Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Analysis of Algorithm Laboratory
Course Code:	DJ19CEL404
Experiment No.:	06

AIM: **TO IMPLEMENT LONGEST COMMON SUBSEQUENCE PROBLEM.**

THEORY:

LCS

- The longest common subsequence (LCS) is defined as the longest subsequence that is
- common to all the given sequences, provided that the elements of the subsequence are
- not required to occupy consecutive positions within the original sequences.
- If S1 and S2 are the two given sequences then, Z is the common subsequence
- of S1 and S2 if Z is a subsequence of both S1 and S2. Furthermore, Z must be a strictly
- increasing sequence of the indices of both S1 and S2.
- In a strictly increasing sequence, the indices of the elements chosen from the original
- sequences must be in ascending order in Z.
- **Pseudocode:**

```
X and Y be two given sequences
Initialize a table LCS of dimension X.length * Y.length
X.label = X
Y.label = Y
LCS[0][] = 0
LCS[][0] = 0
Start from LCS[1][1]
Compare X[i] and Y[j]
If X[i] = Y[j]
    LCS[i][j] = 1 + LCS[i-1, j-1]
    Point an arrow to LCS[i][j]
Else
    LCS[i][j] = max(LCS[i-1][j], LCS[i][j-1])
    Point an arrow to max(LCS[i-1][j], LCS[i][j-1])
```

CODE:

```
#include <stdio.h>
#include <string.h>
int i, j, m, n, LCS_table[20][20];
char S1[20] = "ACADB", S2[20] = "CBDA", b[20][20];
void lcsAlgo()
{
    m = strlen(S1);
```

```
    n = strlen(S2);
    // Filling 0's in the matrix
    for (i = 0; i <= m; i++)
        LCS_table[i][0] = 0;
    for (i = 0; i <= n; i++)
        LCS_table[0][i] = 0;
    // Building the mtrix in bottom-up way
    for (i = 1; i <= m; i++)
        for (j = 1; j <= n; j++)
        {
            if (S1[i - 1] == S2[j - 1])
            {
                LCS_table[i][j] = LCS_table[i - 1][j - 1] + 1;
            }
            else if (LCS_table[i - 1][j] >= LCS_table[i][j - 1])
            {
                LCS_table[i][j] = LCS_table[i - 1][j];
            }
            else
            {
                LCS_table[i][j] = LCS_table[i][j - 1];
            }
        }
    int index = LCS_table[m][n];
    char lcsAlgo[index + 1];
    lcsAlgo[index] = '\0';
    int i = m, j = n;
    while (i > 0 && j > 0)
    {
        if (S1[i - 1] == S2[j - 1])
        {
            lcsAlgo[index - 1] = S1[i - 1];
            i--;
            j--;
            index--;
        }
        else if (LCS_table[i - 1][j] > LCS_table[i][j - 1])
            i--;
        else
            j--;
    }
    printf("S1 : %s \nS2 : %s \n", S1, S2);
    printf("LCS: %s", lcsAlgo);
}
```

```
int main()
{
    lcsAlgo();
    printf("\n");
}
```

OUTPUT:

```
exe' '--interpreter=mi'
S1 : ACADB
S2 : CBDA
LCS: CB
PS C:\Users\Jadhav\Desktop\BTech\4th sem\AOA\Prac\Code>
```

CONCLUSION:

- Thus, we implemented Longest Common Subsequence and found the longest common subsequence in 2 strings.