CS 201, Summer 2014 — Aug 11th — 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++) {
    for (int j = 0; j < n; j++) {
        count++;
    }
}
```

State a big-O upper bound for 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 * n; i++) {
    for (int j = 0; j <= i; j++) {
        count++;
    }
}
```

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

Question 3. [10 points] Complete the definition of the `CaseInsensitiveCharacterComparator` class. Its behavior is shown by the following JUnit test code:

```
Character[] letters = { 'H', 'a', 'E', 'k', 'D' };
Arrays.sort(letters, new CaseInsensitiveCharacterComparator());

assertEquals((Character)'a', letters[0]);
assertEquals((Character)'D', letters[1]);
assertEquals((Character)'E', letters[2]);
assertEquals((Character)'H', letters[3]);
assertEquals((Character)'k', letters[4]);
```

Hint: You can use the `Character.toLowerCase` method to convert a character value to an equivalent lower case character value. E.g., `Character.toLowerCase('A')` would return `'a'`.

```
public class CaseInsensitiveCharacterComparator implements Comparator<Character> {
    public int compare(Character left, Character right) {
```

Question 4. [5 points] Consider the following method:

```
public static int countEvens(LinkedList<Integer> list) {
    int count = 0;
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i) % 2 == 0) { count++; }
    }
    return count;
}
```

State a big-O upper bound for this method, where the problem size N is the number of elements in the `list` parameter. Briefly explain your bound.

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

```
public static int countEvens(LinkedList<Integer> list) {
    int count = 0;
    for (Iterator<Integer> i = list.iterator(); i.hasNext(); ) {
        Integer value = i.next();
        if (value % 2 == 0) { count++; }
    }
    return count;
}
```

State a big-O upper bound for this method, where the problem size N is the number of elements in the `list` parameter. Briefly explain your bound.

Question 6. [10 points] Consider the following static method:

```
public static<E extends Comparable<E>> List<E> mystery(List<E> src) {
    Set<E> set = new TreeSet<E>();
    for (E elt : src) { set.add(elt); }

    List<E> result = new ArrayList<E>();
    for (E elt : set) { result.add(elt); }

    return result;
}
```

(a) State a big-O upper bound on the running time of this method. Assume that the problem size N is the number of elements in the list parameter `src`. Briefly explain your answer.

(b) What output is printed by the following code?

```
List<Integer> myList = new ArrayList<Integer>();
myList.add(9);
myList.add(0);
myList.add(1);
myList.add(2);
myList.add(5);

List<Integer> result = Q6.mystery(myList);
for (Integer x : result) { System.out.print(x + " "); }
```

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 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 7. [10 points] Use recursion to complete the `sumOfDigits` method in the `Q7` class. It takes an integer value as a parameter, and returns the sum of the digits in the decimal representation of that integer. You may assume that the parameter will be non-negative.

Examples:

- `sumOfDigits(0)` returns 0
- `sumOfDigits(1)` returns 1
- `sumOfDigits(12)` returns 3
- `sumOfDigits(808)` returns 16
- `sumOfDigits(90125)` returns 17

There are tests in the `Q7Test` JUnit test class.

Important: Your solution *must* be recursive. Do not use a loop!

Hints:

- Think about an appropriate base case. All recursive methods should check to see if a base case has been reached before doing anything else.
- If n is an int, `n%10` computes the value of the rightmost digit of n.
- If n is an int, `n/10` computes the value the integer that contains all of the digits of n except the rightmost digit.

Question 8. [10 points] Use recursion to complete the `intToString` method in the `Q8` class. This method takes an `int` parameter (which you may assume will be non-negative) and returns the string that is the decimal representation of the integer.

Examples:

- `intToString(0)` returns `"0"`
- `intToString(4)` returns `"4"`
- `intToString(1331)` returns `"1331"`
- `intToString(8675309)` returns `"8675309"`

There are tests in the `Q8Test` JUnit test class.

Important: Your solution *must* be recursive. Do not use a loop!

Hints:

- Think about an appropriate base case. All recursive methods should check to see if a base case has been reached before doing anything else.
- If n is an int, `n%10` computes the value of the rightmost digit of n.
- If n is an int, `n/10` computes the value the integer that contains all of the digits of n except the rightmost digit.
- If `n` is an integer less than 10 — *i.e.*, a one digit number — then `((char)(n+'0'))` is the digit character that represents `n`.

Question 9. [20 points] This question has two parts.

(a) Complete the definition of the `LineItem` class. A `LineItem` object represents the name of an item (`String`), a quantity (`int`), and a unit price (`double`).

You should remove the lines that read

```
throw new UnsupportedOperationException("TODO - implement");
```

There are unit tests in the `LineItemTest` JUnit test class.

(b) Complete the `main` method in the `Q9` class. When run, the program should read lines of text similar to

```
apples,15,0.40
oranges,5,0.80
papayas,12,1.25
```

The data in each line of text should be converted to a `LineItem` object. If `line` is a string that contains the text of a line of input, you can split it into three parts using the code

```
String[] parts = line.split(",");
```

The `parts` array will have three elements.

You can convert the second and third parts of each line into `int` and `double` values using (respectively) the `Integer.parseInt` and `Double.parseDouble` methods. For example:

```
int quantity = Integer.parseInt(parts[1]);
```

The program should store all of the `LineItem` objects in a collection.

The main loop ends when the user types a line containing `quit`. When this happens, the program should print out the information for the least expensive and most expensive line items. The cost of a line item is the quantity times the unit price. Example run (user input in **bold**):

```
apples,15,0.40
oranges,5,0.80
papayas,12,1.25
quit
Least expensive item: oranges, total=4.0
Most expensive item: papayas, total=15.0
```