CS 201, Fall 2014 — Dec 12th — Final Exam Name: ____________________

Note: In questions where you are asked about a static method, assume that the method is in a class called `Q`*n* where *n* is the question number, e.g., `Q1` for Question 1.

Question 1. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q1 {
  public static void mystery(
      String s) {
    s = "Hey there!";
  }
```

```
  public static void main(
      String[] args) {
    String q = "Yup";
    mystery(q);
    System.out.println(q);
  }
}
```

Question 2. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q2 {
  private int x;

  public Q2(int i) { x = i; }

  public int getX() { return x; }

  public static void f(Q2 obj) {
    obj.x = 42;
  }
```

```
  public static void main(
      String[] args) {
    Q2 q = new Q2(17);
    int y = q.getX();
    f(q);
    System.out.printf("%d,%d\n",
      q.x, y);
  }
}
```

Question 3. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q3 {
  public static void mystery(
      int[] a, int[] b) {
    int[] tmp = a;
    a = b;
    b = tmp;
  }
```

```
  public static void main(
      String[] args) {
    int[] x = { 1, 2 };
    int[] y = { 3, 4 };
    mystery(x, y);
    System.out.printf("%d,%d,%d,%d\n",
      x[0], x[1], y[0], y[1]);
  }
}
```

Question 4. [5 points] Consider the following code snippet:

```
Foo f = new Bar();
```

What must be true about the data types `Foo` and `Bar` for this code to be legal?

Question 5. [10 points] Consider the following method:

```
public static int countVowels(String fileName) throws IOException {
  FileReader reader = new FileReader(fileName);

  int count = 0;
  boolean done = false;
  while (!done) {
    int c = reader.read();
    if (c < 0) {
      done = true;
    } else if (Character.isLetter((char) c)) {
      char lower = Character.toLowerCase((char) c);
      if (lower == ’a’ || lower == ’e’ || lower == ’i’
           || lower == ’o’ || lower == ’u’) {
        count++;
      }
    }
  }

  reader.close();

  return count;
}
```

(a) Explain how this method could successfully open the named file, but not make any attempt to close it.

(b) Explain how to modify the method so that if the file is opened, the method is guaranteed to make an attempt to close it. (You can annotate the code above.)

Question 6. [10 points] For each of the following code fragments (a)–(d), state a big-O upper bound on the running time, with the problem size n being the value of the variable `n`. Briefly explain each bound.

(a)
```
int sum = 0;
for (int j = 0; j < n; j++) {
    for (int i = 0; i < n; i++) {
        sum++;
    }
}
```

(b)
```
int sum = 0;
for (int j = 0; j < n; j++) {
    int max = j;
    if (max > 100) { max = 100; }
    for (int i = 0; i < max; i++) {
        sum++;
    }
}
```

(c)
```
int sum = 0;
for (int j = 0; j < n*n; j++) {
    for (int i = 0; i < j; i++) {
        sum++;
    }
}
```

(d)
```
int sum = 0;
for (int j = 0; j < n; j++) {
    for (int i = 0; i < n; i = i*2) {
        sum++;
    }
}
```

Question 7. [10 points] Complete the following method. It should remove all elements of the given `List` that compare as greater than the `val` parameter according to the comparator object given as the `comp` parameter. Note that the method *should* modify `list` (by removing elements).

```
public static<E>
void removeAllGreaterThan(List<E> list, E val, Comparator<E> comp) {
```

Question 8. [5 points] Consider the following method:

```
public static int countDistinct(List<Integer> list, Comparator<Integer> comp) {
    TreeSet<Integer> set = new TreeSet<Integer>(comp);
    set.addAll(list);
    return set.size();
}
```

The idea is that the method counts the number of *distinct* elements in the given `List<Integer>` as compared by `comp`, the `Comparator<Integer>` parameter. In otherwords, if two elements in the list compare as equal to each other according to the comparator, they are not considered distinct.

Complete the implementation of `MagnitudeComparator`, which implements `Comparator<Integer>`. It compares integers by their magnitude, ignoring the sign. For example, 4 and -4 should compare as equal, and 5 should compare as less than -6.

Example JUnit test case:

```
MagnitudeComparator comp = new MagnitudeComparator();
List<Integer> a = Arrays.asList(10, -10, 15);
List<Integer> b = Arrays.asList(-19,11,-5,2,-10,-5,11,5,-18,-11,16,-6);

assertEquals(2, Q8.countDistinct(a, comp));
assertEquals(8, Q8.countDistinct(b, comp));
```

Note that in the test case above, `a` contains integers with 2 distinct magnitudes, and `b` contains integers with 8 distinct magnitudes.

```
public class MagnitudeComparator implements Comparator<Integer> {
  public int compare(Integer left, Integer right) {
```

Programming Questions

First things first: the following web page specifies which resources you may use during the exam, and links to the permitted resources:

```
http://ycpcs.github.io/cs201-fall2014/assign/final.html
```

Start by downloading the exam zipfile using the web browser. (You will be given the URL of the zipfile when the exam starts.)

Import the zipfile as an Eclipse project. You should see a project called **CS201_Final** in your workspace.

There are three classes called **Q9**, **Q10**, **Q11**. Each contains a static method to implement, a point value, along with a detailed comment describing what the method should do and how many points the method is worth. There are corresponding JUnit tests for each method (**Q9Test**, etc.)

Requirements:

- Your implementations of the methods for **Q9** and **Q10** must be recursive

Some general hints:

- If `s` is a string, `s.length()` returns how many characters `s` has
- If `s` is a string and `i` is an integer, `s.charAt(i)` the character at index `i` in `s`
- If `s` is a string and `i` is an integer, `s.substring(i)` returns a string containing all characters from `s` starting at index `i`
- If `s` is a string and `i` and `j` are integers, `s.substring(i, j)` returns a string containing all of the characters in `s` from index `i` (inclusive) to index `j` (exclusive)
- If `s` and `w` are strings, `s.startsWith(w)` returns true if `s` starts with the characters of `w`; for example, `"hello world".startsWith("hello")` returns true, `"hello world".startsWith("world")` returns false
- The `keySet` method returns a map's set of keys (and you can iterate through it using a foreach loop or an explicit iterator)
- The `get` method retrieves the value associated with a particular key in a map

When you are done, submit your code using the blue up arrow icon (Submit project) in Eclipse. Enter your Marmoset username and password when prompted.

Have fun!