

COMPSCI 110 Fall 2002

Midterm

Name: _____

README: Individual effort only, OPEN book and notes, time allowed: 75 minutes. Put your name on each sheet in case the pages get separated. Be brief, but show your work for partial credit in case your answer may be wrong. **State any assumptions you make.**

1) (concurrency and synchronization)

The problem with the solution to the right is:

- (a) Race conditions
- (b) Missed wakeup
- (c) Starvation
- (d) Deadlock
- (e) b and d above
- (f) a and c above.

Answer is e

Repeat for the solution below:

- (a) Race conditions
- (b) Missed wakeup **Answer is d**
- (c) Starvation
- (d) Deadlock
- (e) b and d above
- (f) a and c above.

```
/*turn initially = DUM*/
void Tweedledee()
{
    while(1) {
        mutex->Acquire ();
        x = Quarrel(x);
        turn=DUM;
        cond->Signal(&mutex);
        cond->Wait(&mutex);
        mutex->Release ();
    }
}

void Tweedledum()
{
    while(1) {
        mutex->Acquire ();
        turn=DEE;
        cond->Wait(&mutex);
        x = Quarrel(x);
        cond->Signal(&mutex);
        mutex->Release ();
    }
}
```

/*turn initially = DEE;
semaphores dee and dum initially 0*/

```
void Tweedledee()
{
    while(1) {
        mutex->Acquire ();
        x = Quarrel(x);
        turn=DUM;
        dum->V( );
        while(turn==DUM)
            dee->P( );
        mutex->Release ();
    }
}
```

```
void Tweedledum()
{
    while(1) {
        mutex->Acquire ();
        while(turn==DEE)
            dum->P( );
        x = Quarrel(x);
        turn=DEE;
        dee->V( );
        mutex->Release ();
    }
}
```

COMPSCI 110 Fall 2002
Midterm

Name: _____

2) (Concurrency and Synchronization)

Show how counting semaphores (i.e. semaphores that can hold values greater than 1) can be implemented using only binary semaphores and ordinary instructions.

binary semaphores mutex initially 1; blocking initially 0;
integer count initially n; // whatever n is

```
Pmult()  
{  
    mutex->down();  
    count --;  
    if (count >= 0)  
    { mutex->up();  
      return; }  
    // something about lists -- not relevant  
    mutex -> up ();  
    blocking -> down ()  
}
```

```
Vmult()  
{  
    mutex->down();  
    count ++;  
    if (count > 0) // == is not good enough and conditional is necessary  
    { mutex->up();  
      return; }  
    // something about lists -- not relevant  
    blocking -> up ();  
    mutex -> up ();  
}
```

COMPSCI 110 Fall 2002
Midterm

Name: _____

binary semaphores mutex initially 1; blocking initially 0;
integer count initially n; // whatever n is
integer numwait initially 0;

Pmult()

```
{
    mutex->down( );
    while (count == 0)
    { numwait ++;
      mutex->up( );
      blocking -> down ( );
      mutex->down( ); }
    count --;
    mutex -> up ( );
}
```

Vmult()

```
{
    mutex->down( );
    count ++;
    if (numwait > 0)
    { numwait - -;
      blocking -> up ( );}
    mutex -> up ( );
}
```

COMPSCI 110 Fall 2002

Midterm

Name: _____

3) (Scheduling)

Measurements of a certain system have shown that the average process runs for a time T before blocking on I/O. A preemptive process switch requires a time S , which is effectively overhead. For round robin scheduling with quantum Q , **give a formula for the CPU efficiency** for each of the following cases:

a. $Q > T$

$$\frac{T}{T + S}$$

b. $S < Q < T$

$$\frac{Q}{Q + S}$$

c. Q nearly 0

Approaches 0

d. Give one reason why you might want to modify the scheduling policy to have Q not be a constant value.

Have to capture some notion of varying to adapt to conditions:

to distinguish between I/O and compute bound

to implement proportional scheduling

to adapt to job mix

COMPSCI 110 Fall 2002

Midterm

Name: _____

4) (Deadlock and starvation)

In an electronic funds transfer system, there are hundreds of identical processes that work as follows: each process reads in input line specifying an amount of money, the account number to be credited, and the account number to be debited (i.e., both account numbers known). The process must lock both accounts, then transfer the money, and finally release the locks when finished. The primitive you have available for your use locks a single account at a time (e.g., lock(acctnum)). Locks must be held on both accounts while transferring money otherwise errors could result. Even with thousands of accounts, if there are many processes running in parallel, there is a real danger of encountering already locked accounts when requesting locks and, thus, the danger of deadlock.

Describe a scheme that avoids both deadlock and starvation.

A) lock (lower numbered acct); lock(higher numbered acct);

B) P(mutex); lock anything you feel like -- locking is kind of irrelevant; V(mutex);

C) get both locked at once. Here the issue is HOW? Must be able to check if an acct is locked without blocking on it to do so (without issuing a lock call on that acct).

D) variation on above -- lock (first acct); [if (second acct not locked) lock (second acct)] else release(first acct) and retry

Briefly explain (approx. 25 words or less) why it is deadlock free.

A) no circular wait

B) no hold and wait

C) no hold and wait

D) no no preemption

Now consider the case where one of the accounts is not known until data is read from the first account (e.g., a data dependent decision is made). Would your solution work in that case? Would it need to change (and if so, how)?

**A and C need changes...will probably end up with something like D
B works regardless but no parallelism of course.**

COMPSCI 110 Fall 2002

Midterm

Name: _____

5) (real-time scheduling) A real-time system has four periodic events, A , B , C , and D , with periods of 50, 100, 200, and 300msec respectively. Suppose that the four events have a worst case execution time of 30, 20, 10, and x msec.

(a) What is the largest value of x for which the system is schedulable?

$x \leq 45$

(b) With $x = 10$, what is the schedule for the first 300msec under Earliest Deadline First. (hint: drawing a Gantt chart may help significantly to keep details straight).

A B A C D A B A idle A B A C idle

(c) Assuming the processor can run at $\frac{1}{2}$ and $\frac{3}{4}$ of the maximum clock rate (with accompanying voltage reductions). With $x = 10$, how could dynamic voltage/frequency scaling be effective?

Possibility 1: run B at time 130 at $\frac{1}{2}$ speed and C at time 280 at $\frac{1}{2}$ speed

Possibility 2: run C (both instances) and D at $\frac{1}{2}$ speed