CSE373 Assignment-3
Determination of longest common subsequence (LCS) problem

// Algorithm of LCS

// LCS

Let the input sequences be s1[0..m-1] and s2[0..n-1] of lengths m and n respectively. And let lcsTable(s1[0..m-1], s2[0..n-1]) be the length of LCS of the two sequences s1 and s2. Following is the recursive definition of lcsTable (s1[0..m-1], s2[0..n-1]).

If last characters of both sequences match (or s1[m-1] == s2[n-1]) then
lcsTable (s1[0..m-1], s2[0..n-1]) = 1 + lcsTable (s1[0..m-1], s2[0..n-1])

If last characters of both sequences do not match (or X[m-1] != Y[n-1]) then
lcsTable (s1[0..m-1], s2[0..n-1]) = maxValue(lcsTable (s1[0..m-2], s2[0..n-1]), lcsTable(s1[0..m-1], s2[0..n-2]))

// Print LCS and Length

Following is detailed algorithm to print the LCS. It uses the same 2D table L[][].

1) Construct lcsTable [m+1][n+1] using the steps discussed in previous steps.

2) The value lcsTable[m][n] contains length of LCS. Create a character array lcsStr[] of length equal to the length of lcsIndex plus 1 and (one extra to store \0).

2) Traverse the 2D array starting from L[m][n]. Do following for every cell lcsTable[p][q]

a) If characters (in s1 and s2) corresponding to lcsTable[p][q] are same (Or s1[i-1] == s2[j-1]), then include this character as part of LCS.

b) Else compare values of lcsTable[p-1][q] and lcsTable[p][q-1] and go in direction of greater value.

// Code of LCS

```
// CSE373  Assignment-3
// Longest Common Subsequence algorithm


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

int maxValue(int n1, int n2){
        int result;
        result = (n1 > n2) ? n1 : n2;
```

