Number of happy and sad subsequences — X68437_en

In this exercise you need to do several things.
To begin with, you have to implement a function such that, given a string s and three different characters c_1, c_2, c_3 as parameters, returns how many times $c_1c_2c_3$ occurs in `s` as (non-consecutive) subsequence. In other words, it returns the number of triples of indexes (i_1, i_2, i_3) holding $i_1 < i_2 < i_3$ and $s[i_1] = c_1$, $s[i_2] = c_2$, $s[i_3] = c_3$. This is the header:

```
// 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);
```

Note: The private tests of this exercise are big and are designed so that a linear time implementation of `numberSubsequences` is required to overcome them. A slow implementation will let you pass public tests and hence just get half the score.
Secondly, you'll need to implement two functions computing the number of happy and sad subsequences of a given string, respectively. A happy subsequence is one with three characters, either **:-)** or **(-:**, in the given order. A sad subsequence is one with three characters, either **:-(** or **)-:**, in the given order. These are the headers:

```
// 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);
```

Both previous functions must use function `numberSubsequences` mentioned before. Otherwise, the submission will be invalidated by human assessors later on.
Finally, you have to implement a main program that reads input strings and, for each of them, it prints its number of happy subsequences and its number of sad subsequences.
This program must make convenient use of former functions. Otherwise, the submission will be invalidated by human assessors later on.

Input

The input has several strings over {`':'`,`'-'`,`'('`,`')'`}, each at a line.

Output

For each case, the output has two numbers separated by a white space on one line, the number of happy subsequences and the number of sad subsequences.

Sample input

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

Sample output

```
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
```

Observation

Grading up to 10 points:

- Slow solution: 5 points.
- Fast solution: 10 points.

We understand as a fast solution one which is correct, with linear cost and which passes the public and private tests. We understand as slow solution one which is not fast, but it is correct and passes the public tests.

Problem information

Author : PRO1
Generation : 2023-11-27 23:07:06