CS402 – Compiler Design
Homework 2

2014–2015 Fall

1 Description

In this homework, you will implement a type checker for a programming language called PL.

You are expected to write a flex/bison scanner parser, that parses PL programs, and a type checker that will check the type rules of PL given below.

2 Input and output

The input to your parser will be PL programs. Your parser should accept the name of the input file as a command line argument. You may assume that the input will be syntactically correct.

The output will be consisting of type errors that are found in the input. Each type error will be printed on a separate line. Explanatory messages, at least the line on which the error occurs, must be displayed.

3 PL Grammar

$$
\begin{array}{rcl}
\mathit{Prgrm} & \rightarrow & \mathit{VarDeclLst}\ \mathit{StmtBlk} \\
\mathit{VarDeclLst} & \rightarrow & \mathit{VarDecl}\ \mathit{VarDeclLst} \mid \mathit{VarDecl} \\
\mathit{VarDecl} & \rightarrow & \mathit{Type}\ \mathbf{id}\ ; \mid \mathit{Type}\ \mathbf{id}\ [\ \mathbf{intnum}\ ]\ ; \\
\mathit{Type} & \rightarrow & \mathbf{int} \mid \mathbf{real} \mid \mathbf{boolean} \\
\mathit{StmtBlk} & \rightarrow & \mathbf{begin}\ \mathit{StmtLst}\ \mathbf{end} \\
\mathit{StmtLst} & \rightarrow & \mathit{Stmt}\ \mathit{StmtLst} \mid \mathit{Stmt} \\
\mathit{Stmt} & \rightarrow & \mathit{AsgnStmt} \mid \mathit{IfStmt} \mid \mathit{WhlStmt} \\
\mathit{AsgnStmt} & \rightarrow & \mathit{Lval} = \mathit{Expr}\ ; \\
\mathit{Lval} & \rightarrow & \mathbf{id} \mid \mathbf{id}\ [\ \mathit{Expr}\ ] \\
\mathit{IfStmt} & \rightarrow & \mathbf{if}\ (\ \mathit{Expr}\ )\ \mathbf{then}\ \mathit{StmtBlk}\ \mathbf{else}\ \mathit{StmtBlk} \\
\mathit{WhlStmt} & \rightarrow & \mathbf{while}\ (\ \mathit{Expr}\ )\ \mathit{StmtBlk} \\
\mathit{Expr} & \rightarrow & \mathbf{intnum} \mid \mathbf{realnum} \mid \mathbf{false} \mid \mathbf{true} \mid \mathbf{id} \mid \mathbf{id}\ [\ \mathit{Expr}\ ] \mid \\
 & & (\mathit{Expr}) \mid \mathit{Expr}\ \mathit{BinOp}\ \mathit{Expr} \mid \mathit{UnOp}\ \mathit{Expr} \\
\mathit{BinOp} & \rightarrow & + \mid * \mid < \mid == \mid \mathbf{and} \\
\mathit{UnOp} & \rightarrow & \mathbf{-} \mid \mathbf{not}
\end{array}
$$

Tokens:

- Integer constant (**intnum**): sequence of digits
- Real constant (**realnum**): (possibly empty) sequence of digits, followed by a single dot, followed by a non–empty sequence of digits.
- Identifier (**id**) : a letter followed by a sequence of letters and digits
- For other tokens, the lexemes are the same as the token names

Operator precedences (from highest to lowest): **not**, **-**, $*$, $+$, $<$, $==$, **and**
Operators are left associative.

4 Type rules

- In "**id** [*Expr*]", whether it is derived from an *Lval* or *Expr*:
  - *Expr* must be of integer type
  - **id** must be of an array type.
  - if *Expr* can be evaluated statically, then its value must not be out of bound (array indices start from 1)

- The operators in PL have the following semantic definitions

+	:	integer × integer → integer
+	:	real × integer → real
+	:	integer × real → real
+	:	real × real → real
∗	:	integer × integer → integer
∗	:	real × integer → real
∗	:	integer × real → real
∗	:	real × real → real
<	:	integer × integer → boolean
<	:	real × real → boolean
==	:	integer × integer → boolean
==	:	real × real → boolean
==	:	boolean × boolean → boolean
and	:	boolean × boolean → boolean
not	:	boolean → boolean
-	:	integer → integer
-	:	real → real

- Literals have the following types

intnum	:	→ integer
realnum	:	→ real
false	:	→ boolean
true	:	→ boolean

- In assignment statement, the left hand side and the right hand side must have the same type

- *Expr* of an if statement must be of type boolean

- *Expr* of a while statement must be of type boolean

5 How to submit

Submit a single tar.gz package on SUCourse that contains

- all the sources you've developed,

- Makefile that can be used to build your parser
- A word document containing your grammar (whether you modify it or not).

Your tar.gz file must be named as id.tar.gz, where id is your student.

The Makefile should produce an executable named as **pl**. Note that, the executables will be tested on `flow.sabanciuniv.edu`, so we recommend that you, at least, test your implementation on `flow.sabanciuniv.edu` before submitting.

6 Notes

- **Important**: SUCourse's clock may be off a couple of minutes. Take this into account to decide when to submit.
- No homework will be accepted if it is not submitted using SUCourse.
- Start working on the homework immediately.