Principles of Programming Languages [PLP-2014]
Homework on Compiler Front-End - March 13, 2015

The present homework is **optional**: no student will be penalized in the final evaluation of the course for not having solved it.

Deadline: Wednesday, April 1
Submission: Send all the files which compose the project, together with instructions for using them, by email to the lecturer

For any question write an email to andrea@di.unipi.it.

Goal: Generating an NFA which recognizes the language of a given Regular Expression

Instructions and constraints:

- Design a top-down parsable grammar G_{RE} for generating Regular Expressions. The grammars should not be ambiguous.
- Define an L-attributed Syntax-Directed Translation Scheme based on the grammar G_{RE} to generate the NFA associated with a regular expression using Thompson's algorithm.
- The output NFA has to be generated using the **dot language** as Intermediate Representation [see http://www.graphviz.org/content/dot-language]
- Implement the Syntax-Directed Translation Scheme using **yacc** or **bison**. As the lexical analysis is trivial, you can either use **lex/flex** or simply a function **yylex** to be defined in the third section of the yacc/bison specification.
- Use **dot** to compile the resulting file into **pdf** (or other format)

Some hints

- To play with **lex/flex** and **yacc/bison,** download (from the course pages) and study files **calc.l** and **calc.y**, defining lexer and scanner of a command line calculator
- You may use the following commands (or similar, depending on your system) to generate the executable:

```
yacc  -o parser.c calc.y
flex -o scanner.c calc.l
gcc -o calc parser.c
```

- The dot notation is simple: an example is **parser.dot**