RBU

RAMDEOBABA UNIVERSITY, NAGPUR

Formerly Shri Ramdeobaba College of Engineering & Management (RCOEM) Est. 1984

LEARN | INNOVATE | ACCOMPLISH

Name: Nachiket Jagdale

Section: A3

Batch: B1

Roll no. 02

Subject: Design and Analysis of Algorithms - Lab

Practical No. 5

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

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

LeetCode Assesment:

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

TASK 1:

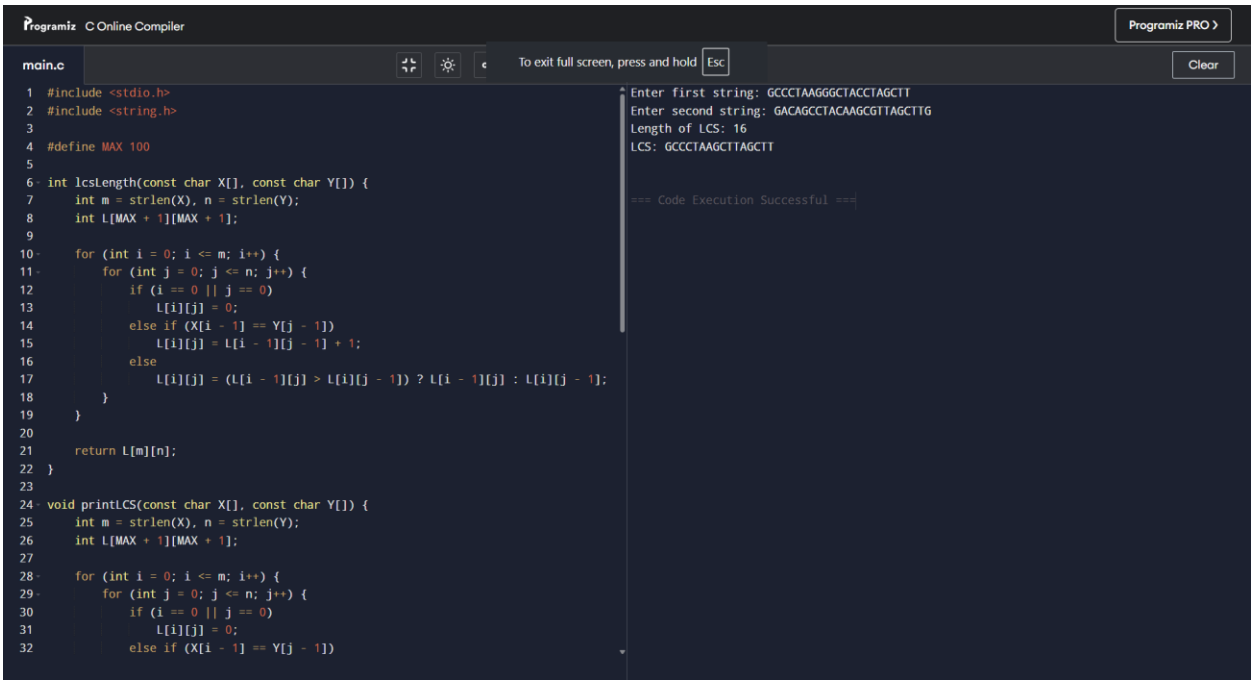
CODE :-

```
#include <stdio.h>

#include <string.h>

int lcsLength(char X[], char Y[])

{

int m = strlen(X), n = strlen(Y);

int L[100][100];

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

{

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

{

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

L[i][j] = 0;

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

L[i][j] = L[i-1][j-1] + 1;

else

L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];

}

}

return L[m][n];

}

void printLCS(char X[], char Y[])
```

```
{
int m = strlen(X), n = strlen(Y);
int L[100][100];
for (int i = 0; i <= m; i++)
{
for (int j = 0; j <= n; j++)
{
if (i == 0 || j == 0)
L[i][j] = 0;
else if (X[i-1] == Y[j-1])
L[i][j] = L[i-1][j-1] + 1;
else
L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
}
}
int index = L[m][n];
char lcs[index + 1];
lcs[index] = '\0';
int 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 (L[i-1][j] > L[i][j-1])
i--;
else
j--;
}
printf("LCS: %s\n", lcs);
}
int main()
{
char X[100], Y[100];
scanf("%s", X);
scanf("%s", Y);
printf("Length of LCS: %d\n", lcsLength(X, Y));
printLCS(X, Y);
return 0;
}
```

