CS 201, Fall 2013 — Dec 13th — 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 f(int[] a) {
    a = new int[1];
    a[0] = 42;
  }
```

```
  public static void main(
      String[] args) {
    int[] b = new int[1];

    b[0] = 17;
    f(b);
    System.out.printf("%d\n", b[0]);
  }
}
```

Question 2. [5 points] Consider the following class and static method:

```
public class Animal {
  public String sound() {
    return "unknown";
  }
}
```

```
public class Q2 {
  public static void mystery(Animal a) {
    System.out.println(a.sound());
  }
}
```

Is it possible to predict with certainty what output will be printed by the `mystery` method when it is called? If so, what is the output? If not, why not? Explain briefly.

Hint: Notice that the code does not contain any specific call to the `mystery` method.

Question 3. [10 points] Specify code that can be used at the points labeled `Missing code 1` and `Missing code 2` to allow the following program to compile and, when the `main` method in the `ColorItem` is executed, produce the output `Apples,3,red`.

```
public class Item {
  private String name;
  private int quantity;

  public Item(String n, int q) {
    name = n;
    quantity = q;
  }

  public String toString() {
    return name + "," + quantity;
  }
}
```

```
public class ColorItem extends Item {
  private String color;

  public ColorItem(String n,
      int q, String c) {
    Missing code 1
  }

  public String toString() {
    Missing code 2
  }

  public static void main(
      String[] args) {
    ColorItem ci = new ColorItem(
      "Apples", 3, "red");
    System.out.println(ci.toString());
  }
}
```

Question 4. [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 i = 0; i < n; i++) {
  for(int j = 0; j < n*n; j++) {
    sum++;
  }
}
```

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

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

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

Question 5. [10 points]

(a) Consider the following program (which begins on the left and continues on the right):

```
public class Q5 {
  public static void mystery(
      ArrayList<Integer> a) {

    int n = a.size() / 2;
    while (n > 0) {
      Integer x = a.remove(0);
      a.add(x);
      n--;
    }
  }
```

```
  public static void main(
      String[] args) {
    ArrayList<Integer> nums =
      new ArrayList<Integer>();

    for (int i = 0; i < 6; i++) {
      nums.add(i);
    }

    mystery(nums);

    for (Integer val : nums) {
      System.out.println(val);
    }
  }
}
```

What output is printed when the program is run?

(b) State a big-O upper bound on the running time of the `mystery` method in part (a). Let the problem size N be the number of elements in the `ArrayList` passed as the parameter. Explain briefly.

Question 6. [5 points] What output is printed by the following code (which begins on the left and continues on the right)?

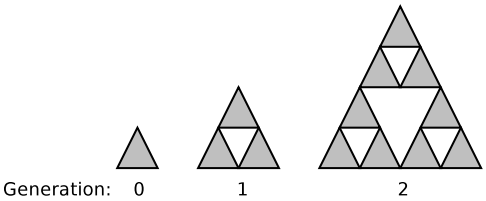
```
public class Q6 {
  public static List<Integer>
    mystery(List<Integer> a) {

    TreeSet<Integer> s =
      new TreeSet<Integer>();
    s.addAll(a);

    List<Integer> result =
      new ArrayList<Integer>();
    for (Integer val : s) {
      result.add(val);
    }

    return result;
  }
```

```
  public static void main(
      String[] args) {
    List<Integer> x =
      new ArrayList<Integer>();

    x.add(9);
    x.add(5);
    x.add(8);
    x.add(1);
    x.add(3);
    x.add(5);

    List<Integer> y = mystery(x);
    for (Integer val : y) {
      System.out.println(val);
    }
  }
}
```

Question 7. [5 points] Briefly explain why the following code will not compile, and how to fix the problem so that the code does compile (and correctly count the number of lines in the file whose filename is passed as the parameter):

```
public static int countLines(String fileName) {
  FileReader fr = new FileReader(fileName);
  try {
    BufferedReader br = new BufferedReader(fr);
    int count = 0;
    while (br.readLine() != null) { count++; }
    return count;
  } finally {
    fr.close();
  }
}
```

Question 8. [5 points] Consider a fractal constructed in the following way. The first generation (0) of the fractal is an isosceles triangle whose base and height are each equal to 2. Each succeeding generation is constructed by connecting connecting three copies of the previous generation as shown below:



Let $f(n)$ be the total area of all of the small (shaded) triangles in a generation n fractal. Our induction hypothesis, *IH*(n), is that for all $n \geq 0$, $f(n) = 2 \cdot 3^n$.

