Laboratorul 10

Exerciții pentru Semigroup și Monoid din HaskellBook

Vom face instanțe ale claselor Semigroup și Monoid și vom verifica ecuațiile asociate acestor structuri folosind QuickCheck.

Definim generic proprietățile de asociativitate și identitate.

```
import           Test.QuickCheck
import           Test.QuickCheck.Gen

semigroupAssoc :: (Eq m, Semigroup m) => m -> m -> m -> Bool
semigroupAssoc a b c = (a <> (b <> c)) == ((a <> b) <> c)

monoidLeftIdentity   :: (Eq m, Monoid m) => m -> Bool
monoidLeftIdentity a = (mempty <> a) == a

monoidRightIdentity   :: (Eq m, Monoid m) => m -> Bool
monoidRightIdentity a = (a <> mempty) == a
```

Completați definițiile lipsă în exercițiile de mai jos.

Pentru instanțele clasei `Arbitrary` citiți Cursul 7.

Exercițiul 1 - Trivial

```
data Trivial = Trivial
   deriving (Eq, Show)

instance Semigroup Trivial where
   _ <> _ = undefined

instance Monoid Trivial where
  mempty  = undefined

instance Arbitrary Trivial where
   arbitrary = undefined

type TrivAssoc = Trivial -> Trivial -> Trivial -> Bool
type TrivId    = Trivial -> Bool

   quickCheck (semigroupAssoc :: TrivAssoc)
   quickCheck (monoidLeftIdentity :: TrivId)
   quickCheck (monoidRightIdentity :: TrivId)
```

Exercițiul 2 - Conjuncții

```
newtype BoolConj = BoolConj Bool
   deriving (Eq, Show)
```

```
instance Semigroup BoolConj where
    BoolConj a <> BoolConj b = undefined

instance Monoid BoolConj where
    mempty = undefined

instance Arbitrary BoolConj where
   arbitrary = MkGen ( \s i -> BoolConj  ((unGen arbitrary) s i))

type BoolConjAssoc = BoolConj -> BoolConj -> BoolConj -> Bool
type BoolConjId    = BoolConj -> Bool

   quickCheck (semigroupAssoc :: BoolConjAssoc)
   quickCheck (monoidLeftIdentity :: BoolConjId)
   quickCheck (monoidRightIdentity :: BoolConjId)
```

Exercițiul 3 - Disjuncții

newtype BoolDisj = BoolDisj Bool deriving (Eq, Show)

```
 instance Semigroup BoolDisj where

 instance Monoid BoolDisj where

instance Arbitrary BoolDisj where
   arbitrary =

type BoolDisjAssoc = BoolDisj -> BoolDisj -> BoolDisj -> Bool
type BoolDisjId    = BoolDisj -> Bool

   quickCheck (semigroupAssoc :: BoolDisjAssoc)
   quickCheck (monoidLeftIdentity :: BoolDisjId)
   quickCheck (monoidRightIdentity :: BoolDisjId)
```

Exercițiul 4 - Identity

```
newtype Identity a = Identity a
  deriving (Eq, Show)

instance Semigroup a => Semigroup (Identity a) where

instance Monoid a => Monoid (Identity a) where

instance Arbitrary a => Arbitrary (Identity a) where
    arbitrary = undefined

type IdentityAssoc a = Identity a -> Identity a -> Identity a -> Bool
type IdentityId    a = Identity a -> Bool

 quickCheck (semigroupAssoc :: IdentityAssoc String)
 quickCheck (monoidLeftIdentity :: IdentityId [Int])
 quickCheck (monoidRightIdentity :: IdentityId [Int])
```

Exercițiul 5 - Pereche

```
data Two a b = Two a b
  deriving (Eq, Show)

instance (Semigroup a, Semigroup b) => Semigroup (Two a b) where
    Two x y <> Two z t = undefined

instance (Monoid a, Monoid b) => Monoid (Two a b) where
    mempty = undefined

instance (Arbitrary a, Arbitrary b) => Arbitrary (Two a b) where
   arbitrary = MkGen ( \s i -> Two ((unGen (arbitrary)) s i) ((unGen (arbitrary)) s i))

type TwoAssoc a b = Two a b -> Two a b -> Two a b -> Bool
type TwoId    a b = Two a b -> Bool

   quickCheck (semigroupAssoc :: TwoAssoc String [Int])
   quickCheck (monoidLeftIdentity :: TwoId [Int] String)
   quickCheck (monoidRightIdentity :: TwoId [Int] [Int])
```

Exercițiul 6 - Alternativă

```
data Or a b = Fst a | Snd b
  deriving (Eq, Show)
```

Faceți `Or a b` instanță a clasei `Semigroup` astfel încât operația de semigrup să fie definită astfel:

```
Prelude> Fst 1 <> Snd 2 Snd 2 Prelude> Fst 1 <> Fst 2 Fst 2
Prelude> Snd 1 <> Fst 2 Snd 1 Prelude> Snd 1 <> Snd 2 Snd 1>
```

```
instance  Semigroup (Or a b) where
   Fst _ <> x = undefined
   y     <> _ = undefined

instance (Arbitrary a, Arbitrary b) => Arbitrary (Or a b) where
 arbitrary = oneof [undefined, undefined]

type OrAssoc a b = Or a b -> Or a b -> Or a b -> Bool

  quickCheck (semigroupAssoc :: OrAssoc String Int)
  quickCheck (semigroupAssoc :: OrAssoc String [Int])
```