Compilers Fall 2015 Project 1

Due We 10/7/2015 at 8:30am.

Project Milestone 1: Syntax Checker

Implement a syntax checker for Tack. The Tack Language Specification is available here:
http://www.inf.usi.ch/faculty/soule/teaching/2015-fall/cc/tack.pdf

For this project milestone, your compiler should accept one command-line parameter with the file name of the input program. If the input program contains at least one syntax error, your compiler should print an error message. Otherwise, it should not print any output. In other words, your compiler should print nothing if and only if the input was syntactically correct. In particular, for this milestone, you need to write productions for both lexical analysis and syntax analysis, but you do not yet need not generate an AST. However, it is recommended that you already incorporate associativity and precedence in the grammar and eliminate left-recursion, which will make future milestones easier.

Please turn in the entire code for your syntax checker (including the grammar, the main program, and auxiliary code, if any). You should also include a README file with instructions for how to compile and run your code.

As an example, assuming the main program for running your parser is in a class `Main`, and the input program `test/023.tack` has a syntax error, the command-line:

```
java -ea -cp .:/usr/local/lib/antlr-4.5.1-complete.jar Main test/023.tack
```

might lead to the following output:

```
test/023.tack:3:7: Syntax error.
```

The following archive file contains a few Tack programs that are syntactically correct. Therefore, they should "succeed" with your compiler, meaning it should not report any errors:
http://www.inf.usi.ch/faculty/soule/teaching/2015-fall/cc/pr1/success.tar

The following archive file contains a few Tack programs that are syntactically wrong. Therefore, they should "fail" with your compiler, meaning your compiler should report syntax errors:
http://www.inf.usi.ch/faculty/soule/teaching/2015-fall/cc/pr1/failure.tar

It is a good idea to also write additional test cases of your own (both succeeding and failing tests), to maximize test coverage.

You should write your syntax checker using the ANTLR4 parser generator. You can find a short introduction for ANTLR4 here:
http://www.inf.usi.ch/faculty/soule/teaching/2015-fall/cc/antlr-intro.pdf

---