CSE 120: Midterm

July 13, 2005—Day 5

Name: _____

Student ID:_____

Write your name 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.

Problem	Score
1	/5
2	/2
3	/2
4	/2
5	/4
6	/5
7	/5
8	/10
9	/10
10	/15
11	/15
Total	/75

Name: _____

1. *5 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)

   (a): N, (B): I, (C): E, (D): E, (E): N

2. *2 pts.* Which of the following is the best definition for *thrashing*?

   (a) State where the CPU utilization is low because each process spends little CPU time before making an I/O call.

   (b) State where the system spends much more time paging than actually executing processes.

   (c) State where the high degree of multiprogramming causes the turnaround time of processes to be high due to the limited amount of CPU time available to each process.

   (B); more time paging than actually executing processes.

3. *2 pts.* Once the system detects thrashing, what can it do to eliminate the problem?

   (a) Decrease the degree of multiprogramming

   (b) Increase the priority of CPU-bound jobs

   (c) Increase the degree of multiprogramming

   (d) Decrease the priority of CPU-bound jobs

   (e) none of the above

   (A) (B) will only change which processes are thrashing. (C) will make it worse. (D) will only change which processes are thrashing.

4. *2 pts.* The Banker's algorithm deals with deadlock via:

(a) Deadlock Avoidance

(b) Deadlock Prevention

(c) Deadlock Detection and Recovery

(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.

5. *3 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?

   n*p/2. Each process wastes half a page, on average, at the end of its address space.

6. *5 pts.* What is the difference between a race condition and deadlock?

   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.

7. *5 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.

Let x be the TLB hit rate. $x*(20+220)+(1-x)(20+200+200)=260$. Thus, $200x=160$ or $x=.8$. The $20+200+200$ is composed of 20ns for the missed TLB lookup, 200 ms to lookup the PTE, and 200 ms to access the physical memory.

8. *10 pts.* Using the reference string <2 1 0 3 2 1 4 2 1 0 3 4>, fill in the two tables below (representing three and four page frames respectively) using the FIFO page replacement policy. How many page faults occur in each case?

	2*	2	2	3*	3	3	4*	4	4	4	4	4
		1*	1	1	2*	2	2	2	2	0*	0	0
			0*	0	0	1*	1	1	1	1	3*	3

Total of 9 page faults

	2*	2	2	2	2	2	4*	4	4	4	3*	3
		1*	1	1	1	1	1	2*	2	2	2	4*
			0*	0	0	0	0	0	1*	1	1	1
				3*	3	3	3	3	3	0*	0	0

Total of 10 page faults

9. *10 pts.* Using the same reference string as in the previous question (<2 1 0 3 2 1 4 2 1 0 3 4>, fill in the two tables below (representing three and four page frames respectively) using the LRU page replacement policy. How many page faults occur in each case?

	2*	2	2	3*	3	3	4*	4	4	0*	0	0
		1*	1	1	2*	2	2	2	2	2	3*	3
			0*	0	0	1*	1	1	1	1	1	4*

Total of 10 page faults

	2*	2	2	2	2	2	2	2	2	2	2	4*
		1*	1	1	1	1	1	1	1	1	1	1
			0*	0	0	0	4*	4	4	4	3*	3
				3*	3	3	3	3	3	0*	0	0

Total of 8 page faults

10. *15 pts.* The following table lists the arrival time, execution time, and priority (higher number means greater priority) of 5 jobs.

Job	Arrival time	Execution time	Priority
A	0	30	3
B	20	40	5
C	30	30	4
D	60	20	1
E	100	60	2

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

(a) Shortest Job First

A start: 0, end: 30
C start: 30, end: 60
D start: 60, end: 80
B start: 80, end: 120
E start: 120, end: 180

(b) Priority

A start: 0, end: 30
B start: 30, end: 70
C start: 70, end: 100
E start: 100, end: 160
D start: 160, end: 180

(c) Round-Robin with a quantum of 20

Three possibilities got full credit:

1) When a running job's quantum expires at the same time another job enters an empty queue: the entering job runs.

2) When a running job's quantum expires at the same time another job enters an empty queue: the running job runs again.

a) Newly entered job enter at the beginning of the queue

b) Newly entered jobs enter at the end of the queue,

Here's 1a:

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

A start: 0, end: 90
B start: 20, end: 110
C start: 40, end: 120
D start: 60, end: 80
E start: 120, end: 180

Here's 1b:

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

A start: 0, end: 50
B start: 20, end: 90
C start: 50, end: 120
D start: 90, end: 110
E start: 120, end: 180

Here's 2a:

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

A start: 0, end: 30
B start: 30, end: 110
C start: 50, end: 120
D start: 70, end: 90
E start: 120, end: 180

Here's 2b:

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

A start: 0, end: 30
B start: 30, end: 90
C start: 50, end: 120
D start: 90, end: 110
E start: 120, end: 180

11. *15 pts.* We have two processes which each repeatedly execute two sections of code, and then increment a shared variable :

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

```
shared Semaphore mutex(1), a1Done(0), b1Done(0);
```

Process A

```
loop begin
     A1;
       a1Done.up();b1Done.down()
     A2;
   mutex.down();
    numIterations++;
      mutex.up();
loop end;
```

Process B

```
loop begin
     B1;
         b1Done.up(); a1Done.down
     B2;
   mutex.down();
    numIterations++;
       mutex.up();
loop end;
```

We want to satisfy the following constraints:

(a) Statement A2 in the ith iteration of A's loop cannot execute until statement B1 executes in the ith iteration of B's loop

(b) Statement B2 in the ith iteration of B's loop cannot execute until statement A1 executes in the ith 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.