1. Aiken-Lang STD Library Functions Test  
     
   rational.ak

Results:  
pub fn mul(left: Rational, right: Rational) -> Rational {

let Rational { numerator: a\_n, denominator: a\_d } = left

let Rational { numerator: b\_n, denominator: b\_d } = right

Rational { numerator: a\_n \* b\_n, denominator: a\_d \* b\_d }

}

test mul\_1() {

mul(ratio(2, 3), ratio(3, 4)) == ratio(6, 12)

}

test mul\_2() {

mul(ratio(-2, 3), ratio(-3, 4)) == ratio(6, 12)

}

test mul\_3() {

let result =

ratio(2, 5)

|> mul(ratio(1, 8))

|> mul(ratio(3, 10))

|> mul(ratio(21, 100))

|> mul(ratio(3, 5))

|> mul(ratio(2, 8))

|> mul(ratio(4, 10))

|> mul(ratio(22, 100))

|> reduce

result == ratio(2079, 50000000)

}  


Tests:  
test geometric\_mean1() {

expect Some(x) = new(1, 2)

expect Some(y) = new(1, 2)

geometric\_mean(x, y) == new(1, 2)

}

test geometric\_mean2() {

expect Some(x) = new(-1, 2)

expect Some(y) = new(1, 2)

geometric\_mean(x, y) == None

}

test geometric\_mean3() {

expect Some(x) = new(1, 2)

expect Some(y) = new(-1, 2)

geometric\_mean(x, y) == None

}

test geometric\_mean4() {

expect Some(x) = new(1, 3)

expect Some(y) = new(1, 6)

geometric\_mean(x, y) == new(1, 4)

}

test geometric\_mean5() {

expect Some(x) = new(67, 2500)

expect Some(y) = new(35331, 1000)

expect Some(yi) = reciprocal(y)

geometric\_mean(x, yi) == new(258, 9398)

}  
