**Minimum number of deletions and insertions.**

**Easy**Accuracy: 59.67% Submissions: 13123 Points: 2

Given two strings **str1** and **str2**. The task is to remove or insert the minimum number of characters from/in **str1** so as to transform it into **str2**. It could be possible that the same character needs to be removed/deleted from one point of str1 and inserted to some another point.

**Example 1:**

**Input:** str1 = "heap", str2 = "pea"

**Output:** 3

**Explanation:** 2 deletions and 1 insertion

**p** and **h** deleted from **heap**. Then, **p** is

inserted at the beginning One thing to

note, though **p** was required yet it was

removed/deleted first from its position

and then it is inserted to some other

position. Thus, **p** contributes one to the

**deletion\_count** and one to the

**insertion\_count**.

**Example 2:**

**Input :** str1 = "geeksforgeeks"

str2 = "geeks"

**Output:** 8

**Explanation:** 8 insertions

**Your Task:**  
You don't need to read or print anything. Your task is to complete the function **minOperations()**which takes both strings as input parameter and returns the minimum number of operation required.

**Expected Time Complexity:**O(|str1|\*|str2|)  
**Expected Space Complexity:**O(|str1|\*|str2|)

**Constraints:**  
1 ≤ |str1|, |str2| ≤ 1000  
All the characters are lower case English alphabets

class Solution{

    public:

    int minOperations(string str1, string str2) {

        // Your code goes here

        int m=str1.length(), n=str2.length();

        int dp[m+1][n+1];

        memset(dp, 0, sizeof(dp));

        for (int i=1; i<m+1; i++) {

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

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

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

            }

        }

        int len=dp[m][n];

        return m+n-(2\*len);

    }

};