Output:-



TASK 2:

CODE :-

```
#include <stdio.h>

#include <string.h>

int lrsLength(char S[])

{

int m = strlen(S);

int L[100][100];

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

{

for (int j = 0; j <= m; j++)

{
```

```
if (i == 0 || j == 0)
{
L[i][j] = 0;
}
else if (S[i-1] == S[j-1] && i != j)
{
L[i][j] = L[i-1][j-1] + 1;
}
else
{
L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
}
}
}
return L[m][m];
}
void printLRS(char S[])
{
int m = strlen(S);
int L[100][100];
char lrs[100];
int index = 0;
for (int i = 0; i <= m; i++)
{
```

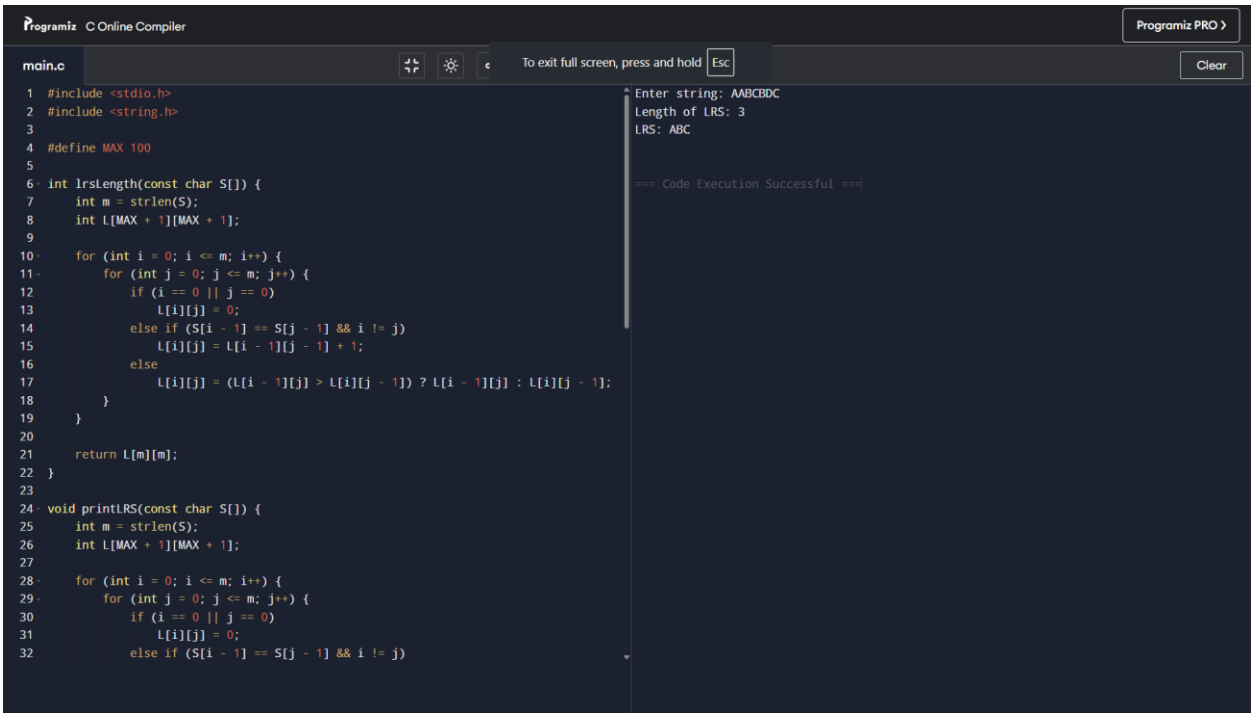
```
for (int j = 0; j <= m; j++)
{
if (i == 0 || j == 0)
{
L[i][j] = 0;
}
else if (S[i-1] == S[j-1] && i != j)
{
L[i][j] = L[i-1][j-1] + 1;
}
else
{
L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
}
}
}
int i = m, j = m;
while (i > 0 && j > 0)
{
if (S[i-1] == S[j-1] && i != j)
{
lrs[index++] = S[i-1];
i--;
j--;
```

```
}
else if (L[i-1][j] > L[i][j-1])
{
i--;
}
else
{
j--;
}
}
for (int k = index - 1; k >= 0; k--)
{
printf("%c", lrs[k]);
}
printf("\n");
}
int main()
{
char S[100];
printf("Enter string: ");
scanf("%s", S);
printf("Length of LRS: %d\n", lrsLength(S));
printf("LRS: ");
printLRS(S);
```

