Assignment 16

calculator using scala

```
package calculator

sealed abstract class Expr
final case class Literal(v: Double) extends Expr
final case class Ref(name: String) extends Expr
final case class Plus(a: Expr, b: Expr) extends Expr
final case class Minus(a: Expr, b: Expr) extends Expr
final case class Times(a: Expr, b: Expr) extends Expr
final case class Divide(a: Expr, b: Expr) extends Expr


 object Calculator {
   def computeValues(
     namedExpressions: Map[String, Signal[Expr]]): Map[String, Signal[Double]] = {
     for {
       (variable, expression) <- namedExpressions
     } yield { (variable -> Signal(eval(expression(), namedExpressions))) }
   }

   def eval(expr: Expr, references: Map[String, Signal[Expr]]): Double = {
     expr match {
       case Literal(v) => v
       case Ref(name) => {
         val ref = getReferenceExpr(name, references)
         eval(ref, references - name)
       }
       case Plus(a, b)   => eval(a, references) + eval(b, references)
       case Minus(a, b)  => eval(a, references) - eval(b, references)
       case Times(a, b)  => eval(a, references) * eval(b, references)
       case Divide(a, b) => eval(a, references) / eval(b, references)
     }
   }
```

```
  private def getReferenceExpr(name: String,
                               references: Map[String, Signal[Expr]]) = {
    references.get(name).fold[Expr] {
      Literal(Double.NaN)
    } { exprSignal =>
      exprSignal()
    }
  }
}
```