Formal Languages and Compilers

26 February 2021

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*, *catalog* and *purchases* sections, separated by means of the sequence of characters "`$$$`". Comments are possible, and they are delimited by the starting sequence "`<<-`" and by the ending sequence "`->>`".

Header section: lexicon

The *header* section can contain 3 types of tokens, each terminated with the character ";":

- `<tok1>`: It is composed of the character "A", a "_", and 3, 23, or 54 repetitions of a number between −23 to 236. Numbers are separated by the character "#". Example: `A_-23#229#3`.
- `<tok2>`: It is composed of the character "B", a "_", and a date between "2020 November 06" and "2021 March 28". Remember that the month of November has 30 days, while the month of February 2021 has 28 days.
- `<tok3>`: It is composed of the character "C", a "_", and by an even number of repetitions, at least 6, of the words "`++`", "`--`", "`+-`" and "`-+`". Example: `C_++--++--+-+-`.

Header section: grammar

In the *header* section `<tok1>` and `<tok3>` can appear in **any order** and number (**also 0 times**), instead, `<tok2>` can appear only **0, 1 or 3** times.

Catalog section: grammar and semantic

The *catalog* section is composed of a list of **at least 3** `<category>` in **odd** number (i.e., 3, 5, 7,...). Each `<category>` is composed by a `<category_name>`, a "-", a non-empty list of `<product>` separated with ",", and a ";". A `<product>` is a `<product_name>`, followed by a `<code>`, a `<price>` and the word "`euro`". Tokens `<category_name>`, `<product_name>`, and `<code>` are *quoted strings*, while `<price>` is a *real* and *positive* number with *two decimals*. The translator must print for each `<category_name>` the *most expensive* `<product>` and the corresponding `<price>` (for the output format see the example).

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

Purchases section: grammar and semantic

The *purchases* section is composed of a non-empty list of `<purchase>`. Each `<purchase>` is a `<category_name>`, **optionally** followed by a `<discount>`, followed by "::", a non-empty list of `<pur_product>` separated with ",", and a ";". Each `<pur_product>` is a `<quantity>` (i.e., an *unsigned integer* number) followed by a `<code>`. `<discount>` is the character "-", a `<percentage>` (i.e., an *integer number*) and a "%". In this section, for each `<pur_product>`, the translator must multiply the `<quantity>` by the `<price>` of the product associated with the couple `<category_name>.<code>` (which can be accessed from the symbol table), apply the `<discount>` if present (i.e., to apply the discount subtract to the `<price>` the

quantity <price>*<discount>/100), and print the <product_name> associated to the <code>, and the result of the operation. At the end of the section, the total of all the previous results must be printed (see 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:

```
A_10#-22#0;             <<- tok1 ->>
C_++--++--+-+-++-- ;    <<- tok3 ->>
B_2020 December 03;     <<- tok2 ->>
A_-3#156#12 ;           <<- tok1 ->>

$$$ <<- division between header and catalog sections ->>

"cars" - "Red car"   "c1" 15000.00 euro,
         "Green car" "c2" 18000.00 euro;

"motorcycles" - "Fast motorcycle" "c1" 8000.00 euro;

"trucks" - "Big truck"    "c1" 70000.00 euro,
           "Medium truck" "c2" 60000.00 euro,
           "Small truck"  "c3" 50000.00 euro;

$$$ <<- division between catalog and purchases section ->>

"trucks" - 10 % :: 4 "c3", 1 "c1";
"cars" :: 2 "c1";
```

Output:

```
"Green car" 18000.00
"Fast motorcycle" 8000.00
"Big truck" 70000.00
---
"Small truck" 180000.00
"Big truck" 63000.00
"Red car" 30000.00
TOTAL: 273000.00
```

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