Computer Science 15-410: Operating Systems
Mid-Term Exam (B), 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		
	50		

1. $\boxed{\text{10 points}}$ 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) $\boxed{\text{2 points}}$ Instruction

   (b) $\boxed{\text{2 points}}$ Heap

   (c) $\boxed{\text{2 points}}$ Stack Pointer

(d) $\boxed{\text{2 points}}$ Interrupt

(e) $\boxed{\text{2 points}}$ Round-robin

2. $\boxed{\text{10 points}}$ What will happen if the following code is executed by a process in user mode?

```
void bar(void)
{
    int sum = 0, *ip;

    ip = &sum;
    while (1) {
        sum += *ip;
        --ip;
    }
}
```

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. $\boxed{\text{10 points}}$ User memory

   (a) $\boxed{\text{5 points}}$ How many page faults can occur in the following piece of code? You may assume pages are 4K bytes.

```
void foo () {
    int i;
    char x[2000];

    for ( i = 0 ; i < 2000 ; i++ ) {
        x[i] = '\0';
    }
}
```

   Explain your answer.

   Note: yes, you should use a #define instead of 2000. We only used 2000 for maximum clarity on the exam.

(b) $\boxed{\text{5 points}}$ When a pointer access occurs in the void between the stack and the heap, most of the time this is a wild access and we want to kill the process. However, when the stack pointer passes the last allocated page for it, we want to grow the stack. Therefore, we need to set a "maximum stack growth due to one memory reference" value: growth smaller than this value is legal and growth larger than this value will kill the process.

What would be a good value, and why?

4. $\boxed{\text{10 points}}$ Deadlock

   (a) $\boxed{\text{5 points}}$ 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) $\boxed{\text{5 points}}$ Draw the resource allocation diagram at this point.

(c) $\boxed{\text{5 points}}$ As can be seen, these two processes are in deadlock. Now, assume that instead of granting all the requests, the system had used deadlock prevention. What rule could you have enforced to prevent this situation from occurring? What would the execution sequence have been?

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

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

do {
    waiting[i] = true;
    while (waiting[j]) {
        if (turn == j) {
            waiting[i] = false;
            waiting[i] = true;
        }
    }
    ...critical section...
    turn = j;
    waiting[i] = false;
    ...remainder section...
} 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;