函数程序设计实验一

一、实现分数的常用运算

编写一个实现分数四则运算的模块。

一个分数a/b可以表示为(a,b), 其中b>0，并且a与b没有大于1的公因子。例如，1/2 和-2/3可分别表示为(1,2)和(-2,3)。为此，我们定义表示分数的类型

```
type Fraction = (Integer, Integer)
```

1. 你的第一个任务是实现Fraction上的下列运算

```
ratplus  :: Fraction -> Fraction -> Fraction
ratminus :: Fraction -> Fraction -> Fraction
rattimes :: Fraction -> Fraction -> Fraction
ratdiv   :: Fraction -> Fraction -> Fraction
ratfloor :: Fraction -> Integer
ratfloat :: Fraction -> Float
rateq    :: Fraction -> Fraction -> Bool
```

前四个函数实现分数上的四则运算，ratfloor 将一个分数转换成不大于它的最大整数，ratfloat 将分数转换成浮点数，rateq判断两个分数是否相等。例如

```
Mail> ratplus (1,2) (1,2)
        (1, 1)
Mail> ratplus (1,2) (-1,2)
        (0, 1)
Main> ratplus (3,5) (rattimes (2,3) (2,1))
        (29, 15)
Main> ratminus (29,15) (3,1)
        (-16, 15)
Main> ratfloor (-16,15)
        -2
```

```
Main> ratfloat (15,8)
        1.875
Main> rateq (ratplus (1,2) (1,2)) (1, 1)
        True
```

注：你可能需要使用函数`fromInteger :: Integer -> a`, 它将一个类型为Interger的数转换为类型为a的数。

2. 使用下列运算符分别表示四则运算函数和相等运算：`<+>`，`<->`，`<-*->`，`</>`，`<==>`，并根据习惯规定其优先级。例如

```
infix 5 <+>
(<+>) :: Fraction -> Fraction -> Fraction
(<+>) (a,b) (c,d) = ratplus (a,b) (c,d)
infix 6 <-*->
(<-*->) :: Fraction -> Fraction -> Fraction
(<-*->) (a,b) (c,d) = rattimes (a,b) (c,d)
```

```
Main> (1,3) <+> (2,1) <-*-> (2,5)
        (17, 15)
```

注意，请务必使用给定的函数名。

二、测试你的函数定义

使用QuickCheck测试你的四则函数实现是否有错误。说明你测试了那些性质，并将这些性质写在你的模块中。例如，检查任意分数(a,b)加(0,1)结果仍然是(a,b)：

```
prop_ratplus_unit :: Fraction -> Property
prop_ratplus_unit (a,b) =  b > 0 ==>(a, b) <+> (0,1) <==> (a, b)
```

其中`prop_ratplus_unit`是表示该性质的函数，函数定义中，符号 “`==>`”前的”`b>0`” 表示这个性质的条件。在解释器下运行：

```
Main>quickCheck prop_ratplust_unit
+++ OK, passed 100 tests.
```

这表明，系统对100个随机生成的分数(a,b)检查该性质没有发现问题。但是，假如在`rat_plus` 定义中有错误，如

```
ratplus :: Fraction -> Fraction -> Fraction
ratplus (a,b) (c,d) = (a*c + b*c, b*d)
```

则测试有可能发现这个错误：

```
Main> quickCheck prop_ratplus_unitlaw
*** Failed! Falsifiable (after 1 test):
(1,1)
```

此时系统返回一个反例：(a,b) = (1,1), 因为根据定义，`(1,1)<+>(0, 1)`结果是(0, 1), 不等于(1,1)。

再比如，你可能想测试乘法对加法是否可分配：

```
prop_rattimes_plus_distr :: Fraction -> Fraction -> Fraction ->Property
prop_rattimes_plus_distr (a,b) (c,d) (e,f) =
     b > 0 && d > 0 && f > 0 ==>
(a,b) <-*-> ((c,d) <+> (e,f)) <==> ((a,b) <-*-> (c,d)) <+> ((a,b) <-*-> (e,f))
```

函数的性质名通常以prop_打头。

注意，使用QuickCheck的步骤：

1. 在你的函数定义模块开始输入Test.QuickCheck:

   ```
   import Test.QuickCheck
   ```

2. 定义函数的性质，如

   ```
   prop_ratplus_unit :: Fraction -> Property
   prop_ratplus_unit (a,b) =  b > 0 ==>(a, b) <+> (0,1) <==> (a, b)
   ```

3. 在解释器中运行命令quickCheck，后接你定义的性质函数名，如

   ```
   Main>quickCheck prop_ratplust_unit
   +++ OK, passed 100 tests.
   ```

三、递交实验要求

- 你提交的报告是包含程序和有关说明的文本文件，说明包括姓名、email、学号和系别。说明作为注释。你编写的模块形如

  ```
  -- 王力 201101001, wangli@163.com, 计算机系;
  -- 其他说明
  module MyFraction where
  import Test.QuickCheck
  ```

注意，请务必使用给定的模块名。

- 请通过elearning.sysu.edu.cn课程网页的作业系统提交。
- 你的程序应该能够正常运行，并说明做了哪些测试。如果尚不能运行，说明理由或者困难。
- 实验记入成绩，请认真对待。
- 切勿抄袭，后果非常严重!