**Сравнение на символни низове**   
    Дадени са два символни низа *s*1 и *s*2. Задачата е от първия низ *s*1 да се получи втория низ *s*2, като се използват операциите:   
    - replace(i, x)- замества i-тия символ на *s*1 със символа x;   
    - insert(i, x) - вмъква символа x на позиция i в *s*1;   
    - delete(i) - изтрива i-тия символ на *s*1.   
За всяка от трите операции е зададена цена. Търсим крайна редица от операции за решаване на задачата, която е с минимална цена.   
    Нека *F*(*i, j*) е минималната цената за уеднаквяване на началните последователности (префикси)

*s*1[1], *s*1[2], ..., *s*1[i]  и *s*2[1], *s*2[2], ..., *s*2[j].

Индексите са взети по началните низове. Нека е *cr* цената за заместване на символ от *s*1 със символ от *s*2; *ci* е цената за вмъкване на символ в *s*1 и *cd* е цената за изтриване на символ от *s*1.

    Ако *s*1[i] = *s*2[j], то

*F*(*i, j*) = *F*(*i -* 1*, j -* 1),

в противен случай ще се вземе минималното от трите числа:

*F*(*i -* 1*, j -* 1) + *cr*   
*F*(*i,  j -* 1) + *ci*   
*F*(*i -* 1*,  j*) + *cd*

Остава да се пресметнат стойностите на *F*(*i,* 0) и *F*(0*, j*). *F*(*i,* 0) означава празен втори низ, следователно *F*(*i,* 0) = *i* *cd*. Аналогично при празен първи низ получаваме *F*(0*, j*) = *j* *ci*.

**// trans.c**   
**#include <stdio.h>**   
**#include <string.h>**   
**#define MAX 100**   
**#define min2(a,b) (((a)<(b)) ? (a) : (b))**   
**#define min3(a,b,c) (min2(min2(a,b),(c)))**

**#define COST\_DELETE 1**   
**#define COST\_INSERT 2**   
**#define COST\_REPLACE(i,j) ((s1[i] == s2[j]) ? 0 : 3)**

**unsigned F[MAX+1][MAX+1];**   
**unsigned n1, n2;**

**const char \*s1 = "\_abracadabra";**   
**const char \*s2 = "\_mabragabra";**

**unsigned editDistance(void)**   
**{ unsigned i,j;**   
**for (i=0; i<=n1; i++) F[i][0] = i\*COST\_DELETE;**   
**for (j=0; j<=n2; j++) F[0][j] = j\*COST\_INSERT;**

**for (i=1; i<=n1; i++)**   
**for (j=1; j<=n2; j++)**   
**F[i][j] = min3(F[i-1][j-1] + COST\_REPLACE(i,j),**   
**F[i  ][j-1] + COST\_INSERT,**   
**F[i-1][j  ] + COST\_DELETE);**   
**return F[n1][n2];**   
**}**

**void printEditOperations(unsigned i, unsigned j)**   
**{**   
**if (j==0) for (j=1; j<=i; j++) printf("DELETE(%u) ",j);**   
**else if (i==0) for (i=1; i<=j; i++) printf("INSERT(%u, %c) ",i, s2[i]);**   
**else if (i>0 && j>0)**   
**{ if ( F[i][j] == F[i-1][j-1] + COST\_REPLACE(i,j) )**   
**{ printEditOperations(i-1,j-1);**   
**if (COST\_REPLACE(i,j)>0) printf("REPLACE(%u, %c) ", i, s2[j]);**   
**}**   
**else if (F[i][j] == F[i][j-1] + COST\_INSERT)**   
**{ printEditOperations(i,j-1);**   
**printf("INSERT(%u, %c) ", i, s2[j]);**   
**}**   
**else if (F[i][j] == F[i-1][j] + COST\_DELETE)**   
**{ printEditOperations(i-1,j);**   
**printf("DELETE(%u) ", i);**   
**}**   
**}**   
**}**

**int main()**   
**{ n1 = strlen(s1)-1; n2 = strlen(s2)-1;**   
**printf("Distance: %u\n", editDistance());**   
**printEditOperations(n1,n2);**   
**printf("\n");**   
**return 0;**   
**}**

|  |
| --- |
| **Distance: 7**  **INSERT(1, m) DELETE(4) DELETE(5) REPLACE(7, g)** |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | **\_abracadabra** |
| **INSERT(1, m)** | 2 | **012345678901**  **\_abracadabra** | **\_mabracadabra** |
| **DELETE(4)** | 1 | **012345678901**  **\_abracadabra** | **\_mabrcadabra** |
| **DELETE(5)** | 1 | **012345678901**  **\_abracadabra** | **\_mabradabra** |
| **REPLACE(7, g)** | 3 | **012345678901**  **\_abracadabra** | **\_mabragabra** |
|  | **7** |  |  |

Southeastern European Regional Programming Contest   
Bucharest, Romania, October 18, 2003

Problem F   
**Common Subsequence**

Input File: F.IN   
Program Source File: F.PAS or F.C or F.CPP or F.JAVA

A subsequence of a given sequence is the given sequence with some elements (possible none) left out. Given a sequence *X* = *<x*1, *x*2, ..., *xm*> another sequence *Z* = <*z*1, *z*2, ..., *zk*> is a subsequence of *X* if there exists a strictly increasing sequence <*i*1, *i*2, ..., *ik*> of indices of *X* such that for all *j* = 1,2,...,*k*, *xij* = *zj*. For example, *Z* = <*a, b, f, c*> is a subsequence of *X* = <*a, b, c, f, b, c*> with index sequence <1, 2, 4, 6>. Given two sequences *X* and *Y* the problem is to find the length of the maximum-length common subsequence of *X* and *Y*.

The program input is from a text file. Each data set in the file contains two strings representing the given sequences. The sequences are separated by any number of white spaces. The input data are correct. For each set of data the program prints on the standard output the length of the maximum-length common subsequence from the beginning of a separate line.

Input   
**abcfbc         abfcab**   
**programming    contest**   
**abcd           mnp**

Output   
4   
2   
0