Programming Assignment 6

Expression Trees, Stacks and a Postfix Evaluator

Submission Deadline: 11:55PM, 7th November 2014

Submissions only via Moodle:

In this assignment, you will consider modelling a language of arithmetical expressions as a tree data type. The language of expressions consists of real number constants (floats), some unary operators and some binary operators. Exactly which operators you need to represent will become clear when you see the examples that you have to handle.

The "assembly language" for the evaluator will be modelled as a data type `opcode` consisting of a constructor `NUM` for real constants, and for each operator an "op-code" represented as a constructor. For example, for addition we may have an opcode `PLUS`, etc.

The "compiler" you will write is merely a post-order traversal of the expression tree to yield a list of opcodes.

The evaluation machine will take such an opcode list, and a stack of floats (initially empty), and go through the opcode list, pushing numerical constants onto the stack, but for each operator opcode, popping the correct number of arguments off the stack, performing the corresponding operation and pushing the result onto the stack. When expression evaluation is complete, a single result should be sitting on the stack. Of course, if any erroneous situation is encountered, an exception should be raised.

Along with your code, please include the expression trees for the following mathematical expressions in your assignment file. Your code will be tested on them.

$$2.0 + \sin(1.0) \cdot \frac{e^{-7.1}}{3.6} \tag{1}$$

$$\tan \left| 31.6 \cdot \cos(3.14) + \sqrt{2.9 \cdot \sqrt{\frac{2.1}{3.0}}} \right| \tag{2}$$

$$0.5 - 2.3 + 9.1 \cdot \log \left(13.0 + \frac{12.9}{|2.3 - 4.5|} \cdot \sqrt{6.25} \right) + 3.0 \tag{3}$$

$$\sqrt{2.0 + \sqrt{2.0 + \sqrt{2.0\sqrt{2.0 + \sqrt{2.0 + 2.0}}}}} \tag{4}$$

$$-\log \left| 3.4 \cdot 6.8 - e^{11.5} \right| \cdot \sin(1.07 \cdot \sqrt{4.0}) + 34.1 \tag{5}$$

You may also be provided some other expressions at demo time and will have to write their expression tree and evaluate them. **Note:**

- You must use your Stack package.
- You must document your programs adequately, and test your program.
- You should also analyse the time and space required.