CS 415 Project 1

Spring 2020

You may use any programming language that runs on the ilab machines for these problems.

1. Use a recursive descent parser to implement an interpreter for this (LL(1)) arithmetic grammar:

   $\langle Program \rangle$ ::= $\langle StmtList \rangle$.

   $\langle StmtList \rangle$ ::= $\langle Stmt \rangle$ $\langle NextStmt \rangle$

   $\langle NextStmt \rangle$ ::= ; $\langle StmtList \rangle$ | ϵ

   $\langle Stmt \rangle$ ::= $\langle Assign \rangle$ | $\langle Print \rangle$

   $\langle Assign \rangle$ ::= $\langle Id \rangle$ = $\langle Expr \rangle$

   $\langle Print \rangle$ ::= ! $\langle Id \rangle$

   $\langle Expr \rangle$ ::= + $\langle Expr \rangle$ $\langle Expr \rangle$
   | - $\langle Expr \rangle$ $\langle Expr \rangle$
   | * $\langle Expr \rangle$ $\langle Expr \rangle$
   | / $\langle Expr \rangle$ $\langle Expr \rangle$
   | $\langle Id \rangle$
   | $\langle Const \rangle$

   $\langle Id \rangle$ ::= a | b | c

   $\langle Const \rangle$ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

   The arithmetic operators indicate addition, subtraction, multiplication, and (integer) division, respectively.

   For example, on the input

   ```
   a=3;b=2;c=+ab;!c.
   ```

   Your program should print 5.

   For invalid inputs, such as

   ```
   a=+12;!a.b
   ```

   you should print "syntax error".

   For runtime errors, print "exception".

   Please treat referencing a variable before it's assigned a value as an error.

   Note that all tokens are single characters, and you can assume we will not include any whitespace, which should help simplify the scanner implementation.

2. Consider this grammar of boolean expressions:

$\langle Program \rangle$ ::= $\langle Expr \rangle$.

$$
\begin{array}{lll}
\langle Expr \rangle & ::= & \langle Expr \rangle \;\&\&\; \langle Expr \rangle \\
& | & \langle Expr \rangle \;||\; \langle Expr \rangle \\
& | & \langle Expr \rangle \;\hat{}\; \langle Expr \rangle \\
& | & \sim \langle Expr \rangle \\
& | & \langle Const \rangle \\
& | & (\; \langle Expr \rangle \;)
\end{array}
$$

$\langle Const \rangle$ ::= F | T

where the expression operators denote AND, OR, XOR, and NOT, respectively.

- What happens if you try to use this grammar in flex/bison?
- Modify the grammar to make flex/bison happy, without changing the language itself. Binary operators should have the precendence `&&, ^, ||` (highest to lowest) and be left associative.
- Use your modified grammar and flex/bison to implement an interpreter for the language. For invalid inputs, you should print "syntax error". For example, on input `F || T.`, you should print `T`. (just `T`, not the period)