CS6868 Quiz 1

Dept of CSE, IIT Madras

Total marks = 24

Time = 45 min

16 Feb 2018

Read the instructions and questions carefully. You can make any reasonably assumptions that you think are necessary; but state them clearly. There are total three questions (eight marks each). You will need approximately 15 minutes for answering an eight marks question (plan your time accordingly). For questions with sub-parts, the division for the sub-parts are given in square brackets.

You will get an answer sheet with 8 pages (if you get a answer booklet with fewer pages then ask for a replacement). Leave the first page empty and start from Page#2. Start each question on a new page. Think about the question before you start writing and write briefly. **For any question, the answer (including the answers for all the sub-parts) should NOT cross more than two pages.** If the answer for any question is spanning more than two pages, we will strictly ignore the spill-over text. If you scratch/cross some part of the answer, you can use space from the next page. You mostly would NOT need any additional sheets.

1. [8] **Concurrency** State Amdahl's law related to speedup of parallel programs in the presence of multiple hardware threads. [1] Consider the following program:

```
main(){
   S1;   // x units of time
   parfor (i=0;i<n/2;++i)
     {S2;} // 0.5*x units of time
   S3;   // 0.75*x units of time
   parfor (i=0;i<n;++i)
     {S4;} // 2*x units of time
   S5;   // 1.25*x units of time
   parfor (i=0;i<2*n;++i)
     {S6;} // 1.5*x units of time
   S7;   // 1.75*x units of time
}
```

   Assume that `parfor` is a construct to create a loop whose iterations may run in parallel. Using Amdahl's law, compute the speedup; assume # processors=`n` [5]. For each of the two schedules given below, state if it is Linearizable and/or sequentially consistent [2].

```
// Thread 1
   q.enq(1); // start time = 0, end time = 5
   q.deq(2); // start time = 7, end time = 8
// Thread 2
   q.enq(2); // start time = 1, end time = 6
   q.deq(1); // start time = 7, end time = 9
               (a)
// Thread 1
   q.enq(1); // start time = 0, end time = 5
   q.deq(2); // start time = 8, end time = 9
// Thread 2
   q.enq(2); // start time = 7, end time = 10
               (b)
```

2. [8] **Non-blocking locks**: Explain the functionality of the `compareAndSet` method of `AtomicInteger`. [1] Give two reasons on how is it more beneficial to use a non-blocking primitive like CompareAndSet to realize synchronization (compared to a blocking scheme like the Java `synchronized` construct). [2] Give an implementation of Java `synchronized` construct (that is, implementations for `monitorEnter` and `monitorExit` methods) using the `AtomicInteger` class and `CompareAndSet` method as primitives. Your implementation should be such that the code '`a.monitorEnter(); S; a.monitorExit()`' would be considered a valid translation of the Java code of the form '`syncrhonized (a) S`' [3+1.5]. In which class, do you suggest the `monitorEnter` and `monitorExit` methods be defined? [0.5]

3. [8] **Java**: Provide an immutable variant of the following Java data structure [4].

```
class List {
   int data;
   List next;
   public List (int d, List n){ data=d; next = n; }
   public int synchronized getElem(){ return data; }
   public void synchronized incrAll(){
      incr();
      if (next != null) next.incrAll();
   }
   public void synchronized incr(){ data++; }
}
```

Which of the three properties *reflexivity, commutativity and transitivity* hold for the Happens-Before relationship? [1] For the following piece of Java code (annotated with labels for convenience), draw a directed graph depicting the happens-before relationship [3]. The nodes of the graph are the labels of the statement instances (a statement may be executed multiple times) and edges indicate the happens-before relation (an edge $a \rightarrow b$ indicates that a happend before b). For the ease of representation, denote a statement with Label `Lx` inside a method called from a label `Ly` as `Ly_Lx`.

```
class A extends Thread{
   public void run(){
      L1: S1;
      L21: synchronized (this) {
         L31 : S31
         L32 : S32
      L22: }
      L4: S4
   }
}
```

```
public class B {
   void main(String []s){
      L5: Thread t1 = new A();
      L6: Thread t2 = new A();
      L7: t1.start();
      L8: t2.start();
      L9: t1.join();
      L10: t2.join();
   }
}
```