This is a closed book (and notes) examination. Answer all questions on the exam itself. Take the number of points assigned to each problem and the amount of space provided for your answer as a measure of the length and difficulty of the expected solution. The exam totals 100 points.
1. (10 points) A programmer creates an application program.

   (a) What is the relationship between this application program and an operating system process?

   (b) How does the application program interface with the operating system to make requests for system services?

2. (12 points) Explain the difference between a preemptive and a non-preemptive process scheduling policy. Why do all interactive systems use preemptive policies even though these policies have a higher overhead?
3. (12 points) In class we discussed the use of `BeginRegion()` and `EndRegion()` to indicate the beginning and end of a critical region. Show the implementation for these routines using semaphores. Be sure to indicate semaphore initialization.

**Initialization:**

```c
BeginRegion()
{
}

EndRegion()
{
}
```

4. (12 points) Assume three processes have the start and execution time requirements shown in the table. Assume the time units are in quantums where a quantum is the time-slice for a round-robin scheduling policy. For each policy given, show which process will execute during each quantum of time (assume time for context switching is negligible).

<table>
<thead>
<tr>
<th>Process</th>
<th>Start Time</th>
<th>Execution Time</th>
<th>Priority (larger number, higher priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

First Come, First Serve

Time → 0 1 2 3 4 5 6 7 8 9 10 11 12

Priority Scheduling using Round Robin

Time → 0 1 2 3 4 5 6 7 8 9 10 11 12
5. (12 points) Assume the process table in an operating system contains room for 10 entries (not including an idle process). What is the minimum and maximum number of processes (out of the 10) that can be in each of the three states on a uni-processor machine? Your answer should consider the state of processes after the scheduler has run. Justify your answer as needed.

<table>
<thead>
<tr>
<th>state</th>
<th>minimum number</th>
<th>maximum number</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ready</td>
<td></td>
<td></td>
</tr>
<tr>
<td>blocked</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. (13 points) In Project 1, you used various system calls for executing a command and gathering statistics about its execution.

(a) Is it possible that a call to fork() can fail? If it cannot fail explain why, if it can fail give an example of when it would fail.

(b) Is it possible that a call to execve() (execvp()) can fail? If it cannot fail explain why, if it can fail give an example of when it would fail.

(c) How does the wait() call work?

(d) How can a program determine the amount of CPU time consumed by the command?

(e) How can a program determine the response (wall-clock) time to execute the command?
7. (8 points) Why is the use of a spin lock often used for mutual exclusion on a multiprocessor, but not on a uniprocessor?

8. (11 points) A semaphore has a counter and a queue associated with it.

(a) What are the counter and queue used for? Specifically what is the meaning if the counter has a value of -2.

(b) Could the queue be replaced by a stack? What advantages and disadvantages would this have?
9. (10 points) A set of 10 processes are created to run. It is desired that up to three of these processes at a time can access a shared resource. Describe how semaphores can be used by these processes to solve this problem.

Now suppose we are on a system where we only have message passing between processes (no semaphores or other IPC mechanism). Is it possible to coordinate the processes as desired using message passing? If so indicate how. If not then indicate why it cannot be done with message passing.