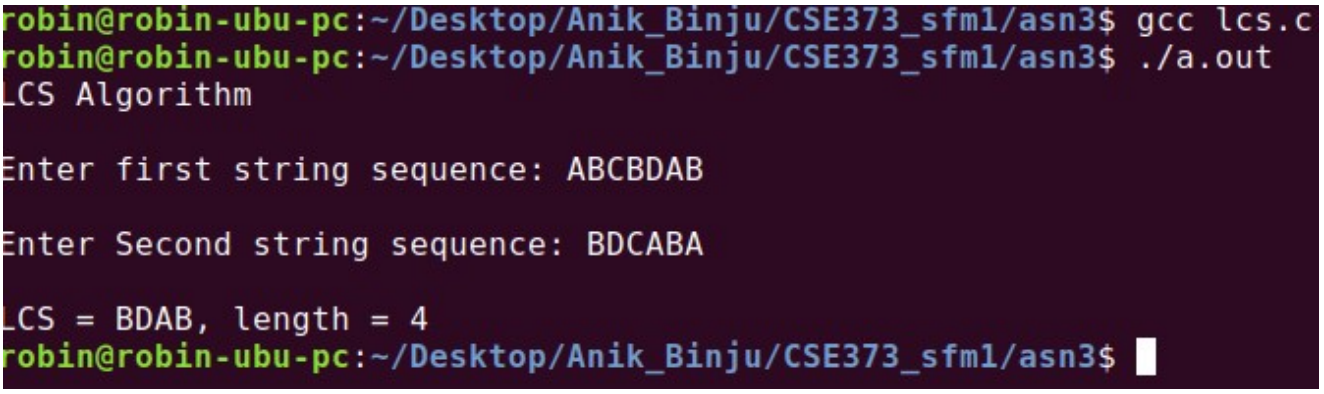
```
	return result;
}

void lcs(char *s1, char *s2, int m, int n){
	// define lcs table
	int lcsTable[m + 1][n + 1];

/*
	// define tracer table;
	char tracerTable[m + 1][n + 1];
*/

	// index variable to traverse row and column of lcs table
	int i, j;

/*
	// base case - if one of the subsequence element is 0
	// then set the value of that element 0
	for(i = 0; i <= m; i++) {
		lcsTable[i][0] = 0;
	}
	for(j = 0; j <= n; j++) {
		lcsTable[0][0] = 0;
	}
*/

	// fill the lcs table
	// c means cross, l means left, u means upper
	// elements in lcs table
	for(i = 0; i <= m; i++){
		for(j = 0; j <= n; j++){
			if(i == 0 || j == 0){
				lcsTable[i][j] = 0;
			}
			else if(s1[i - 1] == s2[j - 1]){
				lcsTable[i][j] = lcsTable[i - 1][j - 1] + 1;
				//tracerTable[i][j] = 'c';
			}
			/*
			else if(lcsTable[i - 1][j] >= lcsTable[i][j - 1]){
				lcsTable[i][j] = lcsTable[i - 1][j];
				tracerTable[i][j] = 'u';
			}
			else if(lcsTable[i - 1][j] <= lcsTable[i][j - 1]){
				lcsTable[i][j] = lcsTable[i][j - 1];
				tracerTable[i][j] = 'l';
			}
			*/
```

```
                else{
                        lcsTable[i][j] = maxValue(lcsTable[i - 1][j], lcsTable[i][j - 1]);
                }
        }
    }



    // Index of LCS length
    int lcsIndex = lcsTable[m][n];

    // create a char array to store lcs string
    char lcsStr[lcsIndex + 1];
    lcsStr[lcsIndex] = '\0';

    // here, p = i, q = j
    int p = m, q = n;
    while(p > 0 && q > 0){
        if(s1[p - 1] == s2[q - 1]){
            lcsStr[lcsIndex - 1] = s1[p - 1];
            p--;
            q--;
            lcsIndex--;
        }
        else if(lcsTable[p - 1][j] > lcsTable[i][q - 1]){
            p--;
        }
        else{
            q--;
        }
    }

    // store the lcs length to print
    int lcsLength = strlen(lcsStr);
    //printf("\nLCS Length: %d\n", lcsLength);


    // print LCS
    printf("\nLCS = %s, length = %d\n", lcsStr, lcsLength);

}

int main()
{
    printf("LCS Algorithm\n");

    // define variables for string sequence input
    char str1[15], str2[15];
```

```
	// take 2 string sequence input
	printf("\nEnter first string sequence: ");
	scanf("%s", str1);
	printf("\nEnter Second string sequence: ");
	scanf("%s", str2);

	// find the length of str1 and str2
	int str1Length, str2Length;

	str1Length = strlen(str1);
	str2Length = strlen(str2);

	// pass the str1 and str2 into lsc() function
	// to find out the Longest Common Subsequence
	lcs(str1, str2, str1Length, str2Length);



	return 0;
}
```

// Screenshot of Code

// ss1



// ss2

```
robin@robin-ubu-pc:~/Desktop/Anik_Binju/CSE373_sfm1/asn3$ gcc lcs.c
robin@robin-ubu-pc:~/Desktop/Anik_Binju/CSE373_sfm1/asn3$ ./a.out
LCS Algorithm

Enter first string sequence: ABC

Enter Second string sequence: AB

LCS = 0, length = 2
robin@robin-ubu-pc:~/Desktop/Anik_Binju/CSE373_sfm1/asn3$
```

// **ss3**

```
File  Edit  View  Search  Terminal  Help
robin@robin-ubu-pc:~/Desktop/Anik_Binju/CSE373_sfm1/asn3$ gcc lcs.c
robin@robin-ubu-pc:~/Desktop/Anik_Binju/CSE373_sfm1/asn3$ ./a.out
LCS Algorithm

Enter first string sequence: AGGTAB

Enter Second string sequence: GXTXAYB

LCS = GTAB, length = 4
robin@robin-ubu-pc:~/Desktop/Anik_Binju/CSE373_sfm1/asn3$
```

// **Time complexity**

Time Complexity of the above implementation is O(mn) .