Name: ____________________________
Student ID:_______________________

Write your test number on all pages because the pages will be separated for grading.

No books, no notes, but calculators are allowed. If you need to make an assumption to solve a problem, state the assumption.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/10</td>
</tr>
<tr>
<td>2</td>
<td>/5</td>
</tr>
<tr>
<td>3</td>
<td>/10</td>
</tr>
<tr>
<td>4</td>
<td>/10</td>
</tr>
<tr>
<td>5</td>
<td>/10</td>
</tr>
<tr>
<td>6</td>
<td>/30</td>
</tr>
<tr>
<td>7</td>
<td>/25</td>
</tr>
<tr>
<td>Total</td>
<td>/100</td>
</tr>
</tbody>
</table>
1. 10 pts. For each of the following, specify whether it results in an Interrupt (I), Exception (trap) (E), or Neither (N):

(a) Test and Set instruction
(b) Disk read completed
(c) Attempt to write to read-only memory
(d) System call (fork, for example)
(e) Call to library routine (strcpy, for example)


2. 5 pts. The Banker’s algorithm deals with deadlock via:

(a) Deadlock Avoidance
(b) Deadlock Prevention
(c) Deadlock Detection and Recovery

Solution: (A): Stay safe. The wrong answers: (B) works by getting rid of one of the four conditions necessary for deadlock (C) will wait until it occurs.

3. 10 pts. If there are $n$ separate process, each with its own address space and a page frame size of $p$ bytes, what is the expected amount of space lost due to internal fragmentation?

Solution: $n \times \frac{p}{2}$. Each process wastes half a page, on average, at the end of its address space.

4. 10 pts. Give a short description of the terms race condition and deadlock.

Solution: A race condition causes different results from accessing shared memory depending on the interleaving of the instructions of multiple processes/threads.

Deadlock consists of a cycle of processes each holding a resource, and waiting for a resource the next process has.

5. 10 pts. If the cost of accessing the TLB is 20 ns. and of accessing main memory is 200 ns., what is the minimum TLB hit rate (percentage of
time a lookup is found in the TLB) necessary in order to achieve an
effective memory access time of 260 ns? Assume a single-level page
table and a serial algorithm.

Solution: Let \( x \) be the TLB hit rate. \( x \cdot (20 + 220) + (1 - x)(20 + 
200 + 200) = 260. \) Thus, \( 200x = 160 \) or \( x = 0.8. \) The \( 20 + 200 + 200 \) is
composed of 20ns for the missed TLB lookup, 200 ns to lookup the
PTE, and 200 ns to access the physical memory.

6. 30 pts. The following table lists the arrival time, execution time, and
priority (higher number means greater priority) of 5 jobs.

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrival time</th>
<th>Execution time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>60</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>100</td>
<td>60</td>
<td>2</td>
</tr>
</tbody>
</table>

Give the start time (the time the job is first scheduled; note that a job
may have to wait when it arrives), the end time, the wait time, and
the turnaround time of each of the jobs using each of the following
scheduling algorithms.

(a) Shortest Job First (without preemption)

<table>
<thead>
<tr>
<th>Job</th>
<th>Start time</th>
<th>End time</th>
<th>Turnaround</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>30</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>60</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>60</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>80</td>
<td>120</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>E</td>
<td>120</td>
<td>180</td>
<td>60</td>
<td>0</td>
</tr>
</tbody>
</table>

Solution:

(b) Priority (without preemption)

<table>
<thead>
<tr>
<th>Job</th>
<th>Start time</th>
<th>End time</th>
<th>Turnaround</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>30</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>30</td>
<td>70</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>70</td>
<td>100</td>
<td>70</td>
<td>40</td>
</tr>
<tr>
<td>E</td>
<td>100</td>
<td>160</td>
<td>80</td>
<td>20</td>
</tr>
<tr>
<td>D</td>
<td>160</td>
<td>180</td>
<td>120</td>
<td>100</td>
</tr>
</tbody>
</table>

Solution:

(c) Round-Robin with a quantum of 20
**Solution:** Here are the assigned timeslices (in order):

A(20)-B(20)-A(10)-C(20)-B(20)-D(20)-C(20)-E(20)-E(20)-E(20)

<table>
<thead>
<tr>
<th>Job</th>
<th>Start time</th>
<th>End time</th>
<th>Turnaround</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>50</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>90</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>C</td>
<td>50</td>
<td>120</td>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>D</td>
<td>90</td>
<td>110</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>E</td>
<td>120</td>
<td>180</td>
<td>80</td>
<td>20</td>
</tr>
</tbody>
</table>

7. **25 pts.** We have two processes which each repeatedly execute two sections of code, and then increment a shared variable:

```java
shared Integer numIterations = 0;
```

Process A

```java
loop begin
  A1;
  A2;
  numIterations++;
loop end;
```

Process B

```java
loop begin
  B1;
  B2;
  numIterations++;
loop end;
```

We want to satisfy the following constraints:

(a) Statement A2 in the \(i\)th iteration of A’s loop cannot execute until statement B1 executes in the \(i\)th iteration of B’s loop

(b) Statement B2 in the \(i\)th iteration of B’s loop cannot execute until statement A1 executes in the \(i\)th iteration of A’s loop

(c) numIterations must always maintain the number of loops process A has completed plus the number of loops process B has completed
Add to the existing code to satisfy the given constraints, but without adding additional constraints (for example, it shouldn’t matter whether A1 or B1 executes first).

You may declare additional shared or local variables of type Integer, Boolean, or Semaphore, but make sure to give them initial values.

Solution:

```plaintext
shared Integer numIterations = 0;
shared Semaphore mutex(1), a1Done(0), b1Done(0);
```

Process A

```plaintext
loop begin
    A1;
    a1Done.V();
    b1Done.P();
    A2;
    mutex.P();
    numIterations++;
    mutex.V();
loop end;
```

Process B

```plaintext
loop begin
    B1;
    b1Done.V();
    a1Done.P();
    B2;
    mutex.P();
    numIterations++;
    mutex.V();
loop end;
```