Name: ________________________________

Rules and Hints

- You may use one handwritten 8.5 × 11” cheat sheet (front and back). This is the only additional resource you may consult during this exam. No calculators.

- Include step-by-step explanations and comments in your answers, and show as much of your work as possible, in order to maximize your partial credit.

Grade

<table>
<thead>
<tr>
<th>Problem</th>
<th>Your Score</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1: Definitions</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Problem 2: Concurrency</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Problem 3: Synchronization</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Problem 4: Resource management</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Problem 5: Deadlock</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Problem 6: Scheduling and queues</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
Problem 1: Definitions (15 points)

Define the following terms as they relate to this course.

Atomic:

Race condition:

Mutual exclusion:

Deadlock:

Response time [of a task]:
Problem 2: Concurrency (15 points)

Consider the following operations on a shared array:

Thread 1

\[ x[0]++; \]
\[ x[1] = 3; \]

Thread 2

\[ x[0] = 8; \]
\[ x[1] += 2; \]

Assuming that the array elements are the possible values of \( x[0] \) and \( x[1] \) after these operations? List each possible pair of values \((x[0], x[1])\).

If we modify the code as follows, using a shared variable \textit{flag} that is initialized to 0:

Thread 1

\begin{verbatim}
while (flag);
flag = 1;
x[0]++; 
x[1] = 3;
flag = 0;
\end{verbatim}

Thread 2

\begin{verbatim}
while (flag);
flag = 1;
x[0] = 8;
x[1] += 2;
flag = 0;
\end{verbatim}

What are the possible values \((x[0], x[1])\) now? Explain.
Problem 3: Synchronization (20 points)

Consider the following function, which is run by multiple threads. All variables that are not declared within the function are shared. The function \( g \) does not read or modify any shared state.

```c
void* f(void * thread_id) {
    long tid = (long) thread_id;

    pthread_mutex_t m;
    pthread_mutex_init(&m);
    // pthread_mutex_lock(&m);
    for (int i = 0; i < 1000; i++)
        if (g(i, tid) > 0) count++;
    // pthread_mutex_unlock(&m);

    pthread_mutex_destroy(&m);
}
```

With the current version of the code (where the lock and unlock operations have been commented out), is the value of \( \text{count} \) guaranteed to be the same as it would be if \( f() \) were called sequentially? Explain.

If the mutex operations are put back into the code, is the value of \( \text{count} \) guaranteed to be the same as it would be if \( f() \) were called sequentially? Explain.

Modify the code to maximize its performance and to correct any remaining bugs. The next page has a fresh copy with more space.
void* f(void * thread_id) {

    long tid = (long) thread_id;

    pthread_mutex_t m;

    pthread_mutex_init(&m);

    // pthread_mutex_lock(&m);

    for (int i = 0; i < 1000; i++)

        if (g(i, tid) > 0) count++;

    // pthread_mutex_unlock(&m);

    pthread_mutex_destroy(&m);

}
Problem 4: Resource management (20 points)

You are writing a program for bouncers to use at the two entrances of a club. Only 200 people are allowed to be in the club at any given time.

Your task is to define the following two functions, which should update a shared variable count. Neither function has any parameters or returns anything.

- The enter function will be called when the bouncer checks the ID of a person trying to enter. Its task should be to call a function ok() exactly once when it is OK for that person to enter the club.

- The leave function will be called when the bouncer sees a person leave.
Problem 5: Deadlock (10 points)

Consider the following resource utilization matrix, where P1, P2, etc. are processes and R1, R2, etc. are resources. An “H” in a given (row, column) means that the process in that row currently holds the resource in that column. A “W” means that the process is currently waiting for that resource.

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>H</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>H</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>W</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td></td>
<td></td>
<td></td>
<td>H</td>
</tr>
</tbody>
</table>

If there is only one of each resource R1...R4 in the system and we cannot make processes give up resources, is this situation a deadlock? Explain.

Make a minimal change to the graph to either create a deadlock (if none exists) or resolve the deadlock (if one does).
Problem 6: Scheduling and queueing (20 points)

Consider the following tasks:

<table>
<thead>
<tr>
<th>Task</th>
<th>Arrives at $t =$</th>
<th>Needed CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>15</td>
<td>30</td>
</tr>
</tbody>
</table>

What are the completion and response times of each task under a FIFO scheduler?

What are the completion and response times of each task under a shortest-job-first scheduler whose timeslice is 10 time units?

Assume that the service times for these tasks are representative of all tasks handled by this system. What is the maximum sustainable arrival rate?