CSC 369H1 F
OPERATING SYSTEMS

UNIVERSITY OF TORONTO
FALL 2003

Midterm Test

NO AIDS ALLOWED

Please **<u>PRINT</u>** in answering the following requests for information:

Family Name: _____

Given Names: _____

Student Number: |_ _ _| |_ _ _| |_ _ _|

Login (@cdf): _____

Notes to students:
1. This test lasts for 50 minutes and consists of 50 marks. Budget your time accordingly.
2. This test has 7 pages and 5 questions; Check that you have all pages before starting.
3. **Write in pen. No pencils.**
4. Write your answers on this "question and answer" paper, in the spaces provided. Be <u>concise</u>. In general, the amount of space provided is an upper bound on the "size" of answer that is expected. If necessary, use space where available and provide explicit pointers.
5. In general, state your assumptions and show your intermediate work, where appropriate.
6. Do **not** go beyond here until instructed to do so. Write your student number at the top of each succeeding page once you get going.

Question	Marks
1	
2	
3	
4	
5	
Total	

VERSION A

1. [10 marks, 2 each] True/False

Circle "T" if the statement is *always* true. Otherwise circle "F".

a) In paging systems, external fragmentation cannot occur.   **T / F**

b) Race conditions cannot occur on a uniprocessor.   **T / F**

c) SJF can be implemented as a priority algorithm, where the priority is   **T / F**
determined by the arrival time of the job.

d) A process in the *Ready* state can only transition to *Running* or *Exit*   **T / F**
states.

e) The two-phase locking protocol guarantees that concurrent transactions   **T / F**
are deadlock-free.

2. [9 marks; 3 each]
Define the following terms in the context of this course:

(a) Starvation:

(b) Page Frame

(c) Turnaround Time

3. [10 marks; 2 each]   Measurements of a certain system have
shown that the average process runs for a time T before blocking on I/O. A process
context switch requires a time S, which is effectively wasted (overhead). The
performance measure of interest is <u>CPU efficiency</u>, defined as the ratio of useful CPU
time over total CPU time. For round-robin (RR) scheduling with a quantum of length Q,
give a formula for CPU efficiency in each of the following cases (be as specific as
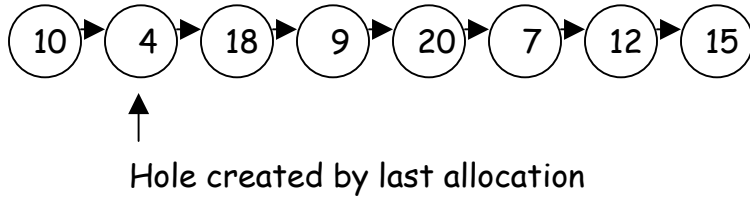possible, give a range if appropriate):

  a) $Q = \infty$

  b) $Q > T$

  c) $S < Q < T$

  d) $Q = S$

  e) Q nearly 0 (or tending to 0)

4. [10 marks; breakdown as given   below]

Consider a dynamic partitioning system in which memory consists of the following holes, sorted by increasing memory address (all sizes are in Kilobytes):

$$10 \rightarrow 4 \rightarrow 18 \rightarrow 9 \rightarrow 20 \rightarrow 7 \rightarrow 12 \rightarrow 15$$

↑

Hole created by last allocation

(a) **[8 marks]** Suppose a new process requiring 11 kB arrives, followed by a process needing 9 kB of memory. **Show** the list of holes **after both** these processes are placed in memory for each of the following algorithms (start with the original list of holes for each algorithm).

   i)   First Fit:

   ii)   Worst Fit:

   iii)   Best Fit:

   iv)   Next Fit:

(b) **[2 marks]** Keeping the hole list sorted by size can make some of the allocation algorithms faster. **State** one advantage of keeping the list sorted **by address** instead.

5. [11 marks]

Following an all-candidates meeting at city hall, politicians and reporters need to take an elevator that will only carry exactly 3 people at a time down to the ground floor to exit the building. (The elevator can come back up again empty). The elevator must carry 2 politicians and 1 reporter on each trip—politicians don't want to be outnumbered by reporters, and reporters don't want politicians to leave without answering any questions. There should be no unnecessary waiting: politicians and reporters should not wait if there is enough of them ready to form a "safe" elevator load.

A politician calls the procedure *PoliticianArrives* and a reporter calls the procedure *ReporterArrives* when they are ready to use the elevator. The procedures arrange the arriving politicians and reporters into "safe" elevator loads; once a full load is ready, one thread calls *EnterElevator* and after the call to *EnterElevator*, all three threads can leave. Assume that the *EnterElevator* procedure correctly blocks the caller until the elevator is ready for another load of passengers. Assume also that there are exactly twice as many politicians as reporters at the meeting.

Below is a semaphore-based implementation, using Posix semaphores and pthreads to represent politicians and reporters. Assume the semaphore waiting queues are FCFS.

```
/* Semaphores are initialized to 0 */
sem_t politician_ready;                      void *ReporterArrives() {
sem_t reporter_ready;                           politician_ready.wait();
                                                politician_ready.wait();
void *PolitionArrives() {                       EnterElevator();
   politician_ready.post();                     reporter_ready.post();
   reporter_ready.wait();                       reporter_ready.post();
   pthread_exit(0);                             pthread_exit(0);
}                                            }
```

(a) **[2 marks] State** whether the implementation <u>works</u> (it is correct) or <u>does not work</u> (it is either always wrong or there exists an execution scenario where it is wrong).

(b) **[9 marks] Justify** your answer to (a) by **either** (i) **proving** that the implementation is correct, **OR** (ii) **showing** an execution scenario where the implementation is not correct and suggest a way to fix it (use the next page for your answer).

Total marks = (50)
End of test