CS 4431: Advanced Operating Systems
Midterm: February 3, 1998 9:30 to 11:00
OPEN BOOK AND NOTES
CONCISE BULLETS GET MORE CREDIT THAN WORDY SENTENCES
LEGIBILITY IS A REQUIREMENT NOT AN OPTION!

1. (1 point, 1 min) With all the news about the Asian economic crises bombarding the air-waves, can you name the currency of Indonesia?

   (a) Dollar

   (b) Pound

   (c) Yen

   (d) Rupiah

   (e) Dinar

   (f) Rouble

2. (4 points, 4 min) (*Answer True or False with justification*). Dynamic delay performs better than static delay at low lock contention for the lock algorithms presented in Anderson's paper on "The performance of spinlock algorithms on shared memory multiprocessors".

3. (5 points, 5 min) (*Answer True or False with justification*). On any shared memory multiprocessor, the following lock algorithm:

   ```
   lock: while (lock == busy) spin;
         if (test-and-set(lock) == busy) go to lock;
   ```

   always results in reducing any disruption to the processor doing useful work compared to the following lock algorithm:

   ```
   lock: while ((test-and-set(lock) == busy) spin;
   ```

4. (5 points, 5 min) (*Answer True or False with justification*). Writing an application as a multi-threaded program on a uniprocessor is purely for modularity not performance.

5. (5 points, 5 min) (*Answer True or False with justification*). It is impossible to implement a pre-emptive user level threads scheduler unless the operating system itself is thread-aware.

6. (5 points, 5 min) In Solaris OS, there are *kernel threads*, *light-weight processes*, and *user level threads*. What are the differences between these three computational abstractions?

7. (10 points, 5 min) What is a *protected procedure call* as defined in the Psyche operating system? How is it implemented? What is the blocking semantics of this call for the caller, and explain the rationale behind it?

8. (15 points, 10 min) Explain the concurrency, priority, and exclusion properties of the following path expressions:

(a) path {A; B} end;

(b) path {A} + {B} end;

9. (20 points, 15 min) Processes P1 and P2 wish to synchronize with each other using a monitor, by calling two entry point procedures A and B respectively. Irrespective of the order of arrival of the processes to the monitor, the intent is for the first arriving process to wait for the other, and for both to resume their respective executions after the rendezvous in the monitor. Does the monitor below achieve the intended synchronization? If not, fix it.

```
monitor
{
        exports A and B;
        x, y: condition; /* condition variables */
        A
        {
             signal(y);
             wait(x);
        }

        B
        {
             signal(x);
             wait(y);
        }
} /* end monitor */
```

```
Processes P1 and P2 do the following:

P1: ...
    ...
    monitor.A;
    ...

P2: ...
    ...
    monitor.B;
    ...
```

10. (30 points, 25 min) Develop a serializer solution for a printer manager that manages a pool of *M* printers with the following attributes:

- a request for a printer should be granted right away if one is available;
- the requester should return the printer to the pool of available printers upon using it;
- waiting requesters (if any) for a printer should get the printer in FCFS order when one becomes available.