Maha Gowri S **12/11/24**

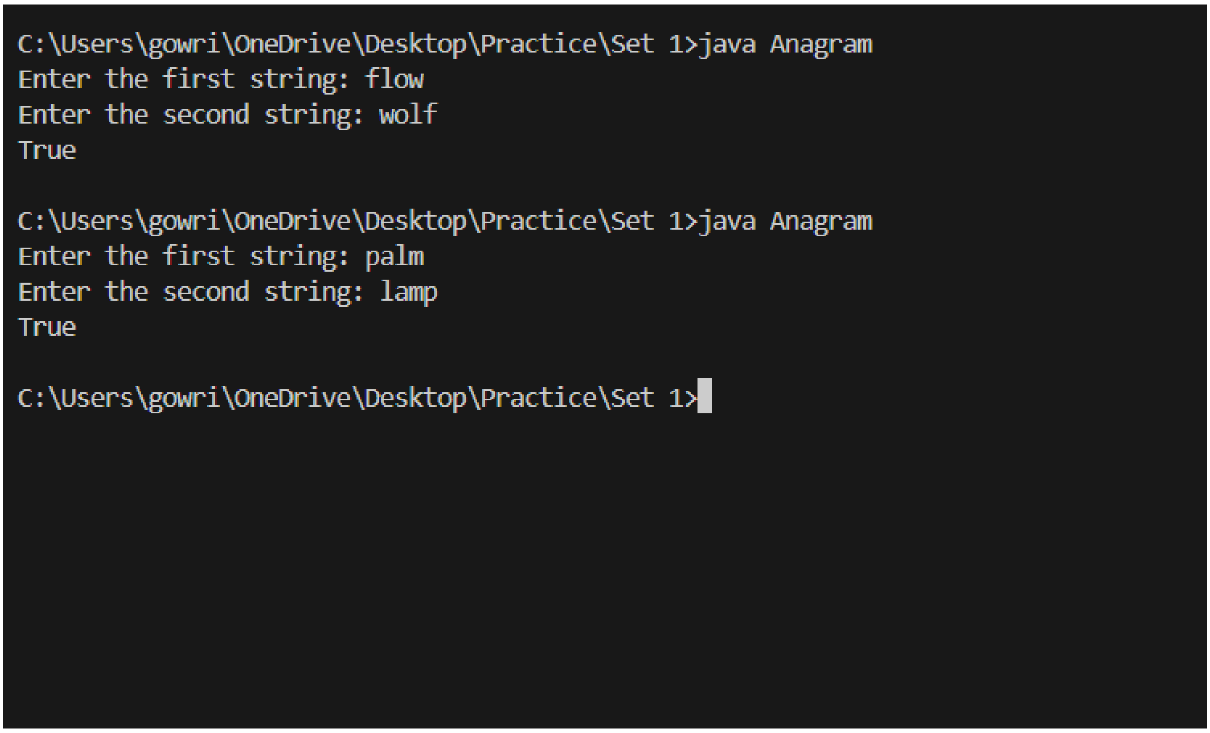
1. Anagram program

Code:

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

class Anagram {
  public boolean isAnagram(String s, String t) {
    char[] sChars = s.toCharArray();
    char[] tChars = t.toCharArray();

    Arrays.sort(sChars);
    Arrays.sort(tChars);

    return Arrays.equals(sChars, tChars);
  }

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

    System.out.print("Enter the first string: ");
    String s = scanner.nextLine();

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

    Anagram anagramChecker = new Anagram();
    boolean result = anagramChecker.isAnagram(s, t);
```

```
        if (result) {
            System.out.println("The strings are anagrams.");
        } else {
            System.out.println("The strings are not anagrams.");
        }
    }
}
```

Output:



```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Anagram
Enter the first string: flow
Enter the second string: wolf
True

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Anagram
Enter the first string: palm
Enter the second string: lamp
True

