Pamphlet 3, INF222, Spring 2021

3.1 Calculator with register variables

In the previous pamphlet we introduced an enumeration `Register` of 10 distinct register names. Here we replace the `Register` type with `String`, giving more flexibility in how to designate the registers.

```haskell
-- | AST for register based integer calculator.
--
-- Author Magne Haveraaen
-- Since 2020-03-14

module CalculatorRegisterVariableAST where


----------------

-- | Expressions for a calculator with 10 registers.
-- The calculator supports literals and operations
-- Addition, multiplication, and subtraction/negation.
data CalcExprAST
  = Lit Integer
  | Add CalcExprAST CalcExprAST
  | Mult CalcExprAST CalcExprAST
  | Sub CalcExprAST CalcExprAST
  | Neg CalcExprAST
  | Reg String
  deriving (Eq, Read, Show)

-- | Statement for setting a register
data CalcStmtAST
  = SetReg String CalcExprAST
  deriving (Eq, Read, Show)


----------------

-- | A few ASTs for register based CalcExprAST.
calculatorRegisterAST1
  = Lit 4
calculatorRegisterAST2
  = Neg (Mult (Add (Lit 3) (Sub (Lit 7) (Lit 13))) (Lit 19))
calculatorRegisterAST3
  = Add (Reg "Reg1") (Reg "Reg4")
calculatorRegisterAST4
  = Reg "Reg2"

-- | A few ASTs for setting registers CalcStmtAST.
calculatorSetRegisterAST1
  = SetReg "Reg4" calculatorRegisterAST1
calculatorSetRegisterAST2
  = SetReg "Reg1" calculatorRegisterAST2
```

```
calculatorSetRegisterAST3
  = SetReg "Reg2" calculatorRegisterAST3
calculatorSetRegisterAST4
  = SetReg "Reg1" calculatorRegisterAST4

-------------------
```

For the interpreter we can still use strings like `"Reg0"`, . . . , `"Reg9"` as register names. We can still use the `CalculatorRegisterStore` as the store model, since it only needs the same range of integer indices.

3.2 Task

The task is to reimplement the register based interpreter, but using string names for the registers rather than the enumeration in pamphlet 2.

You should also adapt the unit tests and the interactive calculator (`main` and its support functions).