NAME: ________________________________ (FIRST NAME FIRST) **SCORE**: ________

COSC 4330 SECOND MIDTERM OCTOBER 30, 2007

*This exam is **closed book**. You can have **one page** of notes. UH expels cheaters.*

1. ***Questions with short answers:*** **(**4×5 points)

a) How do you pass a ***linked list*** to a ***remote procedure***?

By packing into an array and passing it by value.

b) What is the major disadvantage of ***busy waits***?

They waste CPU cycles and cause many context switches.

c) What is the difference between ***blocking sends*** and ***non-blocking sends***?

A blocking send does not return until the message is received by the destination process while a non-blocking send returns immediately..

d) What is the major advantage of ***idempotent procedures***?

We do not have to prevent multiple executions of an idempotent procedure as they will always have the same effect as a single execution of the procedure.

e) What is the major disadvantage of the ***at-most-once*** semantics for remote procedure calls?

It does not prevent partial executions of the remote procedure.

f) Name an application that should use ***streams*** rather than ***datagrams*** and ***explain why***?

Any web browser, any file transfer application.

2. What is wrong with the following solution to the mutual exclusion problem? (10 points)

```
// process ids must be 0 or 1
shared int inside[2] = {1, 1};

void enter_region(int pid) {
    int other = 1- pid;
    while (inside[other] == 2);
    inside[pid] = 2;
} // enter_region

void leave_region(int pid) {
    inside[pid] = 1;
} /}// leave_region
```

Mark the correct answer(s):

■ It does not guarantee mutual exclusion under all circumstances.

O It will sometimes prevent a process to enter the critical section when no other process is inside,

O It will occasion deadlocks.

T: ______

3. Consider the following System V Release 4 scheduler where the question marks represent new priority levels: (2×10 points)

```
#ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait LEVEL
     800        ?        ?       16000        ?    #  0
     400        ?        ?        8000        ?    #  1
     200        ?        ?        4000        ?    #  2
     100        ?        ?        2000        ?    #  3
```

a) What is the ***best value*** for the `ts_tqexp` parameter at ***priority level* 1** and ***why*?**

0 since we should lower the priority of processes that exceeded their time slice.

b) What is the ***best value*** for the `ts_slpret` parameter at ***priority level* 2** and ***why*?**

3 since we should increase the priority of processes that have performed a system call.

4. For each of the statements below, indicate ***in one sentence*** whether the statement is true or false (2 points), *and why* (3 points).

a) Windows NT dynamically adjusts the priorities of real-time processes to ensure that these processes always complete on time.

FALSE, Windows NT uses fixed priorities for all real-time processes.

b) In a RPC, one of the tasks of the ***user stub*** is to exchange messages with the user program.

FALSE, the user stub exchanges messages with the server stub.

c) Peterson's algorithm assumes the existence of a test-and-set instruction.

FALSE, Peterson's algorithm makes no assumptions about the computer instruction set.

d) You can simulate ***non-blocking receives*** using ***blocking receives*** and ***busy waits***.

FALSE, you cannot simulate non-blocking primitives using blocking primitives.

5. A diner can seat a maximum of 40 customers. Complete the following code fragments to ensure (a) that the diner will never contain more than 40 customers and (b) that customers arriving together will be seated together. (*Hint: Your solution will seat all groups of arriving customers in strict FCFS order.*) (20 points).

```
semaphore seats = _40__;

semaphore access = __1__;

arrive (int ncustomers) {

   int i;

   P(&access) ____________________________________________;

   for (i = 0; i < ncustomers ; i++)

         P(&seats) ________________________________________;

   V(&access) ____________________________________________;

} // arrive

leave(int ncustomers) {

   int i;

   for (i = 0; i < ncustomers ; i++)

         V(&seats) ________________________________________;

} // leave
```