EE 382C/EE 361C Multicore Computing Fall 2016
Assignment 3

Due: October 13
Instructor: Professor Vijay K. Garg (email: garg@ece.utexas.edu)
TA: Miao Qi (email:qimiao@utexas.edu)
TA: Asad Malik (email: asadsm@utexas.edu)

1. **(6 points)** For each of the histories below, state whether it is (a) sequentially consistent, (b) linearizable. Justify your answer. All variables are initially zero.

```
Concurrent History H1

P1            [ read(x) returns   1]
P2        [ write(x,1)                   ]    [ read(x)  returns 2]
P3              [write(x,2)      ]

Concurrent History H2

P1            [ read(x) returns   1]
P2        [ write(x,1)                   ]    [ read(x)  returns 1]
P3              [write(x,2)      ]

Concurrent History H3

P1            [ read(x) returns 1]
P2        [ write(x,1) ]                       [ read(x)  returns 1]
P3                         [write(x,2)      ]
```

2. **(4 points)** Consider the following concurrent program.

```
Initially a, b and c are 0.
P1:  a:=1 ; print(b) ; print(c);
P2:  b:=1 ; print(a) ; print(c);
P3:  c:=1 ; print(a) ; print(b);
```

Which of the outputs are sequentially consistent. Justify your answer.
(a) P1 outputs 11, P2 outputs 01 and P3 outputs 11.
(b) P1 outputs 00, P2 outputs 11 and P3 outputs 01.

3. **(30 points)** Newton, Benjamin, and Mary are planting seeds of apple trees. They have decided that Newton digs the holes and Benjamin places a seed into each hole. Then Mary fills the holes. There are some constraints to synchronize their progress:

   (a) Benjamin cannot plant a seed unless at least one empty hole exists, but Benjamin does not care how far Newton gets ahead of him.
   (b) Mary cannot fill a hole unless at least one hole exists in which Benjamin has planted a seed. Mary does not care how far Benjamin gets ahead of her.
   (c) Mary DOES care that Newton does not get more than MAX holes ahead of her. Thus, if there are MAX unfilled holes, Newton has to wait.

(d) There is only one shovel with which both Newton and Mary need to dig and fill the holes, respectively.

The pseudocode of the threads Newton, Benjamin, and Mary are given as follows.

```
Thread Newton {
  while (true) {
    garden.startDigging();
    dig();
    garden.doneDigging();
  }
}

Thread Benjamin {
  while (true) {
    garden.startSeeding();
    plantSeed();
    garden.doneSeeding();
  }
}

Thread Mary {
  while (true) {
    garden.startFilling();
    Fill();
    garden.doneFilling();
  }
}
```

Write a class Garden which uses ReentrantLocks and Conditions as the synchronization mechanism. Your class should implement: `startDigging, doneDigging, startSeeding, doneSeeding, startFilling` and `doneFilling`.

4. **(30 points)** The goal is to develop a concurrent implementation of a sorted linked list of integers. that uses n threads, where $n = 1, 2, 4, 8$. The linked list provides the following methods: `boolean add(int x), boolean remove(int x), boolean contains(int x)`. The method `add` returns true if x was not in the list and was added to the list; otherwise, it returns false (and does not add it multiple times). The method `remove` returns true if x was in the list and was removed from the list; otherwise, it returns false. The method `contains` returns true if x is in the list.

   Implement the following schemes:
   (a) Coarse-grained Locking
   (b) Fine-grained Locking
   (c) Lock-Free Synchronization.

5. **(30 points)** (a) Implement Lock-based and Lock-Free unbounded queues of `Integers`. For the lock based implementation, use different locks for `enq` and `deq` operations. For the variable `count` use `AtomicInteger`. For the lock-free implementation, use Michael and Scott's algorithm as explained in the class. The *deq* operation should block if the queue is empty.
   (b) Implement Lock-based and Lock-Free stacks of `Integer`. You should provide `push(Integer x)` and `Integer pop()`. The `pop` operation should throw an exception called `EmptyStack` if the stack is empty.
   For both the data structures use a list based implementation (rather than an array based implementation).