#include <bits/stdc++.h>

#include<algorithm>

using namespace std;

//check recursion and look for sub problems ; if yes then it's DP problem

int LCS(string s1, string s2, int m, int n) //O(2^n)

{

if(m==0 || n==0) return 0;

if(s1[m-1]==s2[n-1])

{

return 1+LCS(s1,s2,m-1,n-1);

}

else {

return max(LCS(s1,s2,m-1,n),LCS(s1,s2,m,n-1));

}

}

//require 2d array as 2 parameters change in calling

//O(m\*n)

int mem[1000][1000];

int o\_LCS(string s1, string s2, int m, int n)

{

if(mem[m][n]!=-1) return mem[m][n];

if(m==0 || n==0) mem[m][n]={0};

else {

if(s1[m-1]==s2[n-1]) mem[m][n]=1+o\_LCS(s1,s2,m-1,n-1);

else mem[m][n]=max(o\_LCS(s1,s2,m-1,n),o\_LCS(s1,s2,m,n-1));

}

return mem[m][n];

}

int main() {

string a="ABXY";

string b="ABX";

memset(mem,-1,sizeof(mem));

cout<<o\_LCS(a,b,4,3);

return 0;

}