CS 2510 Exam 5 (Alternate) – 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!

45 Points

Problem 1

Develop an implementation of `Comparator<Square>` that orders squares by their area in such a way that the smaller the area, the *greater* the square is considered. So for example, there is no square greater than the one with width = 0 and height = 0.

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

```
// Represents a square with given height and width.
class Square {
  Integer width;
  Integer height;
  Square(Integer width, Integer height) {
    this.width = width;
    this.height = height;
  }
}
```

Problem 2

Design a method:

```
<T> Boolean noSameNeighbors(ArrayList<T> ls, Comparator<T> c)
```

that determines if the given array list has no adjacent elements (i.e. neighboring elements) that are considered of equal size according to the given comparator.

Problem 3

Design a method:

```
<A> Comparator<ArrayList<A>> dict(Comparator<A> ca)
```

that consumes a comparator for `A`s and and produces a comparator for array lists of `A`s that is the *dictionary order* (aka *alphabetic order*).

In other words, given an ordering on individual elements, produce an ordering on *lists of* elements that corresponds to the ordering used in dictionaries. A dictionary uses an ordering on individual letters (single elements) to make an ordering on strings of letters (lists of elements) so that "abc" $<$ "abcd", "abc" $<$ "adc", "abc" $>$ "ab", and "abc" $=$ "abc", etc.