CPS 110 Midterm

Monday, October 12

This is an in-class exam. Read each question carefully before you begin working. Don't waste any words: answers are graded on content, not style. I don't expect you to need more than a page for any sub-question.

If you need to make any assumptions that are not clear from the question, then please state them explicitly. For code, any kind of pseudocode is fine as long as its meaning is clear. You may assume standard routines like lists, queues, hash tables, etc.

This is a closed-everything exam. The usual rules apply: use only the permitted sources of information (i.e., your brain), and don't discuss your answers with anyone until the exam is over. Good luck!

Print your name:

Sign for your honor:

Problem	Score
1	
2	
3	
4	
5	
total	

1 Threads (40 pts)

Answer the following true/false questions about the thread library code that your team wrote for project 1. Since true/false questions are inherently error-prone, you get one free "wrong" answer.

(1) Once the thread library is initialized, all user (application) code runs in the context of some thread.

(2) Once the thread library is initialized, all thread library code runs in the context of some thread.

(3) If a thread overflows its stack, an error or exception is generated in the context of that thread.

(4) A correct thread application may execute incorrectly if calls to thread_yield() are added to the user code.

(5) A condition variable may be associated with multiple locks.

(6) Synchronization within the thread library is based on disabling interrupts on the underlying processor hardware.

(7) When a thread calls thread_lock to acquire a lock, it may receive ownership of the lock immediately even if other threads are waiting to acquire the lock.

(8) Calls to acquire and release a given lock always occur in strict alternating order, regardless of the global thread schedule chosen by the scheduler.

(9) Calls to acquire and release a given lock always complete in strict alternating order, regardless of the global thread schedule chosen by the scheduler.

(10) On a call to signal a condition variable (thread_signal), exactly one thread that was waiting on that condition variable (if any) begins running right away.

(11) If a user program causes deadlock, all of its deadlocked threads are blocked within the thread library on thread_lock and/or thread_wait calls..

2 More Threads (50 points)

Answer the following additional questions about the thread library code that your team wrote for project 1. The questions use the term Thread Control Block (TCB) to refer to an object containing data that represents a thread, such as the saved context for a blocked thread.

(1) In some cases it is necessary for thread library code to obtain a pointer to the TCB for the current thread. Give as many distinct examples as you can. For each example, explain in one sentence how the library code obtains the pointer. [**Extra credit**: suggest how you might obtain a pointer to the current thread's TCB on a multiprocessor.]

(2) In some cases it is necessary for thread library code to obtain a pointer to the TCB for some thread other than the current thread. Give as many distinct examples as you can. For each example, explain in one sentence how the library code obtains the pointer.

3 Sweet Tooth (30 pts)

The Dining Philosophers (Larry, Curly, and Moe) are stepping out for the evening. They visit a bar to do shots of jello. Each (non-alcoholic) jello shot costs 36 cents. To buy one, a customer needs a quarter, a dime, and a penny (they don't give change, and they *don't* take American Express). Larry has a pocket full of pennies, Curly a supply of dimes, and Moe a supply of quarters. A wealthy student walks up to the bar and lays down at random two of the three coins needed for a shot. The philosopher with the third coin takes the money and buys a shot. The cycle then repeats. Show how to synchronize the philosophers and the student using locks and condition variables.

For this problem, it is sufficient to write two procedures: donate and consume. Each participant is represented by a thread. Each thread calls the appropriate procedure in a loop, and blocks within that procedure when its participant cannot make progress. It is not necessary to check for usage errors.

```
Donate(int cents_donated)
{

}

Consume(int cents_needed)
{

}
```

4 Sweet Tooth Again (30 pts)

Show how to synchronize the philosophers and the student from Problem 3, again, this time using semaphores.

```
Donate(int cents_donated)
{

}

Consume(int cents_needed)
{

}
```

5 Early Winter (50 pts)

The Dining Professors are attending a conference in Big Sky. To burn off excess calories, professors occasionally take breaks from the meeting for some exercise. We were hoping to get some hiking in, but winter has come early to Montana this year, so we will ski instead. To ski, a professor must acquire at most two skis, at most two boots, and/or at most two poles from a shared collection of ski gear (some of the professors have brought a few pieces of their own gear). Before returning to the meeting, professors release the borrowed gear for their colleagues to use.

Show how to synchronize the professors using locks and condition variables. Your solution should be free from deadlock and starvation. To simplify the problem, you may assume that all professors are the same size, professors do not misbehave, and locks and condition variables have FIFO scheduling behavior.

```
Borrow(int skis, int boots, int poles)
{

}

Return(int skis, int boots, int poles)
{

}
```