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>
```

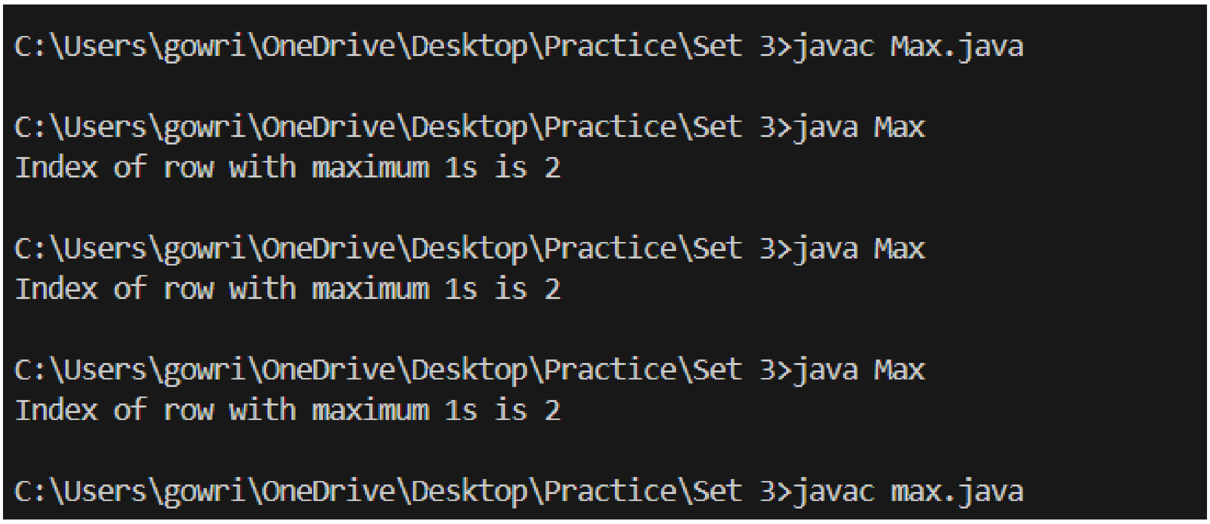
Time Complexity: O(n log n)

2.Row with max 1s'

Code:

```
public class GFG {
    static final int R = 4;
    static final int C = 4;

    // The main function that returns index of row with
// maximum number of 1s
public static int rowWithMax1s(int[][] mat)
{
    int maxRow = -1, row = 0, col = C - 1;

    // Move till we are inside the matrix
    while (row < R && col >= 0) {
        // If the current value is 0, move down to the
        // next row
        if (mat[row][col] == 0) {
            row++;
        }
        // Else if the current value is 1, update maxRow
        // and move to the left column
        else {
            maxRow = row;
            col--;
        }
    }
    return maxRow;
}

// Driver Code
public static void main(String[] args)
{
    int[][] mat = { { 0, 0, 0, 1 },
                    { 0, 1, 1, 1 },
                    { 1, 1, 1, 1 },
                    { 0, 0, 0, 0 } };
        System.out.println(
            "Index of row with maximum 1s is "
            + rowWithMax1s(mat));
    }
}
```

Output:



```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>javac Max.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>java Max
Index of row with maximum 1s is 2

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>java Max
Index of row with maximum 1s is 2

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>java Max
Index of row with maximum 1s is 2

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>javac max.java
```

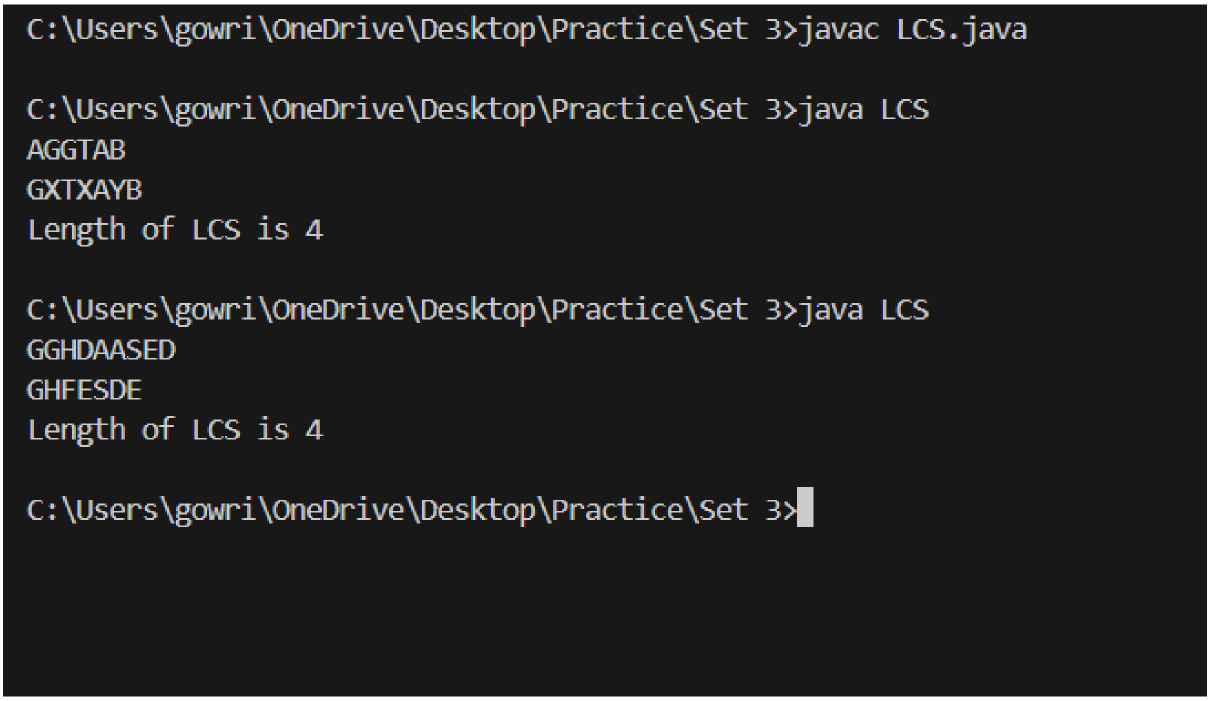
Time Complexity: O(M+N)

3.Longest consecutive subsequence

Code:

```
import java.util.Scanner;
class LCS {

    static int lcs(String S1, String S2) {
        int m = S1.length();
        int n = S2.length();

        int[][] dp = new int[m + 1][n + 1];
        }

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (S1.charAt(i - 1) == S2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1] + 1;
                }
                else {
                    dp[i][j] = Math.max(dp[i - 1][j],
                                        dp[i][j - 1]);
                }
            }
        }

        return dp[m][n];
    }


    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        String S1 = scanner.nextLine();
        String S2 = scanner.nextLine();
        System.out.println("Length of LCS is "
                           + lcs(S1, S2));
        scanner.close();

    }
}
```

Output:



```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>javac LCS.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>java LCS
AGGTAB
GXTXAYB
Length of LCS is 4

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>java LCS
GGHDAASED
GHFESDE
Length of LCS is 4

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>
```

Time Complexity: O(m*n)

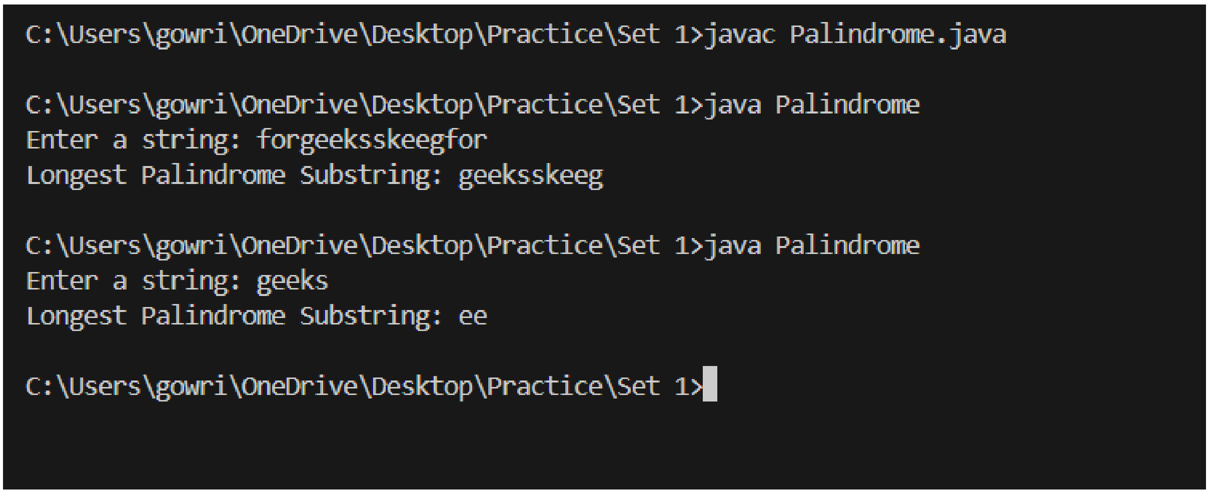
4.Longest palindrome in a string

Code:

```
import java.util.Scanner;


public class Palindrome {


  static String longestPalSubstr(String s) {
    int n = s.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 && s.charAt(low) == s.charAt(hi)) {
                    int currLen = hi - low + 1;
                    if (currLen > maxLen) {
                        start = low;
                        maxLen = currLen;
                    }
                    low--;
                    hi++;
                }
            }
        }

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

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

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

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

Output:



```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Palindrome.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Palindrome
Enter a string: forgeeksskeegfor
Longest Palindrome Substring: geeksskeeg

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Palindrome
Enter a string: geeks
Longest Palindrome Substring: ee

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>
```

Time Complexity: O(n^2)

