G. H. RAISONI COLLEGE OF ENGG., NAGPUR
(An Autonomous Institute under UGC Act 1956)
Department of Computer Science & Engg.

Date: 04/08/2020

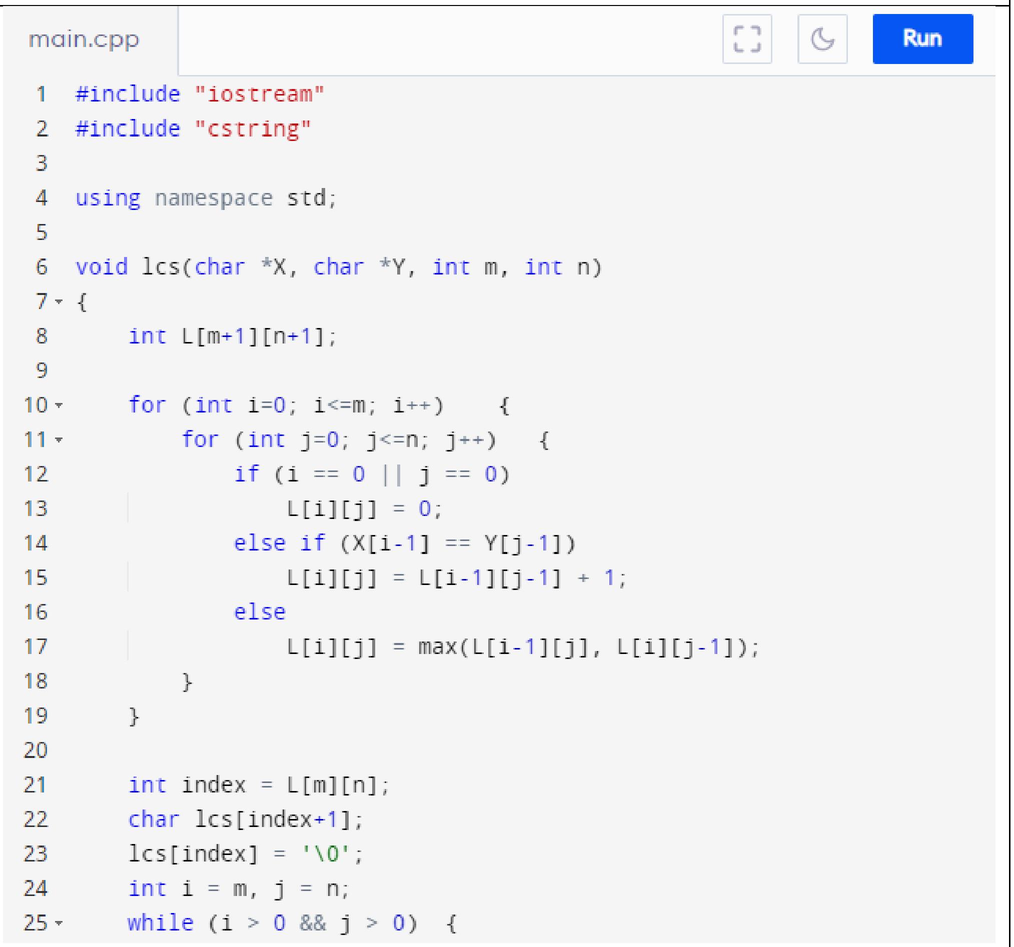
Practical Subject: Design and Analysis of Algorithms
Session: 2020-21

Student Details:

Roll Number	58
Name	Shivam Tawari
Semester	3
Section	A
Branch	Artificial Intelligence

Practical Details: Practical Number- 5

Practical Aim	To Implement and Analyze time complexity of Algorithm of Dynamic Programming Technique.
Theory & Algorithm	***Dynamic Programming:*** Dynamic Programming (DP) is an algorithmic technique for solving an optimization problem by breaking it down into simpler subproblems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its subproblems. ***Longest Common Subsequence:*** The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). It differs from the longest common substring problem: unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences.

	LCS Algorithm: Step 1: START Step 2: Input String X and Y Step 3: Construct a matrix L with (m+1, n+1) dimensions and i=0 Step 4: Check if i <= m, if true then set j = 0 else goto Step 8 Step 5: if either i=0 or j=0, set L[i][j] = 0 else if X[i-1] = Y[j-1], then set L[i][j] = L[i-1][j-1] + 1 else set L[i][j] with max of left and upper cell Step 6: Increment j Step 7: If j <= n, go to Step 5 else goto Step 4 with i += 1 Step 8: Print Maximum possible Subsequence Length and LCS Step 9: STOP
Complexity	***Worst Case:*** O (mn) ***Average Case:*** θ (mn) ***Best Case:*** Ω (mn)
Program	

