Homework 3 Answers

1. **Ex. 23**

```
(defproperty floor-returns-nats
  (divisor :value (+ 1 (random-natural))
   dividend :value (random-natural))
  (natp (floor dividend divisor)))
```

2. **Ex. 24**

```
(defproperty max-greater-or-equal-to-params
  (x :value (random-rational)
   y :value (random-rational))
  (let ((result (max x y)))
    (and (>= result x)
         (>= result y))))
```

3. **Ex. 25**

```
(defproperty exercise-6
  (x :value (random-number))
  (= (+ x x) (* 2 x)))

(defproperty exercise-7
  (x :value (random-number))
  (= (+ (* -1 x) (* 2 x) x) (* 2 x)))

(defproperty exercise-8
  (x :value (random-number)
   y :value (random-number)
   z :value (random-number))
  (= (+ (+ x (* -1 (+ x y)) z) y) z))
```

4. **Ex 26**

```
(defproperty modular-arithmetic-works-plus
  (x :value (random-natural)
   y :value (random-natural)
   m :value (+ 1 (random-natural)))
  (= (mod (+ x y) m)
     (mod (+ (mod x m) (mod y m)) m)))

(defproperty modular-arithmetic-works-times
  (x :value (random-natural)
   y :value (random-natural)
```

```
    m :value (+ 1 (random-natural)))
   (= (mod (* x y) m)
      (mod (* (mod x m) (mod y m)) m)))

(defproperty modular-arithmetic-works-crazy
  (w :value (random-natural)
   x :value (random-natural)
   y :value (random-natural)
   z :value (random-natural)
   m :value (+ 1 (random-natural)))
  (= (mod (- (* w (+ x y)) z) m)
     (mod (- (* (mod w m) (+ (mod x m) (mod y m)))
                 (mod z m)) m)))
```

5.
```
(defproperty append-preserves-elements
  (xs :value (random-list-of (random-atom))
   ys :value (random-list-of (random-atom))
   x  :value (random-atom))
  (let ((result (append xs ys)))
    (and (implies (or (member-equal x xs)
                      (member-equal x ys))
                  (member-equal x result))
         (implies (member-equal x result)
                  (or (member-equal x xs)
                      (member-equal x ys))))))
```