**Shortest Common Supersequence**

**Medium**Accuracy: 54.34% Submissions: 19295 Points: 4

Given two strings X and Y of lengths m and n respectively, find the length of the smallest string which has both, X and Y as its sub-sequences.  
Note: X and Y can have both uppercase and lowercase letters.

**Example 1**

**Input:**

X = abcd, Y = xycd

**Output:** 6

**Explanation:** Shortest Common Supersequence

would be abxycd which is of length 6 and

has both the strings as its subsequences.

**Example 2**

**Input:**

X = efgh, Y = jghi

**Output:** 6

**Explanation:** Shortest Common Supersequence

would be ejfghi which is of length 6 and

has both the strings as its subsequences.

**Your Task:**  
Complete **shortestCommonSupersequence()** function that takes X, Y, m, and n as arguments and returns the length of the required string.

**Expected Time Complexity:**O(Length(X) \* Length(Y)).  
**Expected Auxiliary Space:**O(Length(X) \* Length(Y)).

**Constraints:**  
1<= |X|, |Y| <= 100

class Solution

{

    public:

    //Function to find length of shortest common supersequence of two strings.

    /\*int dp[105][107];

    int CommonSupersequence(int i, int j, string s1, string s2) {

        if (i<0 or j<0) return max(i+1, j+1);

        if (dp[i][j]!=-1) return dp[i][j];

        if (s1[i]==s2[j]) return dp[i][j]=1+CommonSupersequence(i-1, j-1, s1, s2);

        else return dp[i][j]=min(1+CommonSupersequence(i, j-1, s1, s2),

        1+CommonSupersequence(i-1, j, s1, s2));

    }\*/

    int shortestCommonSupersequence(string X, string Y, int m, int n) {

        //code here

        /\*memset(dp, -1, sizeof(dp));

        return CommonSupersequence(m-1, n-1, X, Y);\*/

        int dp[m+1][n+1];

        memset(dp, 0, sizeof(dp));

        for (int i=0; i<n+1; i++) dp[0][i]=i;   // Tabulation DP

        for (int i=1; i<m+1; i++) dp[i][0]=i;

        for (int i=1; i<m+1; i++) {

            for (int j=1; j<n+1; j++) {

                if (X[i-1]==Y[j-1]) dp[i][j]=1+dp[i-1][j-1];

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

            }

        }

        return dp[m][n];

    }

};