CS 376	**Operating Systems**	Test 1
Name:	Winter 1998	Max Pts: 40

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. You will loose five points for not writing your name.

1. [5pt] In virtually all systems that include DMA modules, DMA access to main memory is given higher priority than processor access to main memory. Why?

2. [5 pt] An I/O-bound program is one that, if run alone, would spend more time waiting for I/O than using the processor. A processor-bound program is the opposite. Suppose a short-term scheduling algorithm favors those programs that have used little processor time in the recent past. Explain why this algorithm favors I/O-bound programs and yet does not permanently deny processor time to processor-bound programs.

3. [10 pt] Consider the following program:

```
extern bool blocked[2];    // Shared flags between two processes
extern int  turn;          // Shared variable

void P ( int id )          // id contains the process id (0 or 1)
{
    do
        blocked[id] = true;
        while ( turn != id )
        {
            while ( blocked[1-id] ) /* do nothing */;
            turn = id;
        }

        critical_section();

        blocked[id] = false;

        remainder_section();
    while ( 1 );
}
```

Find a counterexample to demonstrate that this solution is incorrect.

4. [10 pt] The following expression describes the serial/parallel precedence relationship among six processes, p_1 through p_6:

$$P(S(P(p_3, S(p_1, P(p_6, p_5))), p_2), p_4)$$

where $P(x, y)$ implies that x and y can be executed in parallel and $S(x, y)$ implies that y must be executed after x.

Transform this expression into a program using:

(a) `fork`, `join`, and `quit` primitives

(b) `cobegin`/`coend`

5. [5 pt] Assume that each of the five philosophers, i, in the dining philosophers problem executes the following segment of code:

```
do
    /* Get hungry, Pick up forks */
    P ( mutex );
    P ( fork[i] );
    P ( fork [(i+1)%5];
    V ( mutex );
    eat();
    /* Put down forks */
    V ( fork[i] );
    V ( fork [(i+1)%5];
    think();
while ( 1 );
```

Does this code satisfy all requirements of the dining philosophers problem?

6. [5 pt] Jurassic Park consists of a dinosaur museum and a park for safari riding. There are m passengers and n single passenger cars. Passengers wander around the museum for a while, then line up to take a ride in the safari car. When a car is available, it loads the one passenger it can hold and rides around the park for a random amount of time. If the n cars are all out riding passengers around, then a passenger who wants to ride waits; if a car is ready to load but there are no waiting passengers, then the car waits. Use semaphores to synchronize the m passenger processes and the n car processes.

   Is the following solution for the above problem correct? Explain your answer.

```
extern sem car_avail = 0;
extern sem car_taken = 0;
extern sem car_filled = 0;
extern sem passenger_released = 0;

void passenger ( int m )
{
    do
        sleep ( random_amount_of_time );  // Assume a random number as parameter
        P ( car_avail );
        V ( car_taken );
        P ( car_filled );
        P ( passenger_released );
    while ( 1 );
}

void car ( int n );
{
    do
        V ( car_avail );
        P ( car_taken );
        V ( car_filled );
        sleep ( random_amount_of_time );
        V ( passenger_released );
    while ( 1 );
}
```