Computer Science 15-410: Operating Systems
Mid-Term Exam (A), Fall 2003

1. **Please read the entire exam before starting to write. This should help you avoid getting bogged down on one problem.**
2. Be sure to put your name and Andrew ID below *and also* put your Andrew ID at the top of *each* following page.
3. This is a closed-book in-class exam. You may not use any reference materials during the exam.
4. You must complete the exam by the end of the class period.
5. Answer all questions. The weight of each question is indicated on the exam. Weights of question *parts* are estimates which may be revised during the grading process and are for your guidance only.
6. Please be concise in your answers. You will receive partial credit for partially correct answers, but truly extraneous remarks may count against your grade.
7. **Write legibly even if you must slow down to do so!**. If you spend some time to *think clearly* about a problem, you will probably have time to write your answer legibly.
8. Please don't ask us questions of the form "If I answered like this, would it be ok?" or "Are you looking for ...?"

Andrew Username	
Full Name	

Question	Max	Points	Grader
1.	10		
2.	10		
3.	10		
4.	10		
5.	10		
	100		

1. [10 points] Give a *brief* definition of each of the following terms as they apply to this course. You may add a second sentence providing an example or a clarification.

   (a) [2 points] Register

   (b) [2 points] Stack

   (c) [2 points] Program counter

(d) [2 points] System call

(e) [2 points] Preemption

2. [10 points] What will happen if the following code is executed by a process in kernel mode?

```
void foo(void)
{
    while (1)
        (void) malloc(1);
}
```

If you feel that multiple scenarios are possible, please pick one and focus on its details. You may wish to break your answer into several parts, such as "At first, ...", "After some time..." and/or "Eventually...". You may wish to consider the effects, if any, on other processes in the system.

3. [10 points] For project 3, you are writing a preemptive multi-tasking kernel. For each process, the kernel provides a separate kernel stack.

   (a) [4 points] Why is it better to use multiple kernel stacks instead of a single kernel stack? What difficulties would you encounter using a single kernel stack?

(b) [6 points] How can you determine what is a good size for the kernel stack? Assume you are deciding on this value after you have coded and debugged your kernel and have the luxury of performing an analysis of what you wrote.

4. [10 points] Deadlock

   (a) [5 points] State the 4 conditions necessary for deadlock to occur. For each condition, explain what the condition means.

Consider a system with two CD-R burners (S and T). Consider the sequence of requests shown below:

P0	P1
Request(S)	
	Request(T)
Request(T)	
	Request(S)

(b) [5 points] Draw the resource allocation diagram at this point.

(c) [5 points] As can be seen, these two processes are in deadlock. Now, assume that instead of granting all requests, the system had used deadlock avoidance. Also, assume that both Process 1 and Process 2 had stated their worst-case needs to include both S and T. Briefly explain how execution would have proceeded in a deadlock-free fashion.

5. [20 points] Consider the following critical-section protocol:

```
        boolean waiting[2] = { false, false };
        int turn = 0;

 1.     do {
 2.         waiting[i] = true;
 3.         while (waiting[j]) {
 4.             if (turn == j) {
 5.                 waiting[i] = false;
 6.                 while (turn == j)
 7.                    /* do nothing */ ;
 8.                 waiting[i] = true;
 9.             }
10.         }
11.         waiting[i] = false; /* done waiting! */
12.         ...critical section...
13.         turn = j; /* your turn */
14.         ...remainder section...
15.     } while (1);
```

(This protocol is presented in the standard form, i.e., if process 0 is running this code, $i == 0$ and $j == 1$; if process 1 is running this code, $i == 1$ and $j == 0$.)

There is a problem with this protocol. That is, it does not ensure that all three requirements (mutual exclusion, progress, and bounded waiting) are always met. Identify a requirement which is not met and lay out a scenario which demonstrates your claim. Use the format presented in class, i.e.,

P0	P1
waiting[0] = false;	
	turn = 0;