CS 201, Fall 2014 — Nov 19th — Exam 2 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. [10 points] Consider the following code:

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

State a big-O upper bound on the running time of this code, using n (the value of the variable `n`) as the problem size. Briefly explain your answer.

Question 2. [10 points] Consider the following code:

```
int count = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i/2; j++) {
        count++;
    }
}
```

State a big-O upper bound on the running time of this code, using n (the value of the variable `n`) as the problem size. Briefly explain your answer.

Question 3. [10 points] Consider the following method:

```
public static<E> void removeEveryOther(List<E> list) {
    Iterator<E> i = list.iterator();
    int count = 0;
    while (i.hasNext()) {
        i.next();
        if (count % 2 == 1) {
            i.remove();
        }
        count++;
    }
}
```

Note that an `Iterator`'s `remove` method removes from the collection the last element returned by the `Iterator`'s `next` method.

(a) State a big-O upper bound on the running time of this method if `list` is an `ArrayList`. Consider the problem size N to be the number of list elements. Explain briefly.

(b) State a big-O upper bound on the running time of this method if `list` is a `LinkedList`. Consider the problem size N to be the number of list elements. Explain briefly.

Question 4. [10 points] Consider the following method:

```
public static void mystery(List<String> list) {
    Stack<String> stack = new Stack<String>();
    Queue<String> queue = new LinkedList<String>();

    Iterator<String> i = list.iterator();
    while (i.hasNext()) {
        stack.push(i.next());
        if (i.hasNext()) {
            queue.add(i.next());
        }
    }

    while (!stack.isEmpty()) {
        System.out.println(stack.pop());
    }

    while (!queue.isEmpty()) {
        System.out.println(queue.remove());
    }
}
```

What output is printed by the following code?

```
List<String> coll = Arrays.asList("A", "B", "C", "D", "E", "F");
Q4.mystery(coll);
```

Question 5. [10 points] Complete the `containsDuplicates` method below. It takes a `Collection` of elements of type `E` as a parameter, and returns true if the collection contains any duplicate elements, or false if the collection does not contain any duplicate elements. You can assume that the type `E` implements `Comparable<E>`.

Requirement: The method should complete in $O(N \log N)$ running time (or $O(N)$ running time), where N is the number of elements in the collection.

Here are some JUnit tests showing the expected behavior of the method:

```
List<String> listA = Arrays.asList("A", "B", "C", "A", "D");
List<String> listB = Arrays.asList("P", "V", "Z", "Y");

assertTrue(Q5.containsDuplicates(listA));
assertFalse(Q5.containsDuplicates(listB));
```

Hint: What kind of collection is useful for detecting duplicate values?

```
public static<E extends Comparable<E>>
boolean containsDuplicates(Collection<E> coll) {
```

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_Exam2**.

Important: You may use **only** the following information 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/`
- Any written or printed notes that you brought with you

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

Question 6. [25 points] Complete the `replaceIt` static method in the class `Q6`. It takes three parameters: a string `s`, a character `c`, and a replacement string `r`. It should return a string in which each occurrence of `c` in `s` is replaced by `r`.

Requirement: Your method **must** use recursion.

Example JUnit tests:

```
assertEquals("Om NOM NOM NOM", Q6.replaceIt("Om X X X", 'X', "NOM"));
assertEquals("Feed me some turnip, please",
    Q6.replaceIt("Feed me some @, please", '@', "turnip"));
```

A more complete set of JUnit tests can be found in `Q6Test`. Make sure that all of these tests pass when you run them.

Hints:

- Make sure you have an appropriate base case.
- Think about how to find a subproblem that can be solved recursively.
- If `x` is a `String`, then `x.substring(1)` returns a string containing all characters of `x` except for the first character
- If `x` is a `String` and `i` is an `int`, then `x.charAt(i)` returns the character at index `i` in `x`
- The `+` operator performs string concatenation when at least one of the operands is a string

Question 7. [25 points] Complete the `tally` static method in the `Q7` class. It takes two parameters: `prices`, which is a `Map<String, Integer>`, and order, which is a `List<String>`, and it returns an `int` value as follows. The `prices` map associates the names of food items with prices in dollars. The `order` list contains a list of food items. The method should determine the sum of the prices of all of the food items in `order` and return it. Note that an item may appear in the list multiple times, and when this happens the `tally` method should count each occurrence towards the total.

As a special case, if `order` contains a food item that isn't specified in `prices`, then the method should throw an `IllegalArgumentException`.

A set of JUnit tests can be found in `Q7Test`. Make sure that all of these tests pass when you run them.

Hints:

- The `containsKey` method is useful for checking whether a map contains a particular key
- The `get` method retrieves the value associated with a particular key in a map