Name	Sujal Chordia
UID no.	2021700015
Experiment No.	4

AIM:	Implement Dynamic Programming -Longest Common Subsequence.
Program	
PROBLEM STATEMENT:	Implement Longest Common Subsequence code using Dynamic Programming.

ALGORITHM/ THEORY:	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 X and Y are the two given sequences then, Z is the common subsequence of X and Y if Z is a subsequence of both X and Y. Furthermore, Z must be a **strictly increasing sequence** of the indices of both X and Y.
	
	Example 1: if
	`X = {B, C, D, A, A, C, D}`
	
	Then, {A, D, B} cannot be a subsequence of X as the order of the elements is not the same (i.e. not strictly increasing sequence).
	
	Example 2: if
	`X = {B, C, D, A, A, C, D}`
	`Y = {A, C, D, B, A, C}`
	
	Then, the common subsequences are `{B, C}`, `{C, D, A, C}`, `{D, A, C}`, `{A, A, C}`, `{A, C}`, `{C, D}`, `...`
	Among these subsequences, `{C, D, A, C}` is the longest common subsequence. We are going to find this longest common subsequence using dynamic programming.

	Dynamic Programming Steps: 1] Optimal substructure of an LCS: Let X = <x1, x2,…..,xm> and Y = <y1,y2,……,yn> be sequences, and let Z = <z1,z2,……,zk> be any LCS of X and Y . 1. If xm = yn, then zk = xm = yn and Zk-1 is an LCS of Xm-1 and Yn-1. 2. If xm ≠ yn, then zk ≠ xm implies that Z is an LCS of Xm-1 and Y or zk ≠ yn implies that Z is an LCS of X and Yn-1. Thus, the longest common subsequence tells us that an LCS of two sequences contains within it an LCS of prefixes of the two sequences. Thus, the LCS problem has an optimal-substructure property. 2] A recursive solution: c[i,j] = 0 if i = 0 or j = 0 = c[i-1,j-1] +1 if i, j>0 and xi = yj = max(c[i,j-1],c[i-1,j]) if i, j>0 and xi ≠ yj 3] Computing the length of an LCS: Memorize the solutions in a matrix/table . 4] Constructing an LCS: Backtrace the table to find the LCS. The method of dynamic programming reduces the number of function calls. It stores the result of each function call so that it can be used in future calls without the need for redundant calls. In the above dynamic algorithm, the results obtained from each comparison between elements of X and the elements of Y are stored in a table so that they can be used in future computations. So, the time taken by a dynamic approach is the time taken to fill the table (i.e. O(m*n)). Whereas, the recursion algorithm has the complexity of $2^{max(m, n)}$.
PROGRAM:	#include<stdio.h> #include<string.h> int lcs[100][100]; int max(int x, int y){

```
   if(x>y)
     return x;
   else
     return y;

}

int LCS(char a[],char b[],int la,int lb){

    if(la==0 || lb==0){
      lcs[la][lb]=0;
      return 0;
    }
    if(a[la-1]==b[lb-1]){
      lcs[la][lb] = 1 + LCS(a,b,la-1,lb-1);
      return 1 + LCS(a,b,la-1,lb-1);
    }
    else{
      lcs[la][lb] = max(LCS(a,b,la-1,lb),LCS(a,b,la,lb-1));
      return lcs[la][lb];
    }

}

int main(){

   char a[100],b[100];
   printf("\nEnter string 1: ");
   scanf("%s",a);
   printf("\nEnter string 2: ");
   scanf("%s",b);
   int t = LCS(a,b,strlen(a),strlen(b));
   int i = strlen(a),j = strlen(b);
   char c[t+1];
   c[t] = '\0';
   while(i>0 && j>0){
     if(a[i-1]==b[j-1]){
       c[t-1] = a[i-1];
       i--;
       j--;
       t--;
     }
     else if(lcs[i-1][j]>lcs[i][j-1])
       i--;
     else
       j--;
   }
```

```
    printf("\nThe Longest Common Subsequence is %s",c);

}
```

RESULT:

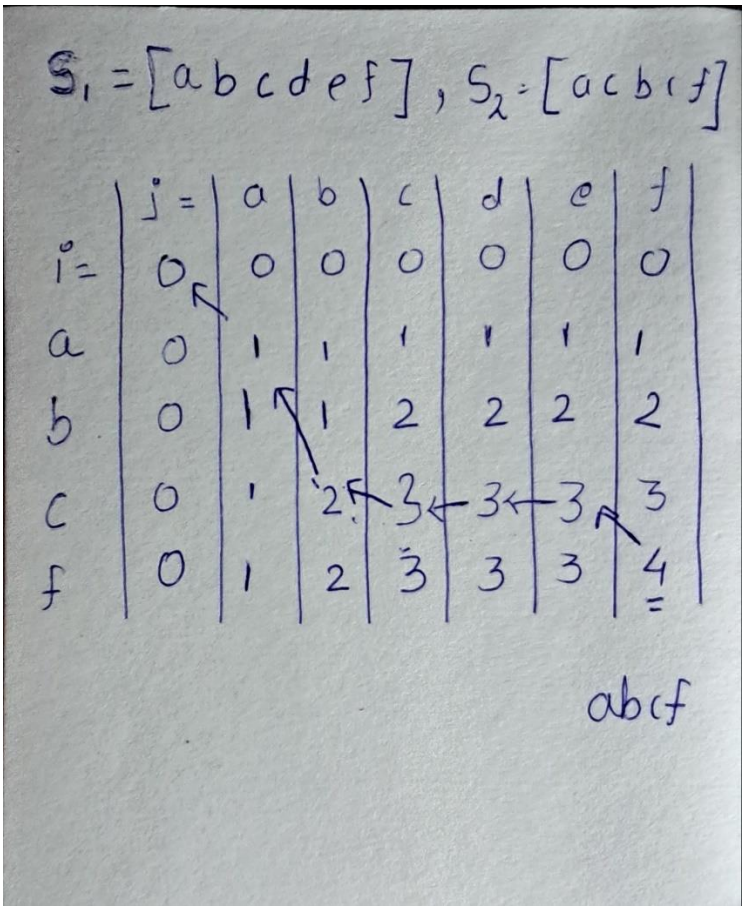
```
Enter string 1: abcdef

Enter string 2: acbcf

The Longest Common Subsequence is abcf

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

WORKING:



CONCLUSION:	We used dynamic programming approach to find the Longest Common Subsequence between to strings. We also analyzed what are the steps of dynamic programming. We saw how Dynamic programming method to solve LCS is better than Recursive method to solve LCS.