Quiz 2 (November 22, 2013)

Your name: ____________________________

Your Athena username: ____________________________

You have 50 minutes to complete this quiz. It contains 7 pages (including this page) for a total of 100 points.
The quiz is closed-book and closed-notes, but you are allowed one two-sided page of notes.
Please check your copy to make sure that it is complete before you start. Turn in all pages, together, when you finish. Before you begin, write your name on the top of every page.
Please write neatly. No credit will be given if we cannot read what you write.
For questions which require you to choose your answer(s) from a list, do so clearly and unambiguously by circling the number(s) or entire answer(s). Do not use check marks, underlines, or other annotations – they will not be graded.
Good luck!

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Grade</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Short Answer</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: Map/Filter/Reduce</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: Deadlock</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: Interleaving</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5: Thread Safety</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (Short Answer) (21 points).
Choose all correct answers for the following questions.

(a) The Java code `class MyWindow extends JFrame {...}` demonstrates the concept(s) of:

A. inheritance  
B. composition  
C. delegation  
D. forwarding  
E. locking  
F. subclassing

(b) The Java code `synchronized (this) { this.accountBalance += depositAmount; }` demonstrates the concept(s) of:

A. shared memory  
B. message passing  
C. locking  
D. thread safety  
E. listeners

(c) The Java code `assert(checkRep())` demonstrates the concept(s) of:

A. synchronization  
B. abstract data type  
C. rep invariant  
D. behavioral equality  
E. failing fast

(d) The Java code `private List<Person> committee` demonstrates the concept(s) of:

A. representation independence  
B. information hiding  
C. static typing  
D. view tree  
E. abstract data type
Problem 2 (Map/Filter/Reduce) (28 points).
You are given a Python interpreter that has a broken len() function – it always throws exceptions, no matter what you pass it. But you’ve been able to verify that map(), reduce() and filter() functions are working. Recall that the method signatures for these functions are as follows:

- map(function, list)
- reduce(function, list, initialValue)
- filter(predicate, list)

(a) Use map, filter, and reduce to implement a working version of len(), the function that counts the number of elements in a list. Your solution must define the functions m, f, and r and the value init so that the len() implementation below is correct.

```python
def len(list):
    def f(__________________):
        return ____________________
    def m(__________________):
        return ____________________
    def r(__________________):
        return ____________________
    init = _______________________
    return reduce(r, map(m, filter(f, list)), init)
```

(b) Also broken is count(list, e), a function that counts the number of times that e occurs in the list. Write a working version of count.

```python
def count(list, e):
    def f(__________________):
        return ____________________
    def m(__________________):
        return ____________________
    def r(__________________):
        return ____________________
    init = _______________________
    return reduce(r, map(m, filter(f, list)), init)
```
Problem 3 (Deadlock) (16 points).
In recitation, we looked at the Wizard class. A portion of its code is shown below. Recall that `Set.add()` returns true if and only if the element wasn’t found in the set and was successfully added. Similarly, `Set.remove()` returns true if and only if the element was found and then was successfully removed.

```java
public class Wizard {
    private final String name;
    private final Set<Wizard> friends;
    // Rep invariant:
    // name, friends != null
    // friend links are bidirectional: for all f in friends, f.friends contains this

    public boolean isFriendsWith(Wizard that) {
        return this.friends.contains(that);
    }

    public synchronized void friend(Wizard that) {
        if (this.friends.add(that)) {
            that.friend(this);
        }
    }

    public synchronized void defriend(Wizard that) {
        if (this.friends.remove(that)) {
            that.defriend(this);
        }
    }
}
```

(a) Which of the following may cause deadlock? (Assume the rep invariant is true before each of these examples starts.) Circle all correct answers.

A. Thread A calls `snape.friend(harry)`, while Thread B calls `harry.friend(snape)`
B. Thread A calls `snape.friend(harry)`, while Thread B calls `snape.friend(harry)`
C. Thread A calls `snape.defriend(harry)` and then `harry.friend(snape)`, while Thread B calls `snape.friend(harry)`.
D. Thread A calls `snape.friend(harry)`, while Thread B calls `harry.defriend(snape)`

(b) Which of the following would fix this deadlock problem? Circle all correct answers.

A. Lock on `Wizard.class` rather than the `Wizard` instance.
B. Make the `Set` a `synchronizedSet`, while keeping the `synchronized` keywords in the friend and unfriend methods.
C. Assign a unique integer `id` to every `Wizard`, and when friending or unfriending two wizards, always acquire the lock of the wizard with the smaller `id` first.
D. Combine the `friend` and `defriend` methods into a single method:
```java
public synchronized void friendOrDefriend(Wizard that, boolean friend) {
    if (friend==true) this.method1();
    else this.method2();
}
```
Problem 4 (Interleaving) (20 points).
Consider the following class:

```java
public class Magic {
    private static int x = 1;

    public static int magicMethod() {
        Thread t1 = new Thread(new Runnable() {
            public void run() {
                x *= 2;
                x *= 3;
            }
        });

        Thread t2 = new Thread(new Runnable() {
            public void run() {
                x *= 5;
                x *= 11;
            }
        });

        t1.start();
        t2.start();

        t1.join(); // join() means wait until the thread finishes its run() method
        t2.join();

        return x;
    }
}
```

Assume that all threads are running on the same processor, interleaving execution with each other. Which of the following are possible return values of `magicMethod`? Circle all correct answers.

- 330
- 30
- 6
- 2
- 22
Problem 5 (Thread Safety) (15 points).
Suppose you are reviewing this code for thread safety:

```java
public class C {
    public static final String[] x = new String[] { "abc" };  
    private final int y = 0;
    ...
    public synchronized double f() {
        double z = 0;
        ...
    }
    ...
}
```

Which of the following statements would be true and appropriate for an argument either in favor of or against the thread safety of this code? Circle all correct answers.

(a) For x:
A. x is thread-confined.
B. x is immutable.
C. x is protected by a lock.
D. x is global.
E. x can be involved in a race condition.

(b) For y:
A. y is thread-confined.
B. y is immutable.
C. y is protected by a lock.
D. y is global.
E. y can be involved in a race condition.

(c) For z:
A. z is thread-confined.
B. z is immutable.
C. z is protected by a lock.
D. z is global.
E. z can be involved in a race condition.