**S1 = "kind"** and **S2 = "ending"** in all four requested ways:

**1️⃣ By Matrix (Dynamic Programming Table)**

We will build a matrix dp where:

* Rows = characters of kind (plus empty prefix)
* Columns = characters of ending (plus empty prefix)
* dp[i][j] = LCS length of S1[0..i-1] and S2[0..j-1].

|  | **""** | **e** | **n** | **d** | **i** | **n** | **g** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| "" | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| k | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| i | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| n | 0 | 0 | 1 | 1 | 1 | 2 | 2 |
| d | 0 | 0 | 1 | 2 | 2 | 2 | 2 |

✅ **LCS length** = dp[4][6] = 2  
✅ **LCS** = "nd"

**2️⃣ By Algorithm (Step Explanation)**

**Algorithm**:

1. Initialize dp table with size (m+1) x (n+1) where m=len(S1) and n=len(S2).
2. For each i in 1..m and j in 1..n:
   * If S1[i-1] == S2[j-1] → dp[i][j] = dp[i-1][j-1] + 1
   * Else → dp[i][j] = max(dp[i-1][j], dp[i][j-1])
3. LCS length is in dp[m][n].
4. Trace back from dp[m][n] to find the sequence.

**Result for “kind” & “ending”**:

* Matches at n and d
* LCS = "nd"
* Length = 2

**3. C++ Code**

#include<iostream>

#include<string>

#include<vector>

using namespace std;

int main() {

    string S1;

    string S2;

    cout << "Enter first string: ";

    getline(cin, S1);

    cout << "Enter second string: ";

    getline(cin, S2);

    int m = S1.length();

    int n = S2.length();

    vector<vector<int>> dp(m+1, vector<int>(n+1, 0));

    for (int i=1;i<=m;i++){

        for (int j=1;j <=n;j++) {

            if (S1[i-1]==S2[j-1]) dp[i][j]= dp[i-1][j-1] +1;

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

        }

    }

    cout << "LCS Length: " << dp[m][n] << endl;

    int index = dp[m][n];

    string lcs(index, ' ');

    int i =m, j= n;

    while (i >0 && j>0){

        if (S1[i-1]== S2[j-1]){

            lcs[index-1] =S1[i-1];

            i--; j--; index--;

        }

        else if (dp[i-1][j]>dp[i][j-1]) i--;

        else j--;

    }

    cout << "LCS: " << lcs << endl;

    return 0;

}

