new\cdc>java maze
Enter the maze matrix (4x4) with 1s and 0s:
1 0 0 0
1 1 0 1
0 1 0 0
1 1 1 1
1 0 0 0
1 1 0 0
0 1 0 0
0 1 1 1
```