COL 380		**Feb 15, 2017**
	Homework 1	
Instructor: Subodh Sharma		**Due: Feb 22, 23:55 hrs**

NOTE: All submissions must be made in the pdf format. Hand written assignments will not be accepted.

Problem 1: Sequential Consistency, Linearizability

- For each of the histories shown in Figs 1a and 1b, are they Sequentially consistent? Linearizable? Justify your answer. (4 marks)

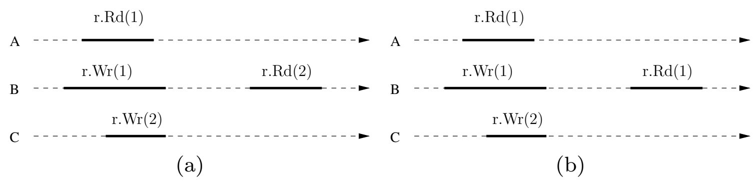

Figure 1: Traces

- Let *strict consistency* be defined in the following way: *Any read on a data item x returns a value corresponding to the result of the most recent write.* In class we had discussed the definitions of *sequential consistency* (SC) and *linearizability*. Assuming, we have a binary relation $\mathcal{W}$ that is irreflexive, antisymmetric and transitive which captures *Weaker-than* relationship among consistency models. Thus, if $(a, b) \in \mathcal{W}$ then model a is weaker than model b. Establish a $\mathcal{W}$ relationship among SC, strict consistency and linearizability. Justify your answer. (4 marks)
- A way to realize logical clocks (for establishing happens-before relation or causality among events that are necessary for SC or linearizability) is by either *Lamport clocks* or *Vector clocks*. Explain a cardinal difference between Lamport clocks and Vector clocks [Reference: Lamport clock video; Vector clock video]. Show at each event the associated vector clocks for a sequentially consistent execution history of example in Figure 1(a). (4 marks)

Problem 2:

Show that the `Filter lock` allows some threads to overtake others an arbitrary number of times. (4 marks)

Problem 3:

Consider the protocol shown below to achieve n-thread mutual exclusion:

```
                           int turn;
                           bool busy = false;

void lock() {                |         void unlock(){
```

```
 int me = tid.get();        |          busy = false;
 do {                       |        }
  do {                      |
    turn = me;              |
  } while(busy);            |
  busy = true;              |
 } while (turn != me);      |
}
```

For each question, either provide proof or display an execution where it fails!

- Does this protocol satisfy mutual exclusion? (3 marks)
- Is this protocol deadlock-free? (2 marks)

Problem 4:

Consider the FIFO queue implementation shown below. Notice datatype `AtomicInt`: it is an integer type of variables that can be updated atomically. It has an function `CompareAndSet` that compares the object's current value to `expect`. If the value is equal then it atomically replaces the object's value with `update` and returns *true*. Otherwise, it leaves the value of the object unchanged and returns *false*. The function `get()` returns the object's actual value atomically. Assume the `items` array is of unbounded size. `head` is the index of the next slot from which to remove an item and `tail` is the index of the next slot in which to place the item. Is the queue implementation linearizable? If so, give a proof and if not, provide a sample execution. (6 marks)

```
class MyQ<T> {

  AtomicInt head, tail;
  T items [MAX_INT_SIZE];
  void enq (T x){
    int slot;
    do {
      slot = tail.get();
    } while (!tail.CompareAndSet(slot, slot+1));
    items[slot] = x;
  }
  T deq(T x){
    T value; int slot;
    do{
      slot = head.get();
      value = items[slot];
      if (value == NULL)
      throw EmptyException();
    } while(!head.CompareAndSet(slot, slot+1));
    return value;
  }
};
```

Problem 5:

Show that the Bakery lock algorithm for n threads is deadlock-free and guarantees mutual exclusion. (6 marks).