CS 376	**Operating Systems**	Test 2
Name:		Maximum Points: 54

Important: This is an open book test. You can use any books, notes, or paper. *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.

1. [4 pt] Explain the difference between busy waiting and blocking.

2. [6 pt] Consider a computer with two processes, H, with high priority and L, with low priority. The scheduling rules are such that H is run whenever it is in ready state. At a certain moment, with L in its critical region, H becomes ready to run (e.g., an I/O operation completes). H now begins busy waiting, but since L is never scheduled while H is running, L never gets the chance to leave its critical region, so H loops forever. This situation is sometimes referred to as the *priority inversion problem*.

   Does the same problem occur if round robin scheduling is used instead of priority scheduling? Discuss.

3. [5pt] Here is the code for the `putdown` function in the dining philosophers problem:

```
putdown ( int i )
{
    /* The number of philosophers around the table is given by n          */

    state[i] = thinking;          /* Start thinking again                  */
    test ( ( i + n - 1 ) % n );   /* Check if left philosopher is hungry   */
    test ( ( i + 1 ) % n );       /* Check if right philosopher is hungry  */
}
```

Suppose the variable `state[i]` was set to `thinking` *efter* the two calls to `test`, rather than *before*. How would this change affect the solution for the case of 3 philosophers ($n = 3$). For 100 philosophers?

4. [12 pt] Five batch jobs A through E, arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4, and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead.

   (a) FIFO

   (b) SJN

   (c) Priority scheduling

   (d) Round robin, with a quantum of 2

For (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b) through (d), assume that only one job runs at a time until it finishes. All jobs are completely CPU bound.

5. [3+3+5+5 pt] Consider the following snapshot of a system. There are no current outstanding queued unsatisfied requests.

	Current allocation				Still needs			
Process	R_0	R_1	R_2	R_3	R_0	R_1	R_2	R_3
p_0	3	0	1	1	1	1	0	0
p_1	0	1	0	0	0	1	1	2
p_2	1	1	1	0	3	1	0	0
p_3	1	1	0	1	0	0	1	0
p_4	0	0	0	0	2	1	1	0
Available	1	0	2	0				

(a) Is this system currently in a safe or unsafe state? Why?

(b) Which processes, if any, are or may become deadlocked?

(c) If a request from p_3 arrives for $(0, 1, 0, 0)$, can that request be safely granted immediately? In what state (deadlocked, safe, unsafe) would immediately granting that whole request leave the system? Which processes, if any, are or may become deadlocked if this whole request is granted immediately?

(d) After granting (or denying) the above request, process p_1 requests an instance of resource R_2. This is followed by a request from p_4 for an instance of R_2. Can these requests be safely granted immediately? In what state (deadlocked, safe, unsafe) would immediately granting these requests leave the system? Which processes, if any, are or may become deadlocked if this whole request is granted immediately?

6. [2 pt] What is the difference between a physical address and a logical address? How does a system end up with the two kinds of addresses (explain with binding times).

7. [9 pt] Consider a swapping system in which the memory consists of the following hole sizes in memory order: 10K, 4K, 20K, 18K, 7K, 9K, 12K, and 15K. Which hole is taken for successive segment requests of

   (a) 12K
   (b) 10K
   (c) 9K

   for first fit? Now repeat the question for best fit and worst fit.