

Operating Systems Comprehensive Examination
Spring Quarter, 2008

ID _____

NOTE: Please write your ID on every page of the exam.

1	
2	
3	
4	
5	
Total	

2. (25 pts) Circular Dependencies.

A serious problem for operating systems developers is dealing with circular dependencies that typically either lead to the potential for deadlock, complex implementations, severe limitations in performance, or a combination of the above. For **three** (and only three) of the following systems: (i) describe the circular dependency problem the system was most concerned about, and (ii) describe the approach taken to address the problem, and how it solved the problem.

(a) Medusa

(b) Soft Updates

(c) THE

(d) Pilot

(e) Swift

3. (25 pts) Implementing Semaphores.

Consider the following kernel implementation of semaphore operations:

```
wait (sem s) {  
    if s equals 0, block the running process and associate with s;  
    decrement s;  
    return;  
}
```

```
signal (sem s) {  
    increment s;  
    if any blocked processes associated with s, unblock one of them;  
    return;  
}
```

Assume that the kernel executes with interrupts disabled.

(a) Is there any problem with this implementation for execution on a uniprocessor? Explain (and state all your assumptions).

(b) Is there any problem with this implementation for execution on a multiprocessor? Explain (and state all your assumptions).

4. (25 pts) **Implementing Semaphores.**

Consider the following atomic instruction FetchAndAdd implemented as follows:

```
function FetchAndAdd(address location) {  
    int value := *location  
    *location := value + 1  
    return value  
}
```

Give pseudocode for acquiring and releasing a spin lock based on FetchAndAdd. You should give code for locking and for unlocking the spin lock. Hint: Unlike the solution used with the TestAndSet atomic instruction, use two integers rather than one to represent the lock, and don't have lock or unlock spin continuously executing FetchAndAdd.