Midterm Exam
March 7, 2001
Anthony D. Joseph
CS162 Operating Systems

General Information:
This is a **closed book and note** examination. You have ninety minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicated the number of points given to the question; there are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answersdirectly on this paper ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

Problem	Possible	Score
1	12	
2	24	
3	23	
4	18	
5	23	
Total	**100**	

1.(12 points total) "Lightweight" versus Heavyweight processes - Pros and Cons:
a.(4 points) List one advantage of running several single-threaded applications as "heavyweight" processes over running the aplications as multiple "lightweight" processes, all in one address space. Be explicit, but *concise*, in your answers.
i)

b.(8 points) List two reasons why lightweight processes are better than heavyweight ones.
i)

ii)

2.(24 points total) Suppose that we have a multiprogrammed computer in whihc each job has identical characteristics. In one computation period, T, for a job, half the time is spent in I/O and the other half in processor activity. Each job runs for a total of N periods. Assume that a simple round-robin scheduling

scheme is used and that I/O operations can overlap with processor operation. We define the following quantities:

Turnaround time = actual time to complete job

Processor utilization = percentage of time that the processor is active (not waiting).

For large N, compute approximate values for these quantities for one, two, and four simultaneous jobs, assuming that the period *T* is distributed in each of the following ways:

a.(15 points) I/O first half, processor second half.

i) 1 job, Turnaround time and Processor utilization:

ii) 2 jobs, Turnaround time and Processor utilization:

ii) 4 jobs, Turnaround time and Processor utilization:

b.(9 points) I/O first and fourth quarters, processor second and third quarters.

i) 1 job, Turnaround time and Processor utilization:

ii) 2 jobs, Turnaround time and Processor utilization:

ii) 4 jobs, Turnaround time and Processor utilization:

3.(23 points total) CPU scheduling.

a.(8 points) Given CPU-bound tasks and a choice between FIFO amd Round-Robin scheduling algorithms, choose the best algortihm for each of the following systems and specify why you chose the algorithm.
i)Multiprogrammed batch system:

ii)Interactive time-sharing system:

b.(15 points) Consider the following processes, arrival times and CPU processing requirements:

Process Name	Arrival Time	Processing Time
1	0	4
2	1	4
3	4	3
4	8	2

For each of the following scheduling algorithms, fill in the table with the process that is running on the CPU. Assume a 1 unit timeslice for timeslice-based algorithms. For RR, assume that an arriving thread runs at the beginning of its arival time.

Time	**FIFO**	**RR**	**SRTF**
0	**1**	**1**	**1**
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
Average response time			

4.(18 points total) Concurrency problem: Building H_2O.

> The goal of this exercise os for you to create a monitor with methods Hydrogen() and Oxygen(), which wait until a water molecule can be formed and then return. Don't worry about actually creating the water molecule; instead only need to wait until two hydrogen threads and one oxygen thread can be grouped together. For example, if two threads call Hydrogen, and then a third thread calls Oxygen, the third thread should wake up the first two threads amd they should all then return.

a.(6 points) Specify the correctness constraints. Be succinct and explicint in your answer

b.(12 points) Observe that there is only one condition any thread will wait for(i.e., a water molecule being formed). However, it will be necessary to signal hydrogen and oxygen threads independently, so we choose to use two condition variables, waitingH and waitingO.

Monitor	**Variable name**	**Initial value**
Number of waiting hydrogen threads	**wH**	0
Number of waiting oxygen threads	**wO**	0
Number of active hydrogen threads	**aH**	0
Number of active oxygen threads	**aO**	0

You start with the following code:

```
Hydrogen(){
  wH++;
  lock.acquire();
  while(aH ==0){
    if(wH>=2 && wO >=1) {
      wH-=2; aH+=2;
      wO-=1; aO=+1;
      waitingH.broadcast();
      waitingO.signal();
    } else {
      lock.release();
      waitingH.wait();
      lock.acquire();
    }
  }
  lock.release();
  aH--;
}

Oxygen() {
  wO++;
  while(aO ==0){
    if(wH >=2 && wO >= 1){
      wH-=2; aH+=2;
      wO-=1; aO=+1;
      waitingH.signal();
      waitingH.signal();
    } else {
      waitingO.broadcast();
    }
```

```
    }
    a0--;
  }
```

For each method, say whether the implementation either (i) works, (ii) doesn't work, or (iii) is dangerous - that is, sometimes works and sometimes doesn't. If the implementation does not work or is dangerous, explain why (there maybe several errors) and show how to fix it so it does work. Also list and fix any inefficiencies.

i. Hydrogen()

ii. Oxygen()

5. (23 points) Deadlock:

Consider the following snapshot of a system with five processes(p1,...p5) and four resources (r1,...r4). There are no current outstanding queued unsatisfied requests.

currently available resources

r1	r2	r3	r4
2	1	0	0

current allocation|max demand|still needs

Process	r1	r2	r3	r4	r1	r2	r3	r4	r1	r2	r3	r4
p1	0	0	1	2	0	0	1	2				
p2	2	0	0	0	2	7	5	0				
p3	0	0	3	4	6	6	5	6				
p4	2	3	5	4	4	3	5	6				

Good Luck!!

p5	0	3	3	2	0	6	5	2

a. (5 points) Compute what each process still might request and fill in the "still needs" columns.

b. (8 points) Is this system currently deadlocked, or will any process become deadlocked? Why or why not? If not, give an execution order.

c. (10 points) If a request from p3 arrives for (0,1,0,0), can that request be safely granted immediately? In what state (deadlocked, safe, unsafe) would immediately granting the whole request leave the system? Which processes, if any, are or may become deadlocked if this whole request is granted immediately?