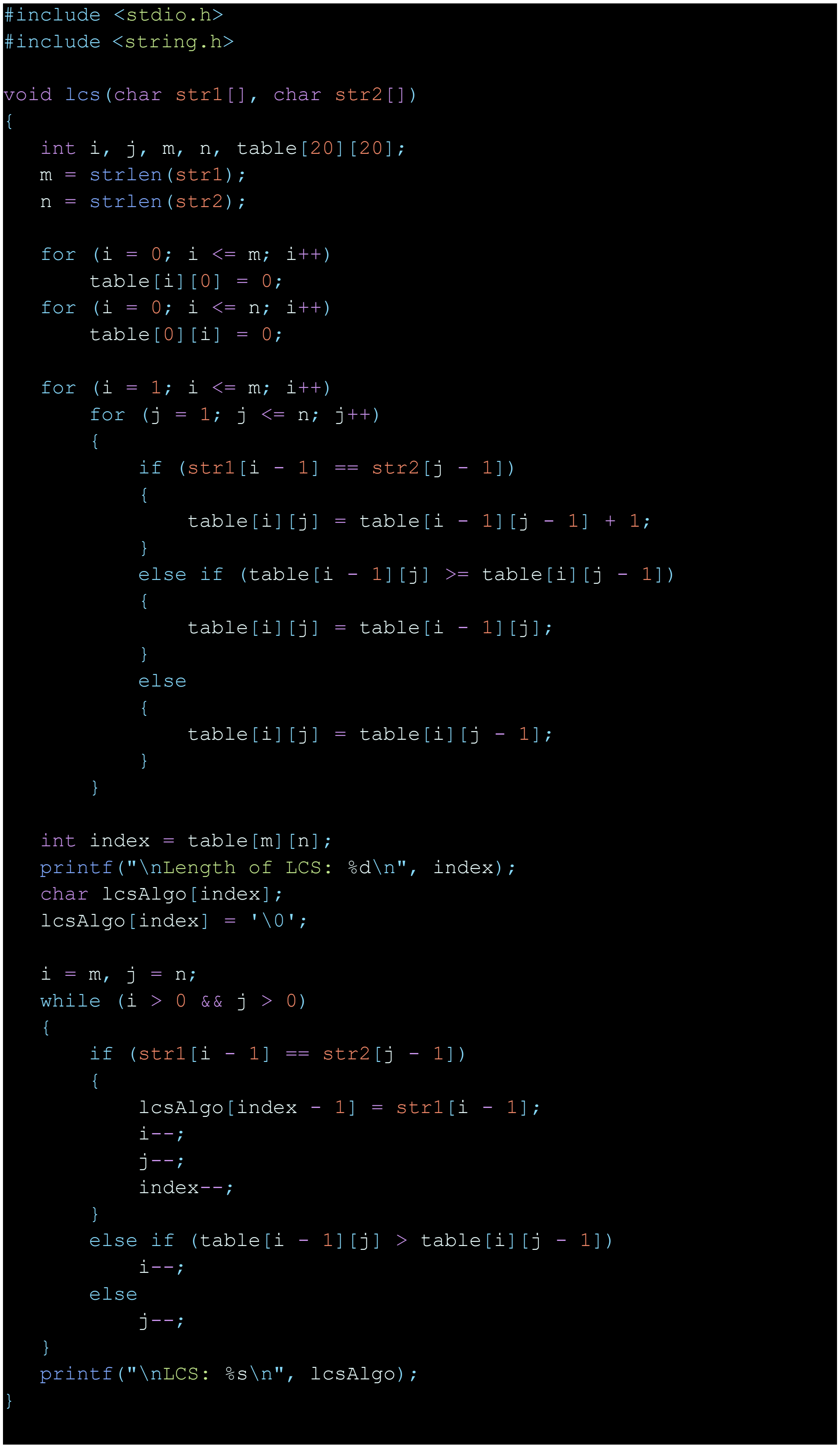
| | |
|---|---|
| **Name** | DARSHIT BHAGTANI |
| **UID** | 2021700006 |
| **Experiment No.** | 4 |

| | |
|---|---|
| **AIM:** | Dynamic Programming - Longest Common Subsequence |
| **PROBLEM STATEMENT :** | Apply the concept of dynamic programming to solve the problem of finding Longest Common Subsequence |
| **ALGORITHM/ THEORY:** | **Algorithm:-**<br>1. Define two sequences X and Y, with lengths m and n, respectively.<br>2. Create a matrix of size (m) x (n) and initialize all entries to zero.<br>3. For i = 1 to m, and for j = 1 to n:<br>a. If the i-th character of X is equal to the j-th character of Y, set the i,j entry in the matrix to be the value of the i-1,j-1 entry plus one.<br>b. Otherwise, set the i,j entry in the matrix to be the maximum of the i-1,j and i,j-1 entries.<br>4. The value of the i,j entry in the matrix represents the length of the longest common subsequence of the first i characters of X and the first j characters of Y.<br>5. Trace back the matrix starting from the bottom right corner and find the longest common subsequence.<br><br>The four steps of dynamic programming are:<br>1. Define the problem and identify subproblems.<br>2. Formulate a recursive relationship between the subproblems.<br>3. Create a memoization table to store solutions to each subproblem.<br>4. Solve subproblems in a specific order to obtain the solution to the original problem.<br><br>Some important applications of LCS include:<br>1. DNA Sequencing<br>2. Text Comparison<br>3. Speech Recognition<br>4. Image Recognition |

