CMSC330 Fall 2009 Practice 7 Problems (Java 1.4)

1. Multithreading, Data Races, and Deadlock
    a. For the following program, give two schedules under which the final value of i differs in the two schedules. Give the schedule as a list of line numbers, and in each case, also give the final value of i.

```
l = new Object();
m = new Object();
i = 0;

Thread 1
1. synchronized(l) {
2.   i = 3;
3. }

Thread 2
4. synchronized (m) {
5.   i = i + 1;
6. }
```

b. For the following program, give one schedule under which there will be a deadlock, and give one schedule under which there will not be a deadlock. Give the schedule as a list of line numbers.

```
l = new Object();
m = new Object();
n = new Object();

Thread 1
1. synchronized (l) {
2.   synchronized (m) {
3.   }
4. }

Thread 2
5. synchronized (m) {
6.   synchronized (n) {
7.   }
8. }

Thread 3
9.   synchronized (n) {
10.   synchronized (l) {
11.   }
12. }
```

c. Using language notation similar to above, write a program that has no data races, but nonetheless may produce different output under different schedules.

d. List all possible outputs from the following program. Indicate next to each possible output whether all threads complete at the end, or, if they do not, which threads remain blocked.

```
l = new Object();

Thread 1
synchronized (l) {
  System.out.print("a ");
  l.wait();
  System.out.print("b ");
}

Thread 2
synchronized (l) {
  System.out.print("c ");
  l.notifyAll();
  System.out.print("d ");
}
```

2. Multithreading Code

a. Using Java Conditions, you must implement a synchronization construct called MyBarrier. A MyBarrier object is created with a certain value n. When a thread calls the method enter( ), it enters the barrier and blocks until a total of n threads have entered the barrier. When the n^{th} threads enters the barrier, all the threads waiting at the barrier wake up and unblock, and the n^{th} thread continues without blocking. When a thread calls the method reset( ), the barrier is reset so that it starts fresh in counting up to n (i.e., n more threads must enter the MyBarrier). You may start by modifying the following code fragment:

```
public class MyBarrier {
        public void MyBarrier (int n) { … }
        public enter( ) { … }
        public reset( ) { … }
}
```

b. Implement MyBarrier using Ruby monitors.

c. Write a Ruby program that creates a barrier for 2 threads, then creates 2 threads that each print out “hello”, enters the barrier, then prints out “goodbye”.

3. Garbage collection

Consider the following Java code.

```
Object a, b, c;
public foo( ) {
        a = new Object( );      // object 1
        b = new Object( );      // object 2
        c = new Object( );      // object 3
        a = b;
        b = c;
        c = a;
}
```

a. What object(s) are garbage when foo( ) returns? Explain why.

b. Describe the difference between mark-and-sweep & stop-and-copy.