## Número de subsecuencias felices y tristes — X68437_es

En este ejercicio tendréis que hacer varias cosas.
En primer lugar, tendréis que implementar una función tal que, dado un string $s$ y tres caracteres diferentes $c_1, c_2, c_3$, recibidos como parámetro, retorna cuantas veces aparece $c_1c_2c_3$ como una subsecuencia (no consecutiva) dentro de `s`. En otras palabras, retorna el número de tripletas de índices $(i_1, i_2, i_3)$ tales que $i_1 < i_2 < i_3$ y $s[i_1] = c_1$, $s[i_2] = c_2$, $s[i_3] = c_3$. Esta es la cabecera:

```
// Pre: c1,c2,c3 are pairwise different characters.
// Post: returns the number of triples (i1,i2,i3) such that 0<=i1<i2<i3<s.size(
//       s[i1]=c1, s[i2]=c2, s[i3]=c3.
int numberSubsequences(const string &s, char c1, char c2, char c3);
```

**Nota:** Los juegos de pruebas privados de este ejercicio son grandes y estan diseñados para que haga falta una implementación de coste lineal de `numberSubsequences`. Una implementación lenta os permitirá solamente superar los juegos de pruebas públicos y obtener la mitad de la nota.
En segundo lugar, tendréis que implementar dos funciones que calculen el número de subsecuencias felices y tristes en un string, respectivamente. Una subsecuencia feliz es una subsecuencia de tres caracteres, y donde esos caracteres son, o bien **:-)** o bien **(-:**, en el orden dado. Una subsecuencia triste es una subsecuencia de tres caracteres, y donde esos caracteres son, o bien **:-(** o bien **)-:**, en el orden dado. Estas son las cabeceras:

```
// Pre:
// Post: returns the number of triples (i1,i2,i3) such that 0<=i1<i2<i3<s.size(
//       either s[i1]=':', s[i2]='-', s[i3]=')' or s[i1]='(', s[i2]='-', s[i3]=
int numberHappySubsequences(const string &s);

// Pre:
// Post: returns the number of triples (i1,i2,i3) such that 0<=i1<i2<i3<s.size(
//       either s[i1]=':', s[i2]='-', s[i3]='(' or s[i1]=')', s[i2]='-', s[i3]=
int numberSadSubsequences(const string &s);
```

Las dos funciones anteriores deben usar convenientemente la función `numberSubsequences` mencionada al principio. En caso contrario, se invalidará la entrega.
Finalmente, tenéis que implementar un programa principal que lee strings de entrada y, para cada uno de ellos, escribe su número de subsecuencias felices y su número de subsecuencias tristes.
Este programa tendrá que usar convenientemente las funciones mencionadas anteriormente. En caso contrario, se invalidará la entrega.

### Entrada

La entrada tiene varios strings sobre {`':'`,`'-'`,`'('`,`')'`}, cada uno en una linea.

## Salida

Para cada caso, la salida tiene dos números separados por un espacio en blanco en una linea, el número de subsecuencias felices y el número de subsecuencias tristes.

## Ejemplo de entrada

```
-):)
-())-:-:::-(-((:(
)---:::)
--)
)(:-)))):
)-(:-
-:(()--()))(:(
-)))(:
)-:
(:-:((-((::-:((
(:(---::---:))-)(
()(-)))(:():)
:):)::())-)-::((:(
::-((:()(:
)-:)::(()-)(-((-:(
::):
(:)(::-)-:)):)-(:
()((::)--((-
()))((:-:
:()(-):--)-
```

## Ejemplo de salida

```
0 0
10 43
0 9
0 0
4 1
0 1
10 8
0 0
0 1
18 22
64 16
4 2
10 51
2 8
16 39
0 0
35 25
0 8
3 3
8 1
```

## Observación

Evaluación sobre 10 puntos:

- Solución lenta: 5 puntos.
- Solución rápida: 10 puntos.

Entendemos como solución rápida una que es correcta, de coste lineal y capaz de superar los juegos de pruebas públicos y privados. Entendemos como solución lenta una que no es rápida, pero es correcta y capaz de superar los juegos de pruebas públicos.

## Información del problema

Autor : PRO1
Generación : 2023-11-27 23:07:08