CS201 Fall 2019
Example midterm questions and topic review

1. In my system, here are the processes in the ready state, along with their burst times:

P1	5 ms
P2	3 ms
P3	10 ms
P4	7 ms

(a) Show the start time, end time, and wait time for each of these processes using a **first-come-first-served** scheduler. (3 points)

[here, draw a timeline diagram]

(b) What is the mean wait time for this example? (1 point)

(c) Show the start time, end time, and wait time for each of these processes using a **shortest-job-first** scheduler. (3 points)

[draw a timeline diagram]

(d) What is the mean wait time for this scheduling? (1 point)

(2) In the example above, I assumed that there were only four processes waiting for the CPU and that no new processes entered the ready state. Suppose that I have a constant stream of processes entering (and leaving) the ready state, and that I always pick the process with the shortest burst time to execute next.

(2a) What potential problem will this cause? (For full credit, give me the one-word term that describes the problem.) (1 point)

(2b) What is a way to prevent this problem? (For full credit, give me the one-word term that describes the solution.) (1 point)

(2c) I cannot look into the future and know what the next burst time will be for all of the processes in the ready state. What is a way that I can estimate what the next burst time will be for a process? (1 point)

(6) Process 1 communicates with Process 2 using blocking send() and blocking receive() calls, like this:

```
Process 1
while true {
    // fetch next message
    send(P2, message)
    receive(P2, messageMod)
```

```
Process 2
while true {
    receive(P1, message)
    modifiedMessage = modify(message)
    send(P1, modifiedMessage)
    // print messageMod                              }
}
```

(6a) True or false: this will work as we want and no additional synchronization is required. (1 point)

(7) Real-time process scheduling

(7a) What is the fundamental system requirement of a real-time process scheduler? (1 point)

(7b) What is preemption? (1 point)

(7c) Why is preemption necessary for real-time process scheduling? (1 point)

(10a) What is the difference between a user thread and a kernel thread? (1 points)

(10b) I write a multithreaded program to try to speed up my program. What must be true about the thread mapping in the operating system for me to get a shorter total run time for my program? (1 point)

(10c) What must be true about the hardware in the system if my program is going to have a shorter total run time? (1 point)

Topic review

- goals of an operating system
- thread; user thread; kernel thread
- relationship of a process to a thread, and the difference between a thread and a process
- context switch: what it means, when it happens, what it involves
- the process-state diagram; the transitions and when they happen
- synchronization: why it's necessary
- synchronization tools: mutex, semaphore
- the critical-section problem