Formal Languages and Compiles

7^{th} February 2006

Write, using JFLEX and CUP, a syntax-guided translator which recognizes a language for calculating the total dimension of mp3 files downloaded by a user.

Input Language

The input file is divided into two sections:

- the first one, identified by the keyword "mp3_list:" where a series of characteristics that identify the audio file (Bit/rate and length) are described;
- the second section is identified by the keyword "server:", in this section the mp3 files downloaded by each user are reported

First section:mp3_list:
In this section is present a series of mp3 files, subdivided by bit/rate.
Each line of this section is therefore initialized by a bit/rate, followed by the character ":" (colon) and by a list of mp3 files associated with a length expressed in seconds and ended by the character ";" (semicolon).
A bit/rate is an integer number followed by the string "Kb/s" (Kbits/second).
Each element of the mp3 files list and the relative length is composed by a file name with mp3 extension and by an integer which represents the song length in seconds. (e.g. WeAreTheChampion1.mp3 120).
The name of the mp3 file is a string of letters, numbers and "_" (Underscore) characters, the first character is a letter. The extension is always "mp3".
The elements of the list are separated by the character "," (Comma).

A mp3_list section example:

```
mp3_list:
256 Kb/s : TheShowMustGoOn_a12.mp3 150 , WeAreTheChampion1.mp3 120 , Time.mp3 30 ;
128 Kb/s : TheUnforgiven.mp3 200 ;
96 Kb/s : TheEnd.mp3 400 , NottePrimaDegliEsami.mp3 250 ;
```

After this first part of the input file has been analyzed, the parser must have created in memory an hash table where the key is given by the file name and the stored information is the total dimension in bits.
To calculate the dimension of each single file the bit/rate must be multiplied by the length. In the case of the file *TheShowMustGoOn.mp3*,it will have a dimension of 150*256 = 38400 bits.
To obtain the bit/rate value, in order to multiply it by the length and insert an entry inside the hash table, the use of the inherited attributes is required.
Solutions that do not use the inherited attributes or that use them in a wrong way (e.g. RESULT=(Integer)stack[top-2] instead of RESULT=(Integer)stack[top-1]) will make the semantic analysis part insufficient.

Generated hash table example:

KEY	VALUE
TheShowMustGoOn_a12.mp3	38400
WeAreTheChampion1.mp3	30720
Time.mp3	7680
TheUnforgiven.mp3	25600
TheEnd.mp3	38400
NottePrimaDegliEsami.mp3	24000

Second section: "server:"
The second section begins with the keyword "server".

Just after the starting section keyword ("server:") the section contains a keyword "data:" followed by a date in the format DD/MM/YYYY and a keyword "time:" followed by an hour in the format "HH:MM". The order in which the keywords appear in the input file is random (always after "server:", but they could be inverted) and must both be present and unique (use a proper grammar to handle this).
In the date format, DD is a number between 01 and 31, MM is a number between 01 and 12. In the hour format, HH is a number between 00 and 23, whereas MM is a number between 00 and 59. (Handle date and hour using scanner's regular expressions)

The input file ends with an eventually empty list of users and relative downloaded files.
Each element of the list is composed by an IP address, followed by the character ":" (colon), by a list of files with extension mp3, separated by the character "," (comma) and terminated by the character ";" (semicolon).
An IP address is a series of four integer numbers separated by a "." (dot). The integer numbers are in the range 0-255; numbers like 000 or 012 are not correct, whereas numbers 0 and 12 are. (Handle this using scanner regular expressions)
A server section example:

Lexically correct	**Lexically wrong (errors underlined)**
server: *time: 00:59* *data: 20/02/2006* *130.0.12.255 : TheEnd.mp3, Time.mp3, TheUnforgiven.mp3 ;* *130.0.12.254 : TheEnd.mp3 ;*	*server:* *data: 20/13/2006* *time: 0:59* *130.0.12.280 : TheEnd.mp3, Time.mp3, TheUnforgiven.mp3 ;* *130.0.012.254 : TheEnd.mp3 ;*

For each IP address the program must calculate the total dimension of the listed files, expressed in bits. Solutions that will use global variables for storing the total dimension will not be accepted. Use synthesized attributes and the predefined object RESULT for propagating the sum through the derivation tree.
The program output must be the following:

```
OUTPUT:
130.0.12.255
TheEnd.mp3 38400
Time.mp3 7680
TheUnforgiven.mp3 25600
TOTALE: 71680
130.0.12.254
TheEnd.mp3 38400
TOTALE: 38400
```

Example

Given the following input file

Input:

```
mp3_list:
256 Kb/s : TheShowMustGoOn_a12.mp3 150 , WeAreTheChampion1.mp3 120 , Time.mp3 30 ;
128 Kb/s : TheUnforgiven.mp3 200 ;
96 Kb/s: TheEnd.mp3 400 , NottePrimaDegliEsami.mp3 250 ;
server:
```

```
time: 00:59
data:20/02/2006
130.0.12.255 : TheEnd.mp3, Time.mp3, TheUnforgiven.mp3 ;
130.0.12.254 : TheEnd.mp3 ;
```

The program must produce the following output:

Output:

```
OUTPUT:
130.0.12.255
TheEnd.mp3 38400
Time.mp3 7680
TheUnforgiven.mp3 25600
TOTALE: 71680
130.0.12.254
TheEnd.mp3 38400
TOTALE: 38400
```