**PRACTICAL - 5**

**Name - Sanskar Edhate**

**Section - A3\_B4**

**Roll No. - 57**

**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 -**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#define MAX 100**

**void FindLCS(char\* X, char\* Y, int m, int n, int dp[MAX][MAX], char direction[MAX][MAX]) {**

**for (int i = 0; i <= m; i++) {**

**dp[i][0] = 0;**

**direction[i][0] = ' ';**

**}**

**for (int j = 0; j <= n; j++) {**

**dp[0][j] = 0;**

**direction[0][j] = ' ';**

**}**

**for (int i = 1; i <= m; i++) {**

**for (int j = 1; j <= n; j++) {**

**if (X[i - 1] == Y[j - 1]) {**

**dp[i][j] = dp[i - 1][j - 1] + 1;**

**direction[i][j] = 'D'; // Diagonal**

**} else if (dp[i - 1][j] >= dp[i][j - 1]) {**

**dp[i][j] = dp[i - 1][j];**

**direction[i][j] = 'U'; // Up**

**} else {**

**dp[i][j] = dp[i][j - 1];**

**direction[i][j] = 'L'; // Left**

**}**

**}**

**}**

**}**

**void printLCS(char direction[MAX][MAX], char\* X, int i, int j) {**

**if (i == 0 || j == 0)**

**return;**

**if (direction[i][j] == 'D') {**

**printLCS(direction, X, i - 1, j - 1);**

**printf("%c", X[i - 1]);**

**} else if (direction[i][j] == 'U') {**

**printLCS(direction, X, i - 1, j);**

**} else {**

**printLCS(direction, X, i, j - 1);**

**}**

**}**

**int main() {**

**char X[] = "AGCCCTAAGGGCTACCTAGCTT";**

**char Y[] = "GACAGCCTACAAGCGTTAGCTTG";**

**int m = strlen(X);**

**int n = strlen(Y);**

**int dp[MAX][MAX];**

**char direction[MAX][MAX];**

**FindLCS(X, Y, m, n, dp, direction);**

**printf("\nCost Matrix :\n");**

**for (int i = 0; i <= m; i++) {**

**for (int j = 0; j <= n; j++) {**

**printf("%2d ", dp[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\nDirection Matrix:\n");**

**for (int i = 0; i <= m; i++) {**

**for (int j = 0; j <= n; j++) {**

**printf(" %c ", direction[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\nFinal cost of LCS: %d\n", dp[m][n]);**

**printf("\nLCS: ");**

**printLCS(direction, X, m, n);**

**printf("\n");**

**return 0;**

**}**

****

****

**TASK-2: Find the longest repeating subsequence (LRS). Consider it as a variation of the longest common subsequence (LCS) problem.**

**Code -**

**#include <stdio.h>**

**#include <string.h>**

**#define MAX 100**

**void computeLRS(char\* S, int n, int dp[MAX][MAX], char direction[MAX][MAX]) {**

**// Initialize dp and direction arrays**

**for (int i = 0; i <= n; i++) {**

**dp[i][0] = 0;**

**direction[i][0] = ' ';**

**}**

**for (int j = 0; j <= n; j++) {**

**dp[0][j] = 0;**

**direction[0][j] = ' ';**

**}**

**for (int i = 1; i <= n; i++) {**

**for (int j = 1; j <= n; j++) {**

**// Characters match and indices are not the same**

**if (S[i-1] == S[j-1] && i != j) {**

**dp[i][j] = dp[i-1][j-1] + 1;**

**direction[i][j] = 'D'; // Diagonal**

**} else if (dp[i-1][j] >= dp[i][j-1]) {**

**dp[i][j] = dp[i-1][j];**

**direction[i][j] = 'U'; // Up**

**} else {**

**dp[i][j] = dp[i][j-1];**

**direction[i][j] = 'L'; // Left**

**}**

**}**

**}**

**}**

**void printLRS(char direction[MAX][MAX], char\* S, int i, int j) {**

**if (i == 0 || j == 0)**

**return;**

**if (direction[i][j] == 'D') {**

**printLRS(direction, S, i-1, j-1);**

**printf("%c", S[i-1]);**

**} else if (direction[i][j] == 'U') {**

**printLRS(direction, S, i-1, j);**

**} else {**

**printLRS(direction, S, i, j-1);**

**}**

**}**

**int main() {**

**char S[] = "AABCBDC";**

**int n = strlen(S);**

**int dp[MAX][MAX];**

**char direction[MAX][MAX];**

**computeLRS(S, n, dp, direction);**

**printf("Length of Longest Repeating Subsequence: %d\n", dp[n][n]);**

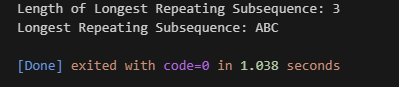
**printf("Longest Repeating Subsequence: ");**