Basis step. Prove that *IH*(0) is true by showing that $f(0) = 2$. (The area of a triangle is $\frac{1}{2}bh$.)

Expectation. State *IH*$(n+1)$: $f(n+1) =$

Recurrence. Define $f(n+1)$ in terms of $f(n)$. In other words, how does the total area of the shaded triangles increase from one generation to the next? Explain briefly.

Induction step. Show that if IH(n) is true, then *IH*$(n+1)$ must also be true. Expand the occurrence of $f(n)$ in your recurrence, and show that the resulting equation can be rewritten to exactly match the expectation.

Programming Questions

To get started, use a web browser to download the zipfile as specified by your instructor. Import it as an Eclipse project using File → Import... → General → Existing Projects into Workspace → Archive file.

You should see a project called **CS201_Final**.

Important: You may use the following resources:

- The lecture notes posted on the course web page
- Your previous labs and assignments
- The Java API documentation at `http://docs.oracle.com/javase/7/docs/api/`

When you finish, use the blue up arrow icon to upload your work to Marmoset.

Question 9. [15 points] Complete the definition of the `Cell` class. An instance of `Cell` represents a cell in a one-dimensional cellular automaton, and is either alive or dead. The constructor takes a boolean value indicating whether the `Cell` object being initialized is alive or dead. The `isAlive` method returns a boolean value indicating whether the cell is alive or dead.

The `computeNext` method returns a new `Cell` that reflects whether the current cell should be alive or dead in the next generation of cells. It implements Wolfram's Rule 110 (you may recall this from CS 101.) It takes two parameters, references to `Cell` objects representing the left and right neighbors. The `computeNext` method should return a cell that is dead if either:

- The left neighbor, current cell, and right neighbor are all dead
- The left neighbor is alive, and the current cell and right neighbor are both dead
- The left neighbor, current cell, and right neighbor are all alive

Otherwise, `computeNext` should return a cell that is alive.

The `CellTest` JUnit test class has unit tests.

If the `Cell` class is implemented correctly and you run the `Rule110` program, you should see the output shown on the right.

Expected output of `Rule110`:

```
..........*.
.........**.
........***.
.......**.*.
......*****.
.....**...*.
....***..**.
...**.*.***.
..*******.*.
.**.....***.
```

Question 10. [15 points] In the class `Q10`, complete the static method `hexToInt`. It takes a string `s`, which is a number in *hexidecimal* format. Hexidecimal is a base-16 representation of an integer value, where the letters A through F represent hexidecimal "digits" with values 10–15. For example, the hexidecimal number 2C3 corresponds to the decimal integer 707, because

$$2 \cdot 16^2 + 12 \cdot 16^1 + 3 \cdot 16^0 = 707$$

(Note that the value hexidecimal digit C is 12.)

Important: Your method must be recursive. Do not use a loop.

The JUnit test class `Q10Test` contains unit tests.

You can use the `Character.isDigit(char)` method to determine whether or not a character is a (decimal) digit (i.e., 0 through 9). If `c` is a decimal digit, then `c - '0'` is the value of that digit. If `c` is a letter A through F, then `10 + (c - 'A')` is the value of that hexidecimal digit.

The `String` class's `substring` method will be useful. It takes two integers, the indices of the first (inclusive) and last (exclusive) characters of the substring you would like to retrieve.

Hints: think about what base case or cases are appropriate. Make sure your recursive call or calls are on subproblem(s) that work towards a base case.

Question 11. [15 points] In the class `Q11`, complete the definition of the static method `hasDuplicates`. It takes a `List` of elements of type `E` (where `E` is a type that implements the `Comparable` interface), and returns true if the list contains any duplicate values, false if it does not contain any duplicate values.

Important: your method **must** execute in $O(N \log N)$ running time, where N is the number of elements in the list.

Hint: what kind of collection is useful for detecting duplicates?

Unit tests are provided in the `Q11Test` JUnit test class.