return 0;

}

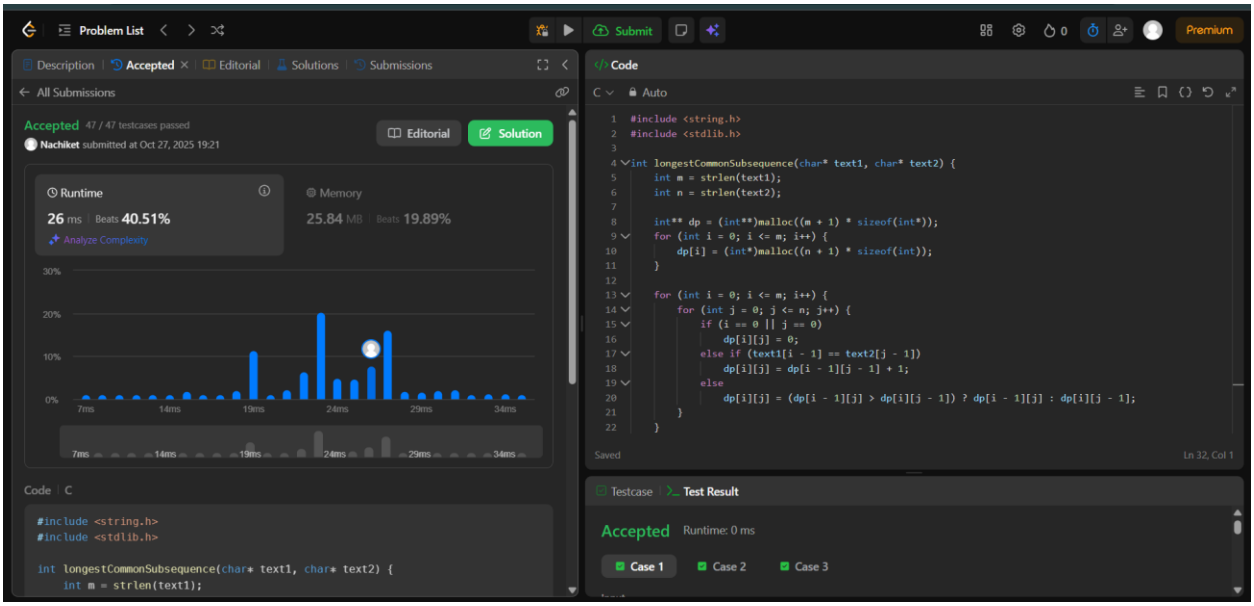
Output:-



LeetCode Assesment:-

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

Leetcode submission :-



My leetcode link :-

https://leetcode.com/u/codd12/