**PROGRAM:**



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

void lcs(char str1[], char str2[])
{
    int i, j, m, n, table[20][20];
    m = strlen(str1);
    n = strlen(str2);

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

    for (i = 1; i <= m; i++)
        for (j = 1; j <= n; j++)
        {
            if (str1[i - 1] == str2[j - 1])
            {
                table[i][j] = table[i - 1][j - 1] + 1;
            }
            else if (table[i - 1][j] >= table[i][j - 1])
            {
                table[i][j] = table[i - 1][j];
            }
            else
            {
                table[i][j] = table[i][j - 1];
            }
        }

    int index = table[m][n];
    printf("\nLength of LCS: %d\n", index);
    char lcsAlgo[index];
    lcsAlgo[index] = '\0';

    i = m, j = n;
    while (i > 0 && j > 0)
    {
        if (str1[i - 1] == str2[j - 1])
        {
            lcsAlgo[index - 1] = str1[i - 1];
            i--;
            j--;
            index--;
        }
        else if (table[i - 1][j] > table[i][j - 1])
            i--;
        else
            j--;
    }
    printf("\nLCS: %s\n", lcsAlgo);
}
```

```
int main()
{
   char str1[20], str2[20];
   printf("String 1: ");
   scanf("%s", str1);
   printf("String 2: ");
   scanf("%s", str2);
   lcs(str1, str2);
}
```

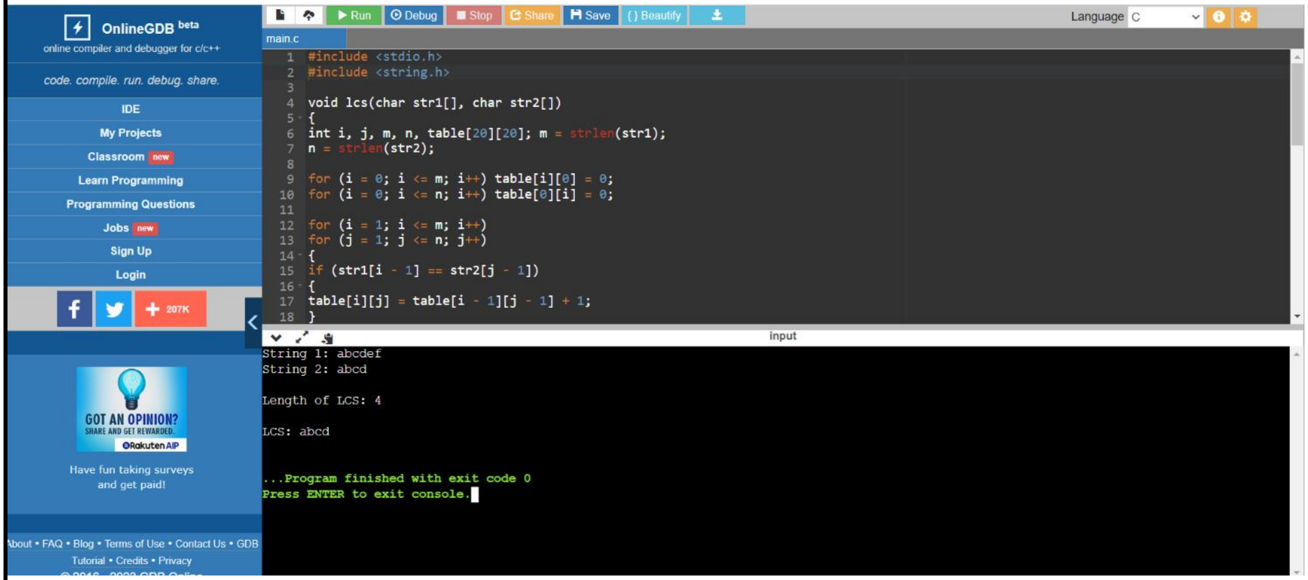
**RESULT:**

Using Dynamic Programming,
Time complexity: O(mn)

Hirschberg's algorithm uses divide-and-conquer technique
Time complexity: O(mn log n)

Using suffix trees or suffix arrays
Time complexity: O(m + n)

where m and n are the lengths of the two input sequences



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

void lcs(char str1[], char str2[])
{
int i, j, m, n, table[20][20]; m = strlen(str1);
n = strlen(str2);

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

for (i = 1; i <= m; i++)
for (j = 1; j <= n; j++)
{
if (str1[i - 1] == str2[j - 1])
{
table[i][j] = table[i - 1][j - 1] + 1;
}
```

```
String 1: abcdef
String 2: abcd

Length of LCS: 4

LCS: abcd


...Program finished with exit code 0
Press ENTER to exit console.
```

| **CONCLUSION:** | Successfully understood the concept of dynamic programming and solved the problem of finding out the longest common subsequence for two strings. |
|---|---|