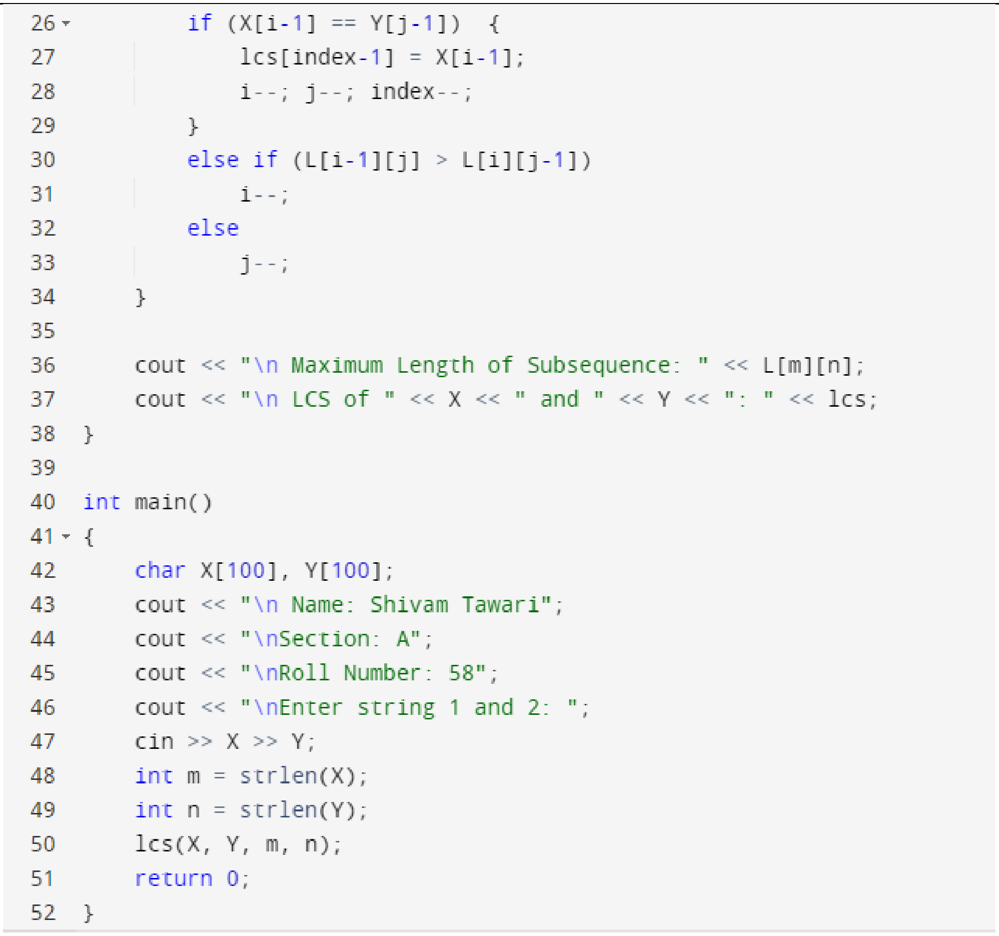
```cpp
#include "iostream"
#include "cstring"

using namespace std;

void lcs(char *X, char *Y, int m, int n)
{
    int L[m+1][n+1];

    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] = max(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)  {
```

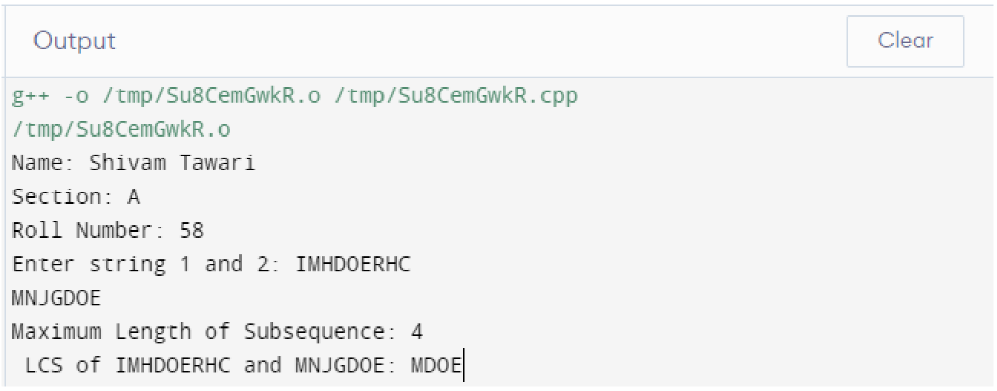


```cpp
        if (X[i-1] == Y[j-1])  {
            lcs[index-1] = X[i-1];
            i--; j--; index--;
        }
        else if (L[i-1][j] > L[i][j-1])
            i--;
        else
            j--;
    }

    cout << "\n Maximum Length of Subsequence: " << L[m][n];
    cout << "\n LCS of " << X << " and " << Y << ": " << lcs;
}

int main()
{
    char X[100], Y[100];
    cout << "\n Name: Shivam Tawari";
    cout << "\nSection: A";
    cout << "\nRoll Number: 58";
    cout << "\nEnter string 1 and 2: ";
    cin >> X >> Y;
    int m = strlen(X);
    int n = strlen(Y);
    lcs(X, Y, m, n);
    return 0;
}
```

Output



```
g++ -o /tmp/Su8CemGwkR.o /tmp/Su8CemGwkR.cpp
/tmp/Su8CemGwkR.o
Name: Shivam Tawari
Section: A
Roll Number: 58
Enter string 1 and 2: IMHDOERHC
MNJGDOE
Maximum Length of Subsequence: 4
 LCS of IMHDOERHC and MNJGDOE: MDOE
```