Substring occurrences X28106_en

Write the subprogram

```
int substring_from_k(const string& x, int k, const string& y)
```

that finds whether the string x occurrs in string y in position k or higher. If it is found, it returns in which position, otherwise, it returns -1
For example assuming x = "la" and y = "aalaaalaaa", the function results would be:

- if $k = 0$, it would return 2.
- if $k = 2$, it would return 2.
- if $k = 3$, it would return 6.
- if $k = 7$, it would return -1.

Next, write another function:

```
vector<int> substrings(const string &x, const string &y);
```

that given two strings x and y returns a vector with all the positions in which x occurs in y.
The main program (already provided) uses the above functions to process a sequence of pairs of strings x_i y_i, and for each pair creates and stores a `Parst` struct, that contains the string (s), the searched substring (sub), the vector of occurrence positions (vap), and the position of the pair in the input sequence ($index$).

```
struct Parst {
      int index;
      string sub, s;
      vector<int> vap;
};
```

The program outputs each string pair in the input, with the positions where the substrings were found, sorted in alphabetical order by the substring, and by the pair position in the input in case of tie.
Thus, you need also to complete the comparison function that will sort the structs appropiately

```
bool comp (const Parst& psa, const Parst& psb);
```

You MUST use the following code. Changing it outside the indicated blocks will render your program INVALID

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;

struct Parst {
      int index;
      string sub, s;
      vector<int> vap;
};

// Pre: 0 <= k < y.size()
// Post: The result is the first position i>=k where substring x is found in y,
//        or -1 if no such position exists
int substring_from_k (const string& x, int k,  const string& y) {
   /// ADD YOUR CODE HERE
}

// Pre: x.size()>0 and y.size()>0
// Post: The result is a vector containing all the positions in y where
//        substring x occurs.
vector<int> substrings(const string& x, const string& y) {
   /// ADD YOUR CODE HERE
}

// Comparison function to sort the output as required
bool comp (const Parst& psa, const Parst& psb) {
   /// ADD YOUR CODE HERE
}

int main() {
     vector<Parst> vparst;
     Parst pst;
     pst.index = 1;
     while (cin >> pst.sub >> pst.s) {
        pst.vap = substrings(pst.sub,pst.s);
        vparst.push_back(pst);
        ++pst.index;
     }
     sort(vparst.begin(),vparst.end(),comp);
     int vpn = vparst.size();
     for (int i = 0; i < vpn;++i){
        cout << vparst[i].sub << " " << vparst[i].s;
        int n = vparst[i].vap.size();
        for (int j = 0; j < n; ++ j)
            cout << " " << vparst[i].vap[j];
        cout << endl;
    }
 }
```

Exam score: 3.500000 **Automatic part:** 30.000000%

Input

The input consists of a sequence of pairs of non-empty strings.

Output

The output consists of each pair of strings x and y, together with the list of positions where substring x occurs in y. The pairs must be sorted alphabetically by the substring x, and by their position in the input in case of a tie.

Sample input

```
la aaaalaaaaalaaaaa
la lalalalala
la aaaaa
la aaaalala
coco cocococoo
coco aaaacocococoaaaa
```

Sample output

```
coco cocococoo 0 2 4 6
coco aaaacocococoaaaa 4 6
la aaaalaaaaalaaaaa 4 10
la lalalalala 0 2 4 6 8
la aaaaa
la aaaalala 4 6
```

Problem information

Author : Nikos Mylonakis Pascual
Generation : 2019-01-08 10:20:05

© *Jutge.org*, 2006–2019.
https://jutge.org