PRACTICAL NO. 5

Roll no.: 14

Batch: B1

Name: Abhishek Angadi

Aim: Implement Longest Common Subsequence (LCS) algorithm to find the length and LCS for DNA sequences.

TASK-1: Find the similarity between the given X and Y sequence.

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

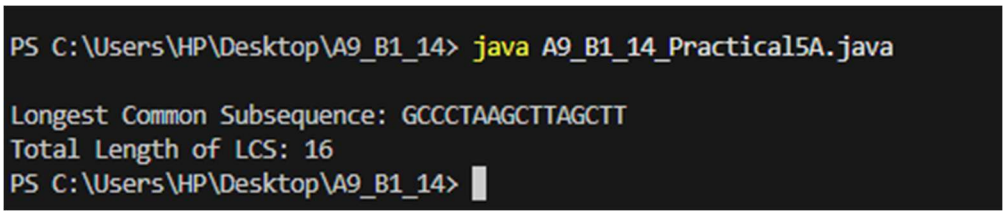
Code:

```
public class A9_B1_14_Practical5A {

    static int[][] Task1(String[] X, String[] Y){
        int n = X.length+1;
        int m = Y.length+1;
        int[][] L = new int[n][m];

        for(int i=0;i<n;i++){
            L[i][0]=0;
        }
        for(int j=0;j<m;j++){
            L[0][j]=0;
        }
        for(int i=1;i<n;i++){
            for(int j=1;j<m;j++){
                if(X[i-1].equalsIgnoreCase(Y[j-1])){
                    L[i][j]=(L[i-1][j-1])+1;
                }
                else{
                    L[i][j]= Integer.max(L[i-1][j], L[i][j-1]);
                }
            }
        }
        return L;
    }
```

```
static void printLCS(String[] X, String[] Y, int[][] L){
    int i = X.length;
    int j = Y.length;
    String lcs = "";

    while(i > 0 && j > 0){
        if(X[i-1].equalsIgnoreCase(Y[j-1])){
            lcs = X[i-1] + lcs;  // prepend matched char
            i--; j--;
        }
        else if(L[i-1][j] > L[i][j-1]){
            i--;
        }
        else{
            j--;
        }
    }

    System.out.println("\nLongest Common Subsequence: " + lcs);
    System.out.println("Total Length of LCS: " + lcs.length());
}

public static void main(String args[]){
    String X="AGCCCTAAGGGCTACCTAGCTT";
    String Y="GACAGCCTACAAGCGTTAGCTTG";
    String[] y = new String[Y.length()];
    String[] x = new String[X.length()];
```

```
        for(int i=0;i<X.length();i++){
            x[i]=X.substring(i,i+1);
        }

        for(int i=0;i<Y.length();i++){
            y[i]=Y.substring(i,i+1);
        }

        int[][] table = Task1(x, y);
        printLCS(x, y, table); //  ◇  Added call to print LCS and its length
    }
}
```

Output:



TASK-2: Find the longest repeating subsequence (LRS). Consider it as a variation of the longest common subsequence (LCS) problem.
Let the given string be S. You need to find the LRS within S. To use the LCS framework, you effectively compare S with itself. So, consider string1 = S and string2 = S.

Code:

```
public class A9_B1_14_Practical5B {

    static int[][] Task2Table(String S) {
        int n = S.length();
        int[][] dp = new int[n + 1][n + 1];

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

        return dp;
    }

    static void printLRS(String S, int[][] dp) {
        int i = S.length(), j = S.length();
        String lrs = "";
```

```
        while (i > 0 && j > 0) {
            if (S.charAt(i - 1) == S.charAt(j - 1) && i != j) {
                lrs = S.charAt(i - 1) + lrs;
                i--;
                j--;
            } else if (dp[i - 1][j] > dp[i][j - 1]) {
                i--;
            } else {
                j--;
            }
        }

        System.out.println("Longest Repeated Subsequence: " + lrs);
        System.out.println("LRS Length: " + lrs.length());
    }

    public static void main(String args[]) {
        String S = "AABCBDC";
        int[][] table = Task2Table(S);
        printLRS(S, table);
    }
}
```

Output:

```
PS C:\Users\HP\Desktop\A9_B1_14> java A9_B1_14_Practical5B.java
Longest Repeated Subsequence: ABC
LRS Length: 3
```

Conclusion:

Hence we successfully implemented LRS and LCS algorithm for the given scenarios.

Github: https://github.com/Shadow3456rh/DAA-Rbu-Practicals/tree/main/Practical%20no%205