**Experiment No: 7**

**TEXT SIMILARITY**

**Aim:** To study and implement text similarity testing.

**THEORY:**

Text Similarity:

A similarity algorithm indicating the length of the longest common subsequence between two strings.

The Longest common subsequence algorithm returns the length of the longest subsequence that two strings have in common. Two strings that are entirely different, return a value of 0, and two strings that return a value of the commonly shared length implies that the strings are completely the same in value and position.

**Example:**

S1 = {B, C, D, A, A, C, D}

S2 = {A, C, D, B, A, C}

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 with length=4

**Algorithm:**

Algorithm longestCommonSubsequece(str1, str2, len1, len2)

{

//create a matrix LCS of order (len1+1)\*(len2+1) to tabulate values

//initializing

for(i=0;i<=len1;i++)

LCS[i][0]=0; //empty str2

for(j=0;j<=len2;j++)

LCS[0][j]=0; //empty str1

//now, start filling the matrix row wise

for(i=1;i<=len1;i++)

{

for(j=1;j<=len2;j++)

{

//if current character of both strings match

if(str1[i-1]==str2[j-1])

{

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

}

//mismatch

else

{

LCS[i][j]=max(LCS[i-1][j],LCS[i][j-1]);

}

}

}

//return the final value in LCS[len1][len2];

}

**Program and Output:**

**Conclusion:** Text similarity testing algorithm was studied and implemented successfully.