/\*

Proiect realizat de Zugravu Cristina-Vasilica, CR1.2B, F.A.C.E

Concluzii generale:

Avem o regula si un fragment de cod si trebuie sa aducem fragmentul de cod la regula in numarul minim de operatii

Operatia consta in: (aceste operatiii sunt de cost 1)

1) Schimbarea unei litere

2) Stergerea unei litere

3) Adaugarea unei litere

Consider ca inversiunea facuta in exemplu este de fapt schimbarea ambelor litere pe rand, mai sus in enunt nu ne spune nimic ca operatia de inversiune este o operatie de cost 1.

Trebuie sa vedem problema ca o problema de programare dinamica, intrucat gandirea noastra trebuie sa fie in modul respectiv:

Pentru a ajunge de exemplu de la "ab" la "aaba" trebuie sa ne gandim care e minimul intre: adaugarea unei litere, stergerea unei litere sau schimbarea unei litere

Calculand totul intr-o matrice dp putem compara foarte simplu acest minim, insa gandirea din spate trebuie sa fie corecta, astfel incat sa nu gasim erori in implementarea algoritmului nostru

\*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define DIM 105 /// Vom considera sirul de maxim 100 de litere, insa pentru siruri sau valori mai lungi putem schimba extrem de usor dimensiunea

int MinOfTwo(int a, int b) /// Pentru simplificarea codului, vom face functia min din C++

{

if(a>b)

return b;

else

return a;

}

char fragment\_cod[DIM];

char regula[DIM];

int dp[DIM][DIM];

int NrMinimOperatii(char regula[DIM], char fragment\_cod[DIM], int dp[DIM][DIM], int n, int m)

{

/// Initializam prima linie si prima coloana a matricei dp cu valorile corespunzatoare

/// pentru transformarea unui sir vid in sirul regula sau fragment\_cod

for(int i=0;i<=n;i++)

{

dp[i][0]=i;

}

for(int j=0;j<=m;j++)

{

dp[0][j]=j;

}

/// Parcurgem ambele siruri, regula si fragment\_cod

for(int i=0;i<n;i++)

{

for(int j=0;j<m;j++)

{

if(regula[i]==fragment\_cod[j]) ///Verificam daca caracterele de pe pozitiile i si j din cele 2 siruri sunt egale

/// Daca caracterele sunt egale atunci costul minim este egal cu costul pana-n punctul respectiv

/// adica costul de transformare pana la caracterele precedente

dp[i+1][j+1]=dp[i][j];

else

/// Daca caracterele sunt diferite, determinam costul minim de transformare pentru a aduce

/// fragment\_cod[j] la regula[i] folosind una din cele trei operatii:

/// 1) Schimbarea unei litere

/// 2) Stergerea unei litere

/// 3) Adaugarea unei litere

dp[i+1][j+1]=MinOfTwo(dp[i+1][j],MinOfTwo(dp[i][j+1],dp[i][j]))+1;

}

}

/// Returnam numarul minim de operatii necesare pentru a transforma fragmentul de cod in regula data

return dp[n][m];

}

int main (){

/// fragment\_cod = "fnuc(myFuncion"

/// regula = "func(myFunction)"

printf("Introduceti codul care va defini regula programului: ");

scanf("%s",regula);

printf("Introduceti fragmentul de cod dorit: ");

scanf("%s",fragment\_cod);

int n=strlen(regula); /// lungimea codului valid

int m=strlen(fragment\_cod); /// lungimea fragmentului de cod introdus

int nr=NrMinimOperatii(regula,fragment\_cod,dp,n,m); /// numarul minim de operatii cerut

printf("Numarul minim de operatii pentru a ajunge la un cod valid de la fragmentul introdus este egal cu ");

printf("%d",nr);

printf("!\n");

return 0;

}