Number of happy and sad subsequences in a string with three sections — X16298_en

In this exercise you have to carry out several tasks.
Firstly, you'll have to implement a function that receives a string s holding some very particular conditions: s is formed using three different characters c_1, c_2, c_3. Moreover, at the beginning of s there are one or more occurrences of character c_1, next there are one or more occurrences of character c_2, and finally there are one or more occurrences of character c_3. The function will also be given three extra integer parameters n_1, n_2, n_3 by reference, where it has to assign the number of occurrences of c_1, c_2, c_3, respectively. This is the header:

```
// Pre: s is formed with three different characters c1,c2,c3, and is of the for
// Post: n1, n2, n3 are the number of occurrences of c1, c2, c3 in s, respectiv
void numberOccurrences(const string &s, int &n1, int &n2, int &n3);
```

Note: the private tests of this exercise are big and designed so that one needs an implementation with logarithmic cost of `numberSubsequences`. A slow implementation will let you overcome the public tests and get half the score.
Secondly, you'll have to implement a function that receives a string s holding one of the following conditions:

1. s begins with one or more occurrences of :, followed by one or more occurrences of $-$, followed by one or more occurrences of), and there are no more characters in s.
2. s begins with one or more occurrences of (, followed by one or more occurrences of $-$, followed by one or more occurrences of :, and there are no more characters in s.
3. s begins with one or more occurrences of :, followed by one or more occurrences of $-$, followed by one or more occurrences of (, and there are no more characters in s.
4. s begins with one or more occurrences of), followed by one or more occurrences of $-$, followed by one or more occurrences of :, and there are no more characters in s.

In former cases (1) and (2), the function will return the number of happy subsequences of s, and in former cases (3) and (4), the function will return the number of sad subsequences of s. A happy subsequence is one having three characters, where those three characters are, either **:-)** or **(-:**, in the given order. A sad subsequence is one having three characters, where those three characters are, either **:-(** or **)-:**, in the given order. This is the header:

```
// Pre: s begins with one or more occurrences of a character c1, followed by on
//      occurrences of a character c2, followed by one or more occurrences of a
//      and there are no more characters in s.
//      moreover, either c1c2c3 = ":-)" or c1c2c3 = "(-:" or c1c2c3 = ":-(" or
// Post: If c1c2c3 = ":-)" or c1c2c3 = "(-:", the function returns the number o
//       If c1c2c3 = ":-(" or c1c2c3 = ")-:", the function returns the number o
int numberHappyOrSadSubsequences(const string &s);
```

Former function must conveniently use function `numberOccurrences` described at the beginning. Otherwise, the delivery will be invalidated.

Observation

You only need to submit the required procedure; your main program will be ignored.

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 logarithmic 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 : 2023-12-19 13:12:29