DAA Practical 5

Name: Ishita Premchandani

Roll no: 08

Section: A3

Batch: B1

TASK 1:

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

Problem Statement:

(i) DNA sequences can be viewed as strings of A, C, G, and T characters, which represent nucleotides. Finding the similarities between two DNA sequences are an important computation performed in bioinformatics.

[Note that a subsequence might not include consecutive elements of the original sequence.]

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

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

Output: Cost matrix with all costs and direction, final cost of LCS and the LCS.

Length of LCS=16

CODE:

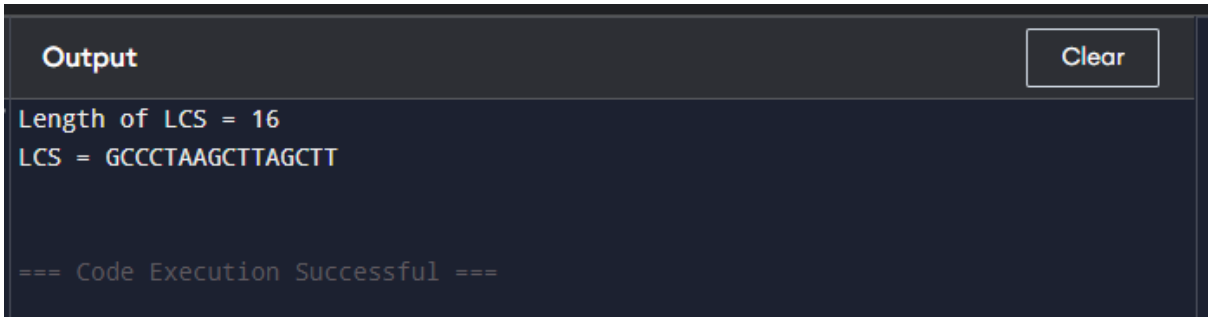
```
#include <stdio.h>
#include <string.h>
```

```
#define MAX 100

void LCS(char X[], char Y[]) {
    int m = strlen(X);
    int n = strlen(Y);
    int C[MAX][MAX];
    char LCS[MAX];
    int i, j;

    for (i = 0; i <= m; i++)
        C[i][0] = 0;
    for (j = 0; j <= n; j++)
        C[0][j] = 0;

    for (i = 1; i <= m; i++) {
        for (j = 1; j <= n; j++) {
            if (X[i - 1] == Y[j - 1])
                C[i][j] = C[i - 1][j - 1] + 1;
            else if (C[i - 1][j] >= C[i][j - 1])
                C[i][j] = C[i - 1][j];
            else
                C[i][j] = C[i][j - 1];
        }
```

```
    }

    int index = C[m][n];
    LCS[index] = '\0';
    i = m;
    j = n;

    while (i > 0 && j > 0) {
        if (X[i - 1] == Y[j - 1]) {
            LCS[index - 1] = X[i - 1];
            i--;
            j--;
            index--;
        } else if (C[i - 1][j] > C[i][j - 1])
            i--;
        else
            j--;
    }

    printf("Length of LCS = %d\n", C[m][n]);
    printf("LCS = %s\n", LCS);
}

int main() {
```

```
    char X[] = "AGCCCTAAGGGCTACCTAGCTT";
    char Y[] = "GACAGCCTACAAGCGTTAGCTTG";

    LCS(X, Y);
    return 0;
}
```

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.

Example:

AABCBDC

LRS= ABC or ABD

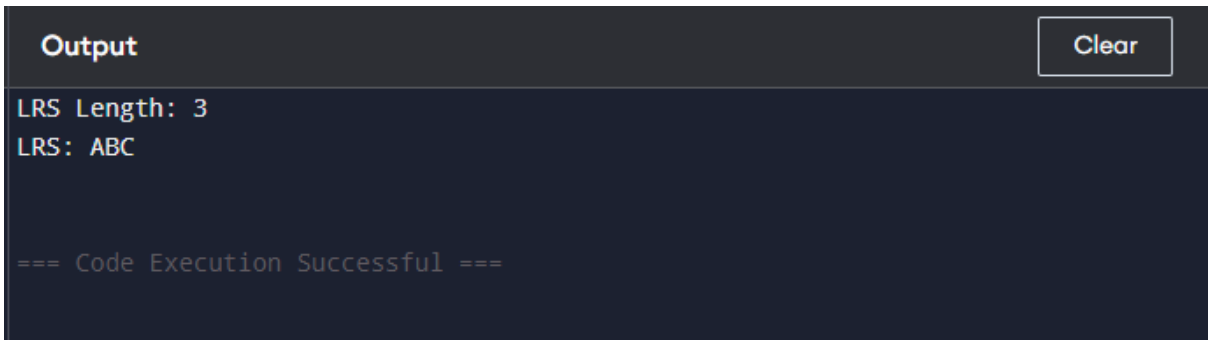
CODE:

```
#include <stdio.h>
#include <string.h>
```

```
int max(int a, int b) { return a > b ? a : b; }

void LRS(char *str) {
    int n = strlen(str);
    int dp[n+1][n+1];
    for (int i = 0; i <= n; i++)
        for (int j = 0; j <= n; j++)
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (str[i-1] == str[j-1] && i != j)
                dp[i][j] = dp[i-1][j-1] + 1;
            else
                dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
    int i = n, j = n, k = dp[n][n];
    char lrs[k+1];
    lrs[k] = '\0';

    while (i > 0 && j > 0) {
        if (str[i-1] == str[j-1] && i != j) {
            lrs[--k] = str[i-1];
            i--; j--;
        } else if (dp[i-1][j] > dp[i][j-1])
            i--;
        else
```

```
            j--;
        }

        printf("LRS Length: %d\n", dp[n][n]);
        printf("LRS: %s\n", lrs);
}

int main() {
        char S[] = "AABCBDC";
        LRS(S);
        return 0;
}
```

Output:



LEET CODE:

https://leetcode.com/problems/longest-common-subsequence/description/

CODE:

```
int longestCommonSubsequence(char *text1, char *text2) {
    int m = strlen(text1);
    int n = strlen(text2);
    int dp[m + 1][n + 1];

    for (int i = 0; i <= m; i++)
        for (int j = 0; j <= n; j++)
```

```
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (text1[i - 1] == text2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j -
1];

    return dp[m][n];
}
```

Output: