Mid-Term for CSE153 (Winter 2014)

6th February 2014 **Name:**

Student ID:

Instructions:

***** This mid-term is for a total of 20 points.

***** Answer in brief. You will be graded for correctness, not on the length of your answers.

* Make sure to write legibly. Incomprehensible writing will be assumed to be incorrect.

I. For each of the 4 questions below, mark all the correct options. Every question has *one or more* correct options. (4 x 1 = 4)

1. Which of the following happen when fork() is invoked?

____ A new PCB is allocated

____ A new program is loaded into memory

____ A new address space is created

2. When a thread calls thread_yield, which of the following can potentially be the new state of the thread after the execution of thread_yield completes?

____ Waiting

____ Ready

____ Running

3. Which of the following statements are true in describing the various data structures that can be used for synchronization?

____ Lock.release() has no effect if no other thread is stuck on Lock.acquire()

____ Signal() on condition variable can impact future calls to Wait()

____ Implementation of lock that uses Test-and-Set does not waste CPU

4. Which of the following properties are true regarding lottery scheduling?

____ Prevents starvation

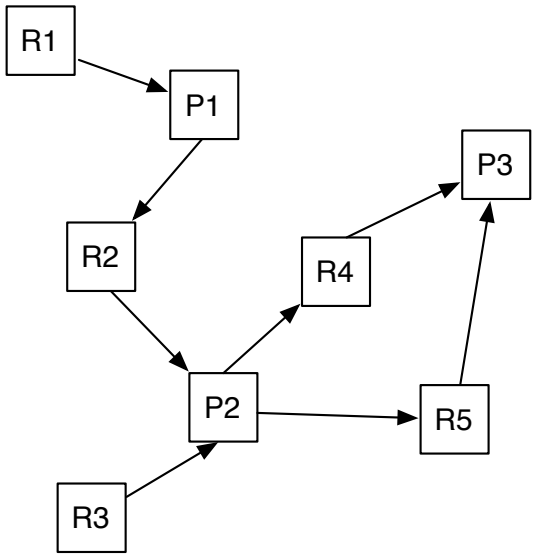
____ Results in higher waiting times compared to shortest job first scheduling

____ Can be used to implement round robin scheduling

II. Why is it okay to use the code "*if (condition not true) wait(condition_variable)*" within a Hoare monitor, whereas the equivalent code needs to be "*while(condition not true) wait(condition_variable)*" inside a Mesa monitor? (1 point)

III. What is one similarity between faults and interrupts? What is one difference between them? (2 points)

IV. Draw the Waits For Graph (WFG) corresponding to the Resource Allocation Graph (RAG) shown here. P1, P2, and P3 correspond to processes and R1, R2, R3, R4, and R5 correspond to resources. (2 points)



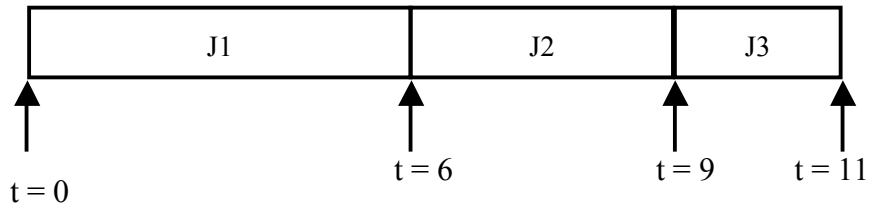
V. Three jobs are submitted in the following order: (2 + 2 + 2 = 6 points)

* Job J1 arrives at time t=0 and has a runtime of 6 seconds

* Job J2 arrives at time t=2 and has a runtime of 3 seconds

* Job J3 arrives at time t=6 and has a runtime of 2 seconds

On a machine with one CPU, draw the timeline of execution of these jobs for the following implementations of a scheduler: (A) non-preemptive shortest job first, (B) preemptive shortest job first, (C) round robin with a quantum of 1 second. Compute the average waiting time in each case. As an example, the timeline of execution for a first come first serve scheduler is shown below.



VI. Given the code below for the readers/writers problem using semaphores, answer each of the following questions in brief. (5 x 1 = 5 points)

a) Suppose that a writer is in the midst of performing a write and a reader is waiting at statement 6 in Read(). If another reader arrives at this instant, that reader will wait at which statement?

b) Suppose that a reader is in the midst of performing a read and a writer has invoked Write(). If a second reader invokes Read() at this instant,

i) the second reader will wait at statement 6 in Read(). True or False?

ii) the second reader will wait at statement 4 in Read(). True or False?

c) Starvation will not occur with this solution since a waiting thread, whether a reader or a writer, will always be awoken by a signal(wmutex) call. True or False?

d) What would be the effect of removing statements 8 and 10 from the code for the Read() function?

```
1:  semaphore wmutex = 1, rmutex = 1;
2:  integer num_readers = 0;

3:  Read() {
4:      wait(rmutex);
5:      if num_readers = 0
6:          wait(wmutex);
7:      num_readers += 1;
8:      signal(rmutex);
9:      Perform read;
10:     wait(rmutex);
11:     num_readers -= 1;
12:     if num_readers = 0
13:         signal(wmutex)
14:     signal(rmutex);
15: }
```

```
Write() {
    wait(wmutex);
    Perform write;
    signal(wmutex);
    }
```