Introduction to Language Theory and Compilation
Exercises

Session 11: `lex/flex` scanner generator

Reminders

A *scanner* is a program that reads text on the standard input and prints it modified on standard output. For example, a filter that replaces all as with `b`s and that receives `abracadabra` on input would output `bbrbcbdbbrb`.

Specification format

A `JFex` (a Java implementation of flex) specification is made of three parts separated by lines with `%%`:

- **Part 1**: arbitrary programming code to be prepended in the output Java scanner program
- **Part 2**: regular expression definitions and arbitrary Java code (between `%{` and `%}`) to be inserted at the start of the scanner program
  - The JFlex options (%class **Name**, %unicode, %line, %column, %standalone, %cup, . . .)
  - The regular expression definitions are used as "macros" in part 3.
  - The Java additional code of the scanner (`%{ code }%`, `%init{ code executed before the parsing }init%`, `%eof{ code executed after the parsing }eof%`, . . .)
- **Part 3**: translation rules of the following shape: `Regex  {Action}`
  - `Regex` is an *extended regular expression* (ERE)
  - `Action` is a *Java code snippet* that will be executed each time a *token* matching `Regex` is encountered.
  - The regular expressions defined in Part 2 can be used by putting their names in curly braces `{ }`.

Variables and special actions

When writing *actions*, some special variables and macros can be accessed:

- `yylength()` contains the *length* of the recognized token
- `yytext()` is a the actual string that was matched by the regular expression.
- `yyline` is the line counter (requires the option %line).
- `yycolumn` is the column counter (requires the option %column).

Meta states

You can define inclusive or exclusive (use **x**state instead of state) states with the command:

```
%xstate states list separated by a comma;
```

Each state has to contain some regular expressions and action can be a change of state by using the function `yybegin(Name of the state)`. The first state used is defined by CUP and it called **YYINITIAL**. For instance:

```
⋮
xstate YYINITIAL, PRINT;
%%
<YYINITIAL> {
   "print" {yybegin(PRINT);}
}
<PRINT> {
   ";" {yybegin(YYINITIAL);}
   .   {System.out.println(yytext());}
}
```

Exercises

Ex. 1. Write a scanner that outputs its input file with line numbers in front of every line.

Ex. 2. Write a scanner that outputs the number of alphanumeric characters, alphanumeric words and alphanumeric lines in the input file.

Ex. 3. Write a scanner that only shows comments in the input file. Such comments are comprised within curly braces `{ }`.

Ex. 4. Write a scanner that transforms the input text by replacing the word "compiler" with "ewww" if the line starts with an "a", with "???" if it starts with a "b" and by "profit!!!" if it starts with a "c".

Ex. 5. Write a *lexical analysis function* that recognizes the following *tokens*:

- Decimal numbers in scientific notation (i.g. -0.4E-1)
- C99 variable identifiers (start by an alpha, followed by arbitrary number of alphanumeric or underscore)
- Relational operators ($<, >, ==, ! =, >=, <=, !$)
- The `if`, `then` and `else` keywords

The point of this function is then to be used by a `yacc` implementation, `cup` for `Java`.