**Department of Computer Science & Engineering**
**Indian Institute of Technology Kharagpur**
**Mid-semester Examination, Spring 2017**
**CS30002: Operating Systems**

**Full Marks: 100** **Time: 2 Hours**

**Answer ALL questions**

1. (a) List four important things that are part of the context of a process. (4)
   (b) When is a process said to be running in kernel mode (do not just say when it runs OS code)? (4)
   (c) Describe aging in the context of CPU scheduling. (4)
   (d) What is meant by a set of instructions being "atomic"? (4)
   (e) Describe (list steps as 1, 2, 3,...) how processes are run in a time-sharing system with round robin scheduling policy. In your answer, whenever you use some data (ex, the context of a process etc.), indicate the corresponding OS data structure that stores it. (8)

2. (a) Consider the following set of jobs. What would be the average waiting time if the following scheduling policies are used: (i) FCFS, (ii) Shortest-Remaining-Time-First? Show the Gantt chart and all calculations. Ignore the context switch time. (12)

| Process | Arrival Time | Duration |
|---|---|---|
| A | 0 | 12 |
| B | 3 | 7 |
| C | 6 | 2 |
| D | 8 | 5 |
| E | 9 | 2 |
| F | 12 | 12 |

   (b) Show with an example that Shortest-Job-First scheduling does not necessarily give the minimum waiting time for a set of jobs if the jobs arrive at different times. Do not use more than 3 jobs in your answer. (4)

3. (a) Define a semaphore. (5)

   (b) Suppose you have to write the code for the atomic *wait(S)* and *signal(S)* functions on a semaphore S for a non-preemptive OS on a machine that supports the *test-and-set* instruction, but does not have any instruction to enable/disable interrupts. A function *scheduler( )* is already implemented by someone else to invoke the scheduler that you can call. Write the pseudo code for the *wait( )* and *signal*() functions. Your pseudo code should talk directly about PCB and OS queues, and not just say block/wakeup process etc. Just write the pseudo code, no explanation is needed. (8 + 7)

4. (a) Give an example of race condition in which only one process writes to any shared data. (5)

(b) Suppose the CPU on which an OS runs supports an **atomic** instruction ***fetch_and_increment*** as follows:

```
int fetch_and_increment(int *x) {
    last_value = *x;
    *x = *x + 1;
    return last_value;
}
```

Show how you can use the above instruction to implement mutual exclusion. Your answer should be in two distinct parts: (i) give a solution that achieves mutual exclusion and progress, but may have starvation, (ii) give a modified solution that also achieves bounded waiting. For each part, just show the entry section and exit section codes clearly, no explanation is needed. (8 + 7)

(c) A barber shop has a barber, a barber chair, and a waiting room with 5 chairs. When a barber finishes cutting a customer's hair, the barber fetches another customer from the waiting room if there is a customer, or stands by the barber chair and daydreams if the waiting room is empty. A customer who needs a haircut enters the waiting room. If the waiting room is full, the customer goes away and comes back later. If the barber is busy but there is a waiting room chair available, the customer takes a seat. If the waiting room is empty and the barber is daydreaming, the customer sits in the barber chair and wakes the barber up.

Think of the barber and customers as processes which should be synchronized. You are required to write the code for the barber process and the customer process using semaphores. Your answer should first (i) list (in English sentences) what synchronization and/or critical section problems you have to solve, (ii) define what semaphores you have to use (including their initial value), and (iii) then write the pseudo code of the two processes. You can assume that *wait* and *signal* are available as primitive calls on a semaphore with their usual meanings. (20)