Haskell Problems

Contents

Parse Pseudo-Assembly

You are a TA at a university, and you want to evaluate your student's homework without executing their (untrusted) code. You decide to write a small web-service that takes bytecode as input, and interprets the results.

The bytecode language you need to support includes basic arithmetic and variables. The bytecode language is stack, rather than register based. ByteCode (right) is given for the following pseudo code (left):

```
function f() {

    x = 1                 // LOAD_VAL 1
                          // WRITE_VAR 'x'

    y = 2                 // LOAD_VAL 2
                          // WRITE_VAR 'y'

    return (x + 1) * y    // READ_VAR 'x'
                          // LOAD_VAL 1
                          // ADD

                          // READ_VAR 'y'
                          // MULTIPLY

                          // RETURN_VALUE
}
```

Add a data type `ByteCode` that can represent bytecode like in the example above, along with an interpreter for said bytecode. Make sure your code is total.

```haskell
data ByteCode = ...
runByteCode :: ...
```

Document the above exercise

Show and go over the functionality and design choices you've made in Haskell code you've written. This should include at least a rudimentary performance analysis.

You can use either the code from exercise (1) or provide a snippet from your personal project.

The triply-linked list

Discuss the problem of implementing a triply-linked list in Haskell to the best of your ability, including what you think it could be used for.

System to detect collisions

Implement a system for detecting collisions of paths with geometric shapes on a bounded subset of Z^2. Include a rudimentary visualization (may be terminal-based)