Formal Languages and Compilers

26th June 2012

Using the JFLEX lexer generator and the CUP parser generator, realize a Java program capable of recognizing a language useful to manage auctions.

Input language

The input file is divided into three sections: header, currencies and auctions. Each section is separated from the other by means of the token "*****".

The header section is composed of two types of tokes:

- <hour>: is an hour with the formats HH:MM or HH:MM:SS and values between 08:31:12 and 23:21:10 (Correct examples are: 08:31, 23:21:09, 12:54:51)
- <code>: it begins **optionally** with a string of **at least 3** characters "X" or "Y" in any order and in **odd number** (ex. XXX, XYXYY, XXXXYYY,...), followed by an **even** number between −138 and 824

Each token is ended by the character ";". In the header section, the <code> token must appear **exactly two** times, while <hour> token must appear **at least one** time.

The currencies section is composed of a list of **at least 3** <currencies>. Each <currency> is composed of a <conversion_rate> (a floating point number with two decimals), of two strings of alphabetic characters, of the character ":", of a <user_list> and it is terminated by the ";" character. <user_list> is a list **eventually empty** of users separated by ",". Each <user> is composed of a <user_code> followed by a <capital> (a floating point number with two decimals).

<User_code> is a word composed of an **odd number** of **at least 3** alphabetic characters, followed by a dot ".", followed by some numbers separated by the character ".". Numbers have values between 12 and 132 (the numbers must be **at least 2** and in **even number**).

During the analysis of this section the translator will have to create in memory a hash table with key <user_code> and with associated value **the result of the multiplication** between the <conversion_rate> and the <capital>. The result of this operation is the capital in euro owned by a specific user. The hash table is the **only global variable** allowed. **Solutions using other global variables will not be accepted.**

The third section is composed of a list (**eventually empty**, of **at least 2** elements and in **even number**) of auctions.

Each element of the list is composed of the word "Auction", followed by an integer number, by the character ":", by a <product_name> (a quoted string), by the character ":", by a <duration> (an integer number), by the word "min", by an arrow "− >", by a **non empty** list of <advances> separated by the character "," and terminated by the character ";"

An <advance> is composed of a <user_code> followed by the character ":", by an <advance_time> (an integer number that represents when the advance has been made), by the character ":", by a <value> (a floating point number with two decimals that represents the amount of the advance) and by the word "euro".

Goals

The translator must write, for each <advance> made by a user, the <user_code> and the result of the advance: success or error. See the example. Particularly it must write:

- "Error, advance less than the current auction value": if the advance is lower then the current auction value. The auction value is the best advance without errors between the advances before the current one
- "Error, advance out of time": if <advance_time> is greater than <duration>
- "Error, available only XX.XX euro": if the user does not have enough money to buy the object. XX.XX are money still available for the given user (this value can be obtained from the hash table)
- "New auction price XX.XX euro": if none of the above errors is detected, the new auction price is the <value> of the current advance. This value must be printed in place of XX.XX

Assume that each auction start form 0.00 euro, that exist at least one correct advance and that advances are carried out in a chronological order (the <advance_time> values for a specific auction are increasing).

At the end of each auction the translator must print the <user_code> of the auction winner and the price at which the object is adjudicated.

Example

Input:

```
YYY-26; 21:12:00; 22:12; 562; 09:12:14;
*****
0.79 dollari euro  : Usr.13.13 200.00, Usr.13.14 300.00, Usr.13.15 100.00;
0.81 sterline euro : Usr.14.13 1000.00;
1.00 euro euro     : Usr.15.13 50.00, Usr.15.14 80.00;
*****
Auction 1 : "Product *1" : 100 min ->
       Usr.13.13 : 2  : 20.00 euro,
       Usr.13.15 : 20 : 12.00 euro,
       Usr.15.14 : 60 : 70.00 euro,
       Usr.15.13 : 65 : 75.00 euro,
       Usr.13.13 : 200: 100.00 euro;
Auction 2 : "Product *2" : 20 min ->
       Usr.15.14 : 10 : 20.00 euro,
       Usr.13.14 : 12 : 20.00 euro;
```

Output:

```
Auction 1:
 Usr.13.13: New auction price 20.00 euro
 Usr.13.15: Error, advance less than the current auction value
 Usr.15.14: New auction price 70.00 euro
 Usr.15.13: Error, available only 50.00 euro
 Usr.13.13: Error, advance out of time
Winner is Usr.15.14 price 70.00 euro
Auction 2:
 Usr.15.14: Error, available only 10.00 euro
 Usr.13.14: New auction price 20.00 euro
Winner is Usr.13.14 price 20.00 euro
```