

## CS4411 Intro. to Operating Systems Exam 1 Fall 2005

150 points – 10 pages

Name: \_\_\_\_\_

- Most of the following questions only require very short answers. Usually a few sentences would be sufficient. Please write to the point. If I don't understand what you are saying, I believe, in most cases, you don't understand the subject.
- Justify your answer with a convincing argument. If there is no justification when it is needed, you will receive no credit for that question even though you have provided a correct answer. *I consider a good and correct justification more important than a right answer. Thus, if you provide a very vague answer without a convincing argument to show your answer being correct, you will likely receive a low grade.*
- Do those problems you know how to do first. Otherwise, you may not be able to complete this exam on time.



**2. Fundamentals of Processes and Threads**

(a) **[15 points]** Draw the state diagram of a process from its creation to termination, including all transitions, and briefly elaborate **every state** and **every transition**.

(b) **[10 points]** What is a *context*? Provide a detailed description of *all* activities in a *context switch*.

(c) **[10 points]** What is *thread cancellation*? Define the term and discuss the two commonly used versions and their differences.

### 3. Synchronization

- (a) [10 points] Define the meaning of *race condition*? Answer the question first and use an execution sequence to illustrate your answer. **You will receive no credit if only an example is provided without an elaboration.**
- (b) [5 points] Programming Assignment #1 has three steps: **(1)** thread  $T_i$  ( $0 \leq i < n$ ) initializes array element  $w[i]$  to 1; **(2)** threads  $T_{i,j}$  writes a 0 into  $w[i]$  (*resp.*,  $w[j]$ ) if  $x_i < x_j$  (*resp.*,  $x_i > x_j$ ) holds, where  $x_i$ 's are input values; and **(3)** thread  $S_i$  prints  $x_i$  if  $w[i]$  is 1. Of these three steps, which step or steps could have potential race conditions? **You should provide a clear answer with a convincing argument. Otherwise, you will receive no credit.**
- (c) [10 points] The methods `Wait()` and `Signal()` must be atomic to ensure a correct implementation of mutual exclusion. Use an execution sequence to show that if `Wait()` is not atomic then mutual exclusion cannot be maintained. **You must use an execution sequence as we did many times in class to present your answer. Otherwise, you risk low or no credit.**

- (d) [10 points] Consider the solution to the mutual exclusion problem for two processes  $P_0$  and  $P_1$ , where `flag[2]` is a Boolean array of two elements and `turn` is an integer variable with an initial value 0 or 1. Both are global to processes  $P_0$  and  $P_1$ . Show, with a step-by-step execution sequence table, that mutual exclusion is implemented incorrectly. **You will risk low or very low grade if you do not use an execution sequence table.**

```
bool flag[2]; // global flags
int turn; // global turn variable

Process i (i = 0 or 1)

flag[i] = TRUE; // I am interested
while (turn != i) { // while it is not my turn
    while (flag[j]) // while you are interested
        ; // do nothing: busy waiting
    turn = i; // because your are not interested, it is my turn
}
// critical section
flag[i] = FALSE // I am done and not interested
```

## 4. [25 points] Problem Solving: I

- (a) A multithreaded program has two global arrays and a number of threads that execute concurrently. The following shows the global arrays, where  $n$  is a constant defined elsewhere (*e.g.*, in a `#define`):

```
int a[n], b[n];
```

Thread  $T_i$  ( $0 < i \leq n-1$ ) runs the following (pesudo-) code, where function `f()` takes two integer arguments and returns an integer, and function `g()` takes one integer argument and returns an integer. Functions `f()` and `g()` do not use any global variable.

```
while (not done) {
    a[i] = f(a[i], a[i-1]);
    b[i] = g(a[i]);
}
```

More precisely, thread  $T_i$  passes the value of `a[i-1]` computed by  $T_{i-1}$  and the value of `a[i]` computed by  $T_i$  to function `f()` to compute the new value for `a[i]`, which is then passed to function `g()` to compute `b[i]`.

Declare semaphores with proper initial values, and add `Wait()` and `Signal()` calls to thread  $T_i$  to make sure it will compute the result correctly. The syntax is unimportant. For example, you can declare and initialize a semaphore `S` with “`Sem S = 1`” and use `Wait(S)` and `Signal(S)` for semaphore wait and semaphore signal. Your implementation should not have any busy waiting, race condition, and deadlock.

**A convincing correctness argument is needed. Otherwise, you will receive no credit for this problem.**

Use this and next page for your answer

Use this blank page for your answer

## (b) [25 points] Problem Solving: II

There are two groups of threads  $A$  and  $B$  with unspecified number of threads in each group. Two threads from group  $A$  and one thread from group  $B$  are required to establish a *rendezvous* in order to perform a specific task. Thus, each thread has the following execution pattern:

```
thread-in-A(....)           thread-in-B(....)
{                             {
    while (1) {               while (1) {
        // does something     // does something
        A-rendezvous(..);    B-rendezvous(..);
        // does something else // does something else
    }                         }
}
```

When a thread in group  $A$  (*resp.*,  $B$ ) reaches the rendezvous point, it calls function `A-rendezvous()` (*resp.*, `B-rendezvous()`). This function blocks the calling thread until a rendezvous of two- $A$ -one- $B$  can be made. Once such a rendezvous occurs, two threads in  $A$  and one thread in  $B$  return from `A-rendezvous()` and `B-rendezvous()`, respectively.

Use semaphores to write functions `A-rendezvous()` and `B-rendezvous()`. The syntax is unimportant. For example, you may declare and initialize a semaphore  $S$  with “Sem  $S = 1$ ” and use `Wait(S)` and `Signal(S)` for semaphore wait and semaphore signal. Your implementation should not have any busy waiting, race condition, and deadlock.

Use this and next page for your answer

Use this blank page for your answer

## Grade Report

<i>Problem</i>		<i>Possible</i>	<i>You Received</i>
1	a	8	
	b	8	
	c	8	
	d	6	
2	a	15	
	b	10	
	c	10	
3	a	10	
	b	5	
	c	10	
	d	10	
4	a	25	
	b	25	
<b>Total</b>		150	