CS 201, Fall 2014 — Dec 12th — Final Exam Name: Rest of solution

Note: In questions where you are asked about a static method, assume that the method is in a class called `Qn` 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);
  }
}
```

Yup

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);
  }
}
```

17 42

17,42

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]);
  }
}
```

this just swaps what a and b refer to, but doesn't change the contents of the arrays, and doesn't affect x/y

1, 2, 3, 4

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?

Bar must be a subtype of Foo

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

```
public static int countVowels(String fileName) throws IOException {
  FileReader reader = new FileReader(fileName);
try {
  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++;
      }
    }
  }
} finally {
  reader.close();
}
  return count;
}
```

IOException could be thrown

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

An IOException could be thrown from the call to reader.read()

(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.)

Use try/finally (See above)

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) {
        if (left < 0) {
            left = -left;
        }
        if (right < 0) {
            right = -right;
        }
        return left.compareTo(right);
  }
}
```