CS 4760	**Operating Systems**	Test 2
Name:	Fall 2004	Max Pts: 50

Important: This is an open book test. You can use any books, notes, or paper. *You are not allowed to use any communications device. Do not log into the computer during the test.* Any calculations and rough work can be done on the back side of the test pages. If there is a syntax error in any program segment, just write it down and you will get full credit for the problem. You will lose five points for not writing your name.

1. [4 pt] What is the difference between binary and general semaphores?

2. [4 pt] What is the distinction between *blocking* and *nonblocking* with respect to messages?

3. [6 pt] One of the ways to prevent deadlocks was given as creating a total order on the resources and allocating resources in the order of enumeration. Show by an example why a partial order will not be sufficient to achieve the same.

4. [8 pt] Consider the following program:

```
const int n = 10;
int tally;

void total()
{
    int count;
    for ( count = 0; count < n; count++ )
        tally++;
}

int main()
{
    tally = 0;
    cobegin
        total();
        total();
    coend

    printf ( "%d\n", tally );

    return ( 0 );
}
```

Determine the proper lower bound and upper bound on the final value of the shared variable `tally` output by this concurrent program. Assume processes can execute at any relative speed and that a value can only be incremented after it has been loaded into a register by a separate machine instruction.

5. [10pt] Assume that you have the following jobs to be executed with one processor:

Process	Burst time	Arrival time
p_0	5	0
p_1	7	2
p_2	6	3
p_3	4	9
p_4	5	13

Give the average wait time and average turnaround time for each process using the following algorithms. Is the CPU idle at any time in the given algorithms?

(a) First in first out

(b) Shortest job next (no preemption)

(c) Shortest remaining time next

(d) Round robin, with a quantum of 3

(e) Round robin, with a quantum of 4 plus context switch time of 1

6. [10 pt] Assume a system with four resource types, $C = \langle 9, 8, 13, 9 \rangle$ (this is the total number of resources in the system, and not what is currently available), and the maximum claim table shown below.

Process	R_0	R_1	R_2	R_3
p_0	1	5	9	5
p_1	4	6	5	5
p_2	2	7	9	4
p_3	7	5	2	4
p_4	1	1	1	8

The resource allocator is considering allocating resources according to the following table:

Process	R_0	R_1	R_2	R_3
p_0	1	1	6	1
p_1	0	1	4	3
p_2	2	1	0	1
p_3	0	4	2	0
p_4	1	1	1	4

Run the safety algorithm on this system to determine if this state is safe. If it is safe, give the sequence in which processes can be run. If it is unsafe, enumerate the processes that may get involved in a deadlock.

7. [8 pt] Consider a system $\langle \sigma, \pi \rangle$ with $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ and $\pi = \{p_0, p_1, p_2\}$. State changes are:

$$\begin{array}{lllll}
p_0(\sigma_0) = \Omega & p_0(\sigma_1) = \{\sigma_2, \sigma_4\} & p_0(\sigma_2) = \{\sigma_1\} & p_0(\sigma_3) = \{\sigma_2\} & p_0(\sigma_4) = \{\sigma_0\} \\
p_1(\sigma_0) = \Omega & p_1(\sigma_1) = \Omega & p_1(\sigma_2) = \{\sigma_0, \sigma_3, \sigma_4\} & p_1(\sigma_3) = \Omega & p_1(\sigma_4) = \Omega \\
p_2(\sigma_0) = \{\sigma_0, \sigma_2\} & p_2(\sigma_1) = \{\sigma_0\} & p_2(\sigma_2) = \{\sigma_0\} & p_2(\sigma_3) = \Omega & p_2(\sigma_4) = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}
\end{array}$$

Draw the state diagram. Can you identify a knot in there? Is there a possibility of a deadlock?