Formal Languages and Compilers

17 July 2023

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* and *dresses*, and *shopping* 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 "}}", or by C++ style comments (i.e., from the sequence of characters "//" to the end of the line).

Header section: lexicon

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

- `<tok1>`: consists of a hexadecimal number between 27A and 12b3, followed by a "*" and by an odd number of alphabetic characters, at least 5, followed by the character "-". It is optionally terminated by an even number of "*" characters, at least 4, or by the word YXY (where the number of X's must be odd: YXY, YXXXY, YXXXXXY, ...).
- `<tok2>`: is an IP address (four integers ranging from 0 to 255 and separated by a "."), followed by a "-", and followed by a date in the format "YYYY/MM/DD" ranging from 05/10/2023 to 03/03/2024. Please note that the month of November has only 30 days, and the month of February has 29 days.
- `<tok3>`: is composed of 3 or 5 numbers, each of which is composed of 4 or 6 digits. Numbers are separated by means of the characters "-" or "+".

Header section: grammar

In the *header* section the 3 tokens can appear in two ways:

1. at **least 3**, and in **odd** number (3, 5, 7,...) repetitions of `<tok1>`, followed by **3 or 9 or 10** repetitions of `<tok2>`
2. **1 or 2** `<tok2>`, and **any number** of `<tok1>` and `<tok3>` (**even 0**) in any position of the sequence except for the first. This sequence **must start** with a `<tok2>`, the second repetition of `<tok2>` can be in **any position** of the sequence.

Dresses section: grammar and semantic

The *dresses* section is composed of a list of clothes with **at least 3** `<dresses>` in **odd** number (i.e., 3, 5, 7,...).

Each `<dress>` is a `<dress_name>` (i.e., a quoted string), a "-", a `<prod_list>`, and a ";". The `<prod_list>` is a non-empty list of `<prod>` separated with ",", where each `<prod>` is a `<prod_id>` (i.e., an unsigned integer number), a `<prod_name>` (i.e., a quoted string), a `<prod_cost>` (i.e., a real number), and the word "euro". All the data of this section must be stored in a symbol table with `<dress_name>` as the key. **This symbol table is the only global data structure allowed in all the examination, and it can be written only in this section.**

Shopping section: grammar and semantic

The *shopping* section is composed of a list of `<purchasing>` commands. Each `<purchase>` is a `<dress_name>` a `<percent>` (i.e., an unsigned integer number), a "%", a "-", a list of `<purch_prod>` separated with ",".

A `<purch_prod>` is a `<prod_name>` and a `<quantity>`. For each `<purchase>` the translator must print the `<dress_name>`, and for each `<purch_prod>` it must print the `<prod_name>`, the `<price>` and the word "euro". The `<price>` is obtained with the following operation: `<prod_cost>` * `<quantity>` * `<percent>` / 100. The `<prod_cost>` can be obtained from the symbol table by mandatorily accessing the `<dress_name>` and the `<percent>` through inherited attributes.

In addition, for each `<purch_prod>` the translator must compute and print the total price and the total number of purchased products (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
10.0.0.1-01/03/2024;     {{tok2}}
1234-123456+9876;        {{tok3}}
27b*abCdefg-;            {{tok1}}
130.192.78.1-29/02/2024; {{tok2}}
***
// Dresses section
"shirts"   - 11 "summer" 35.00 euro, 12 "winter" 40.00 euro, 13 "mid-season" 41.00 euro;
"trousers" - 14 "summer" 30.00 euro,
             15 "winter" 35.00 euro;
"shoes"    - 16 "man" 50.00 euro, 17 "woman" 90.00 euro;
***
// Shopping section
"shirts" 80 % - "summer" 1, "winter" 2;
"shoes"  90 % - "woman" 2;
```

Output:

```
"shirts"
"summer" 28.00 euro
"winter" 64.00 euro
TOTAL: 92.00 euro N_PROD: 3
"shoes"
"woman" 162.00 euro
TOTAL: 162.00 euro N_PROD: 2
```

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