CS4430 HOMEWORK 2 (50 POINTS TOTAL)

Issued: Thursday, March 16th, 2017.
Due: Wednesday, March 22nd, 2017 by 11:59pm.

DIRECTIONS

To turn in your solution, please email me the code (harrisonwl@missouri.edu) with the subject "CS4430 HW2". It is **important** that you get this small detail right because otherwise I may miss your submitted solution. Your solution should be in the form of a single Haskell file named "*Last-name*`_HW2.hs`". So, if I did this homework, I'd turn in "`Harrison_HW2.hs`".

PROBLEM DESCRIPTION

Here's a context-free grammar for a language we'll call "Toy":

<program>	→ { <stmt_list> }
<stmt_list>	→ <stmt> ; <stmt_list>
<stmt_list>	→ λ
<stmt>	→ <decl> | <assign> | <read_stmt> | <write_stmt>
<decl>	→ **integer** NAME | **float** NAME
<assign>	→ <var> := <expr>
<read_stmt>	→ **read** <var>
<write_stmt>	→ **write** <expr>
<expr>	→ <term> | <term> + <term> | <term> - <term>
<term>	→ INTEGERNUM | FLOATNUM | <var> | (<expr>)
<var>	→ NAME

Anything bracketed (e.g., <stmt>) is a non-terminal symbol and anything else is a terminal symbol. You can assume that tokens matching

- INTEGERNUM are Haskell **`Int`**'s,
- FLOATNUM are Haskell **`Float`**'s, and
- NAME are Haskell **`String`**'s.

Problem 1. (30 points) Write an abstract syntax for Toy in Haskell. Recall from class that this means that you will represent each nonterminal using a **`data`** declaration. So, your solution to this problem will be in the form of a number of datatype declarations in Haskell.

Problem 2. (20 points) Add your abstract syntax to the Haskell **`Show`** class. You may NOT use "`deriving`" at all.