Formal Languages and Compilers

26 January 2022

Using the JFLEX lexer generator and the CUP parser generator, realize a JAVA program capable of recognizing and executing the programming language described in the following.

Input language

The input file is composed of three sections: *header*, *components* and *PC* sections, separated by means of the sequence of characters "`***`" or "`*****`". Comments are possible, and they are delimited by the starting sequence "{{{" and by the ending sequence "}}}".

Header section: lexicon

The *header* section can contain 2 types of tokens, each terminated with the character ";":

- `<tok1>`: It is composed of the character "A", a "_", and by 3, 7, or 13 repetitions of even numbers between −36 and 382. Numbers are separated by a "`*`". Example: `A_-36*184*0`
- `<tok2>`: It is composed of the character "B", a "_", an even number of repetitions, at least 4, of the words "`aaa`", "`bb`", and "`c`", and optionally it is followed by a date in the format YYYY/MM/DD between 2021/10/13 and 2022/03/30. Remember the the months of November and February have 30 and 28 days, respectively. Example: `B_aaabbccbbc2022/03/30`

Header section: grammar

The *header* section contains one of these two possible sequences of tokens:

1. **1 or 2** `<tok1>`, and **any number** of `<tok2>` (**even 0**) in any position of the sequence except for the first position. This sequence **must start** with a `<tok1>`, the second repetition of `<tok1>` can be in **any position** of the sequence.
2. at **least 5**, and in **odd** number (5, 7, 9,...) repetitions of `<tok2>`, followed by **0, 5, 10, or 15,...** repetitions of `<tok1>` (i.e., the number of repetitions of `<tok1>` must be **0** or **multiple of** 5).

Components section: grammar and semantic

The *components* section is composed of a list of `<component_type>`. Each `<component_type>` is composed of a `<type>`, a "`->`", a "`[`", a `<component_list>`, a "`]`", and a ";".

A `<component_list>` is a non-empty list of `<component>` separated with "`-`". A `<component>` is a `<unit_price>` (i.e., a *real* and *positive* number with *two decimals*), the word "`euro`", and a `<component_name>`. Tokens `<type>` and `<component_name>` are *quoted strings*.

At the end of this section, all the information needed for the following *PC* section must be stored into an entry of a global symbol table with key `<type>`. **This symbol table is the only global data structure allowed in all the examination, and in the components section can only be written, while in the PC section can only be read**.

In addition, at the end of this section, for each `<type>` the translator must compute and print the average between all the `<unit_price>` associated with this `<type>` (for the output format see the example).

PC section: grammar and semantic

The *PC* section is composed of a list, which can be **empty**, of <pc>. Each <pc> is a <pc_name> (i.e., a *quoted strings*), a <pc_quantity> (i.e., an *unsigned integer* number), a ":", a non-empty list of <piece> separated with ",", and a ";". Each <piece> is a <type> a "[", a non-empty list of <elem> separated with "-" and a "]". Each <elem> is a <elem_quantity> (i.e., an *unsigned integer* number), and a <component_name>.

In this section, for each <component_name>, the translator must print the <component_name>, the <pc_quantity> associated to the <component_name> (access it through **inherited attributes**), a "*", the <elem_quantity> associated to the <component_name>, a "*", the <unit_price> of the <component_name> referred to the related <type> (access the <type> through **inherited attributes** in order to retrieve the <unit_price> from the symbol table), and the result of the multiplication between <pc_quantity>, <elem_quantity>, and <unit_price> (see the example).

Goals

The translator must execute the language, and it must produce the output reported in the example. For any detail not specified in the text, follow the example.

Example

Input:

```
{{{ Header section: first type of header }}}
A_-22*12*14*20*100*328*200 ;   {{{ tok1 }}}
B_cccc;                        {{{ tok2 }}}

*** {{{ division between header and components sections }}}
{{{ Components section }}}

{{{ AVG (150.00+50.00+70.00)/3=90.00 }}}
"electronic" -> [ 150.00 euro "CPU" - 50.00 euro "RAM 4GB" - 70.00 euro "SSD 512MB" ];

{{{ AVG (12.00+5.00)/2=8.50 }}}
"peripherals" -> [ 12.00 euro "Wi-Fi adapter" - 5.00 euro "Ethernet" ] ;

{{{ AVG (40.00+200.00)/2=120.00 }}}
"others" -> [ 40.00 euro "Power supply" - 200.00 euro "Motherboard" ];

***** {{{ division between components and PC sections }}}
{{{ PC section }}}
"PC" 10 : "electronic" [ 2 "SSD 512MB" - 1 "CPU" - 2 "RAM 4GB" ],
          "peripherals" [ 1 "Wi-Fi adapter" ],
          "others" [ 1 "Motherboard" - 2 "Power supply" ];

"PC part" 2 :  "electronic" [ 4 "CPU" - 2 "RAM 4GB" ];
```

Output:

```
AVG "electronics" 90.00
AVG "peripherals" 8.50
AVG "others" 120.00
***
"SSD 512MB" 10*2*70.00 1400.00
"CPU" 10*1*150.00 1500.00
"RAM 4GB" 10*2*50.00 1000.00
"Wi-Fi adapter" 10*1*12.00 120.00
"Motherboard" 10*1*200.00 2000.00
"Power supply" 10*2*40.00 800.00
"CPU" 2*4*150.00 1200.00
"RAM 4GB" 2*2*50.00 200.00
```

Weights: **Scanner** 8/30; **Grammar** 9/30; **Semantic** 10/30