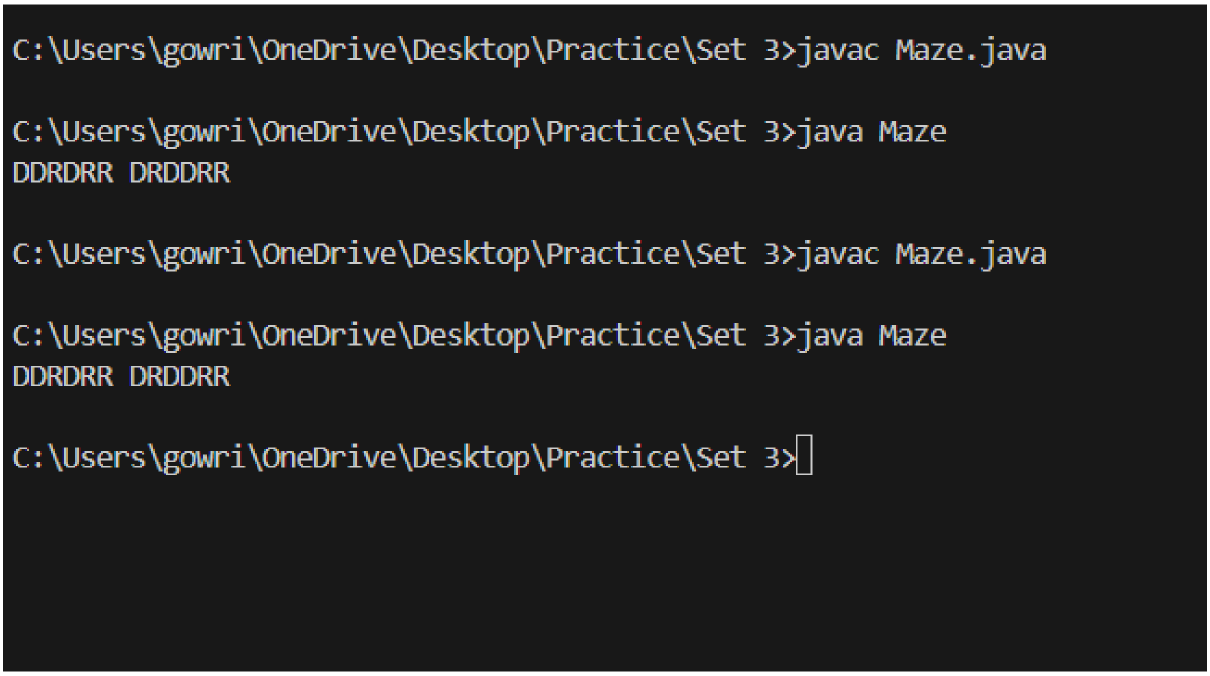
5.Rat in a maze problem

Code:

```
import java.util.ArrayList;

import java.util.List;


public class Maze {


    static String direction = "DLRU";


    static int[] dr = { 1, 0, 0, -1 };

    static int[] dc = { 0, -1, 1, 0 };


    static boolean isValid(int row, int col, int n,

                           int[][] maze)

    {

        return row >= 0 && col >= 0 && row < n && col < n

            && maze[row][col] == 1;

    }


    static void findPath(int row, int col, int[][] maze,
          int n, ArrayList<String> ans,
          StringBuilder currentPath)
{

  if (row == n - 1 && col == n - 1) {
    ans.add(currentPath.toString());
    return;
  }

  maze[row][col] = 0;

  for (int i = 0; i < 4; i++) {

    int nextrow = row + dr[i];

    int nextcol = col + dc[i];

    if (isValid(nextrow, nextcol, n, maze)) {
      currentPath.append(direction.charAt(i));

      findPath(nextrow, nextcol, maze, n, ans,
          currentPath);
      currentPath.deleteCharAt(currentPath.length() - 1);
    }
  }

  maze[row][col] = 1;
}
```

```
    public static void main(String[] args)
    {
        int[][] maze = { { 1, 0, 0, 0 },
                         { 1, 1, 1, 1 },
                         { 1, 1, 0, 0 },
                         { 0, 1, 1, 1 } };

        int n = maze.length;

        ArrayList<String> result = new ArrayList<>();

        StringBuilder currentPath = new StringBuilder();

        if (maze[0][0] != 0 && maze[n - 1][n - 1] != 0) {
            findPath(0, 0, maze, n, result, currentPath);
        }

        if (result.size() == 0)
            System.out.println(-1);
        else
            for (String path : result)
                System.out.print(path + " ");
        System.out.println();
    }
}
```

Output:



```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>javac Maze.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>java Maze
DDRDRR DRDDRR

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>javac Maze.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>java Maze
DDRDRR DRDDRR

C:\Users\gowri\OneDrive\Desktop\Practice\Set 3>
```

Time Complexity: O(3^(m*n))