CSCI-GA.2250-001
Midterm Exam (October 4th, 2011)
Duration: 90 minutes

Last Name: _____ First Name: _____
ID#:

Notes:

- **If you perceive any ambiguity in any of the questions, state your assumptions clearly**
- **Questions vary in difficulty; it is strongly recommended that you do not spend too much time on any one question.**
- **This exam is open book/notes.**

---

1. (5 points) Circle the correct answer among the choices given. If you circle more than one answer, you will loose the grade of the corresponding question.

(A) Which of the following is NOT a task of the operating system?

1. Abstraction of resources
2. Resource management
3. Provide a friendly user interface
4. none of the above

(B) Which one of the following will more likely cause race condition?

1. Two threads from the same process
2. Two threads from different processes
3. Both have the same probability
4. Two threads with no critical regions

(C) Which of the following conditions is NOT necessary in order to have a possibility of race condition

1. The existence of at least two processes
2. The existence of critical region in all processes involved
3. The existence of critical region in at least two processes
4. Scheduling is preemptive

(D) Relocation, during linking, is needed because:

1. The compiler does not know where the program or modules will be loaded
2. The compiler does not know how big the memory will be
3. The compilers does not know the size of each module
4. It is not the fault of the compiler.

(E) Which of the following is the slowest?

1. function call 2. call to a library 3. system call 4. They are all the same.

2. [3 points] Assume we have two kernel-level threads corresponding to the same process. Also assume we have a processor with single core. Which one is slower: context switch between the two threads (i.e. OS scheduler decides to stop one thread and run the other) or context switch between two different processes? and why?

3. [4 points] We have seen several scheduling algorithms in class. Two of those are round-robin and lottery scheduling. How can we simulate round-robin using lottery scheduling?

4. [4 points] Suppose that a multi-threaded program requires a lot of I/O. Is it better (from program execution time) to have kernel-level threads? or user-level threads? and Why?

5. [4 points] We have studied several techniques that make a process use the CPU while doing nothing but waiting for some event (e.g. another process to release a lock) instead of being blocked. This is called *busy waiting*. We also said that busy waiting is not desirable. When do you think busy-waiting is desirable?