**printLRS(direction, S, n, n);**

**printf("\n");**

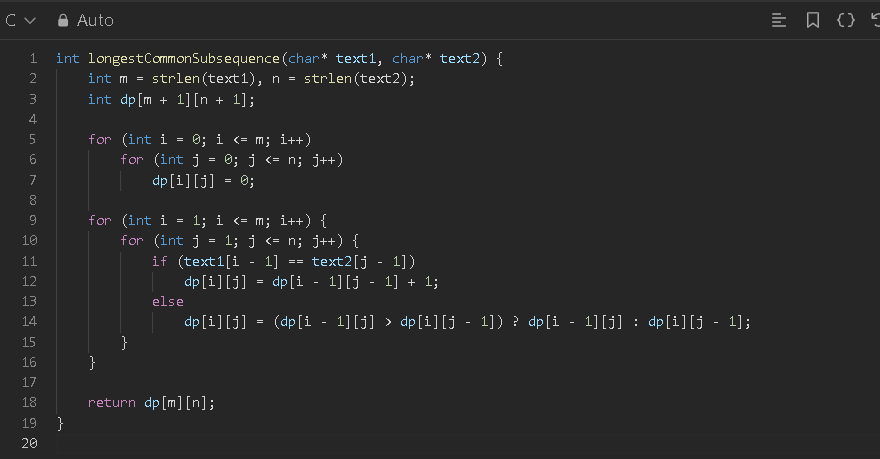
**return 0;**

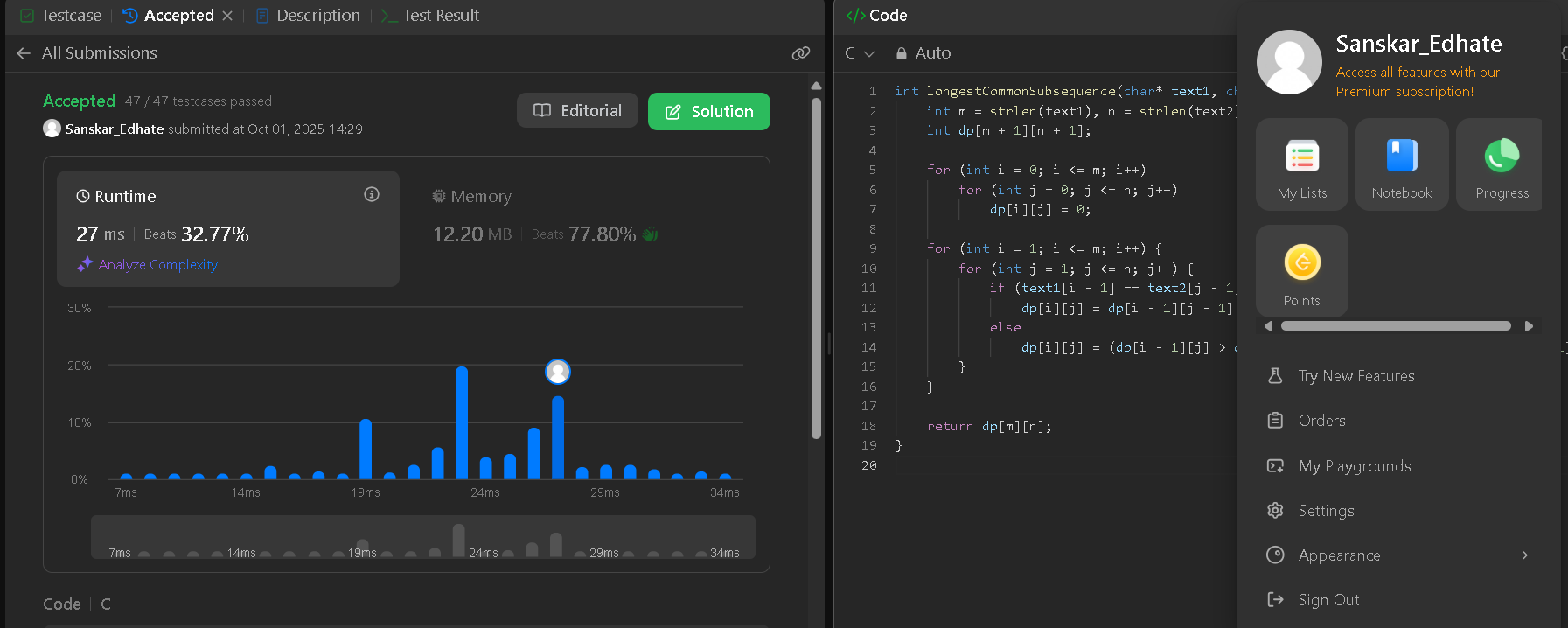
**}**

****

**LeetCode Assesment:**

**Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.**

****

****