CS 347 QUIZ 5 (August 25, 2016)

Name: _____ ***Roll No.*** _____

1. Consider the following three processes that arrive in a system at the specified times, along with the duration of their CPU bursts. Process P1 arrives at time t=0, and has a CPU burst of 10 time units. P2 arrives at t=2, and has a CPU burst of 2 units. P3 arrives at t=3, and has a CPU burst of 3 units. Assume that the processes execute only once for the duration of their CPU burst, and terminate immediately. Calculate the time of completion of the three processes under each of the following scheduling policies. For each policy, you must state the completion time of all three processes, P1, P2, and P3. Assume there are no other processes in the scheduler's queue. For the preemptive policies, assume that a running process can be immediately preempted as soon as the new process arrives (if the policy should decide to premempt).

A. First Come First Serve

B. Shortest Job First (non-preemptive)

C. Shortest Remaining Time First (preemptive)

D. Round robin (preemptive) with a time slice of (atmost) 5 units per process

2. Consider the exit system call in xv6. The exit function acquires ptable.lock before giving up the CPU (in the function sched) for one last time. Who releases this lock subsequently?

3. Consider the following actions that happen during a context switch from thread/process P1 to thread/process P2 in xv6. (One of P1 or P2 could be the scheduler thread as well.) Arrange the actions below in chronological order, from earliest to latest.

A. Switch ESP from kernel stack of P1 to that of P2
B. Pop the callee-save registers from the kernel stack of P2
C. Push the callee-save registers onto the kernel stack of P1
D. Push the EIP where execution of P1 stops onto the kernel stack of P1.

4. xv6 always performs a context switch by switching to the scheduler thread first, and then switching from the scheduler thread to the context of the next process. However, is it possible for any other OS design to switch directly from the context of one process to the context of another without going via a separate scheduler thread? (Yes/No)

5. Consider a user program thread that has locked a pthread mutex lock (that blocks when waiting for lock to be released) in user space. In modern operating systems, can this thread be context switched out or interrupted while holding the lock? (Yes/No)

6. Repeat the previous question when the thread holds a pthread spinlock in user space.

7. Consider a process that has switched to kernel mode and has acquired a spinlock to modify a kernel data structure. In modern operating systems, will this process be interrupted by external hardware before it releases the spinlock? (Yes/No)

8. In xv6, is it possible that one process acquires a lock, but some other process releases it? If yes, give an example where such a thing happens. If not, explain why not.

9. When a user space process executes the wakeup/signal system call on a pthread conditional variable, does it always lead to an immediate context switch of the process that calls signal (immediately after the signal instruction)? (Yes/No)

10. In xv6, when a process calls wakeup on a channel to wakeup another process, does this lead to an immediate context switch of the process that called wakeup (immediately after the wakeup instruction)? (Yes/No)

11. When a process terminates in xv6, when is the struct proc entry of the process marked as unused/free?

A. During the execution of exit
B. During the sched function that performs the context switch
C. In the scheduler, when it iterates over the array of all struct proc
D. During the execution of wait by the parent