CS 2510 Exam 6 – Summer 2012

Name: ______________________

Student Id (last 4 digits): ______________________

• Write down the answers in the space provided.
• You may use all syntax of Java that we have studied in class.
• For tests you only need to provide the expression that computes the actual value, connecting it with an arrow to the expected value. For example `s.method() -> true` is sufficient.
• Remember that the phrase "design a class" or "design a method" means more than just providing a definition. It means to design them according to the **design recipe**. You are *not* required to provide a method template unless the problem specifically asks for one. However, be prepared to struggle if you choose to skip the template step.

Score		45

Good luck!

A useful utility

You may assume, **for the purposes of examples only**, the existence of a class `Iter<X>` that implements `Iterable<X>`. It has a somewhat magical constructor in that it can take *any* number of `X`s as arguments. Its `iterator` method produces an iterator that iterates through those arguments.

For example:

```
new Iter<Integer>().iterator().hasNext()  --> false
new Iter<Integer>(1).iterator().hasNext() --> true
new Iter<Integer>(1).iterator().next()    --> 1
Iterator<Integer> i =
  new Iter<Integer>(1,2,3).iterator();
i.next() --> 1
i.next() --> 2
i.next() --> 3
```

$\boxed{\text{45 Points}}$

Problem 1

Design a method

```
    Double avg(Iterable<Double> nums)
```

that computes the average of a series of numbers. It should raise an exception if the series is empty.

Problem 2

Design a class

```
class SumSquares implements Iterable<Integer>
```

which is given an `Iterable<Integer>` when constructed and whose `iterator` method produces an iterator that produces the sum of the squares the given iterator produces.

For example, if given a `new Nat()`, the iterable you wrote in lab that produces the natural numbers `0, 1, 2, 3, 4`, ..., then iterator should produce `0, 1, 5, 14, 30`, etc.

Problem 3

Design a class

```
class Zip<X,Y> implements Iterable<Pair<X,Y>>
```

which is given an `Iterable<X>` and an `Iterable<Y>` when constructed and whose `iterator` method produces an iterator that produces pairs of values: one from the `X` series, one from the `Y` series. The iterator should have no more elements whenever either of the given iterators have no more elements.

You may rely on the following definition of `Pair`:

```
class Pair<X,Y> {
    X left;
    Y right;
    Pair(X left, Y right) {
        this.left = left;
        this.right = right;
    }
}
```