DAA PRACTICAL 5

NAME: DIVYA SONTAKKE

SECTION: A4\_B1 ROLL NO.: 11

**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.

**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.

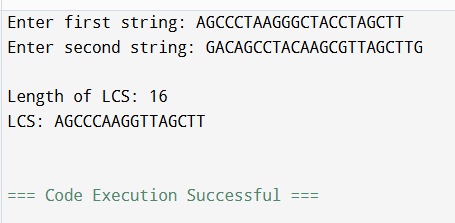
Length of LCS=16

**CODE**

#include <stdio.h>  
#include <string.h>  
#define MAX 100    
  
void printLCS(char b[MAX][MAX], char X[], int i, int j) {  
    if (i == 0 || j == 0)  
        return;  
    if (b[i][j] == 'D') {          
        printLCS(b, X, i - 1, j - 1);  
        printf("%c", X[i - 1]);  
    }  
    else if (b[i][j] == 'U') {      
        printLCS(b, X, i - 1, j);  
    }  
    else {                          
        printLCS(b, X, i, j - 1);  
    }  
}  
  
void lcsLength(char X[], char Y[], int m, int n) {  
    int c[MAX][MAX];                
    char b[MAX][MAX];

    for (int i = 0; i <= m; i++) c[i][0] = 0;  
    for (int j = 0; j <= n; j++) c[0][j] = 0;  
  
    for (int i = 1; i <= m; i++) {  
        for (int j = 1; j <= n; j++) {  
            if (X[i - 1] == Y[j - 1]) {  
                c[i][j] = c[i - 1][j - 1] + 1;  
                b[i][j] = 'D';  
            }  
            else if (c[i - 1][j] >= c[i][j - 1]) {  
                c[i][j] = c[i - 1][j];  
                b[i][j] = 'U';  
            }  
            else {  
                c[i][j] = c[i][j - 1];  
                b[i][j] = 'L';  
            }  
        }  
    }  
  
    printf("\nLength of LCS: %d\n", c[m][n]);  
    printf("LCS: ");  
    printLCS(b, X, m, n);  
    printf("\n");  
}  
  
int main() {  
    char X[MAX], Y[MAX];  
  
    printf("Enter first string: ");  
    scanf("%s", X);  
    printf("Enter second string: ");  
    scanf("%s", Y);  
    int m = strlen(X);  
    int n = strlen(Y);  
    lcsLength(X, Y, m, n);  
  
    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>  
#define MAX 100  
  
int main() {  
    char str[MAX];  
    int dp[MAX][MAX];  
    int i, j;  
    printf("Enter a string: ");  
    scanf("%s", str);  
    int n = strlen(str);  
  
    for (i = 0; i <= n; i++) {  
        for (j = 0; j <= n; j++) {  
            dp[i][j] = 0;  
        }  
    }  
  
    for (i = 1; i <= n; i++) {  
        for (j = 1; j <= n; j++) {  
            if (str[i - 1] == str[j - 1] && i != j)  
                dp[i][j] = 1 + dp[i - 1][j - 1];  
            else  
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];  
        }  
    }  
    printf("\nLength of LRS: %d\n", dp[n][n]);  
  
    char lrs[MAX];  
    int index = dp[n][n];  
    lrs[index] = '\0';  
  
    i = n;  
    j = n;  
    while (i > 0 && j > 0) {  
        if (dp[i][j] == dp[i - 1][j - 1] + 1 &&  
            str[i - 1] == str[j - 1] && i != j) {  
            lrs[index - 1] = str[i - 1];  
            i--;  
            j--;  
            index--;  
        }  
        else if (dp[i - 1][j] > dp[i][j - 1])  
            i--;  
        else  
            j--;  
    }  
    printf("LRS: %s\n", lrs);  
    return 0;  
}

**OUTPUT**

