NAME: _____ (FIRST NAME FIRST) SCORE: _____

COSC 4330 SECOND MIDTERM NOVEMBER 4, 2009

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

1. Which of the following statements are ***true*** or ***false*** (2 points) and ***why***? (3 points)

a) You cannot combine non-blocking sends with blocking receives.

False, this is the default for BSD socket calls send() and receive().

b) Peterson's algorithm for mutual exclusion assumes the existence of a ***test-and-set*** instruction.

False, Peterson's algorithm makes no assumption about the hardware on which it will run.

c) Good programmers always put their ***notify*** operations at the end of their monitor procedures.

False, notify operations can be put anywhere in your code as they never release the monitor.

d) In a RPC package, 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.

e) Messages are ***reusable resources*** since they can be forwarded to other computers.

False, they are consumed as soon as they have been received.

f) Datagrams preserve message boundaries.

True, messages are sent and received individually.

2. Consider the following solution to the mutual exclusion problem and explain when it fails (5 points) and what happens then. (5 points)

```
shared int reserved[2] = {0, 0}; // global variable

void enter_region(int pid) {  // pid will always be 0 or 1
     int other;
     other = 1 - pid; // pid of other process
     reserved[pid] = 1; // reserve
     while (reserved[other] && reserved[pid]);   // busy wait
} // enter_region

void leave_region(int pid) {
     reserved[pid] = 0;
} // leave_region
```

When two processes try to enter the critical section in lockstep

then a deadlock occurs

T: _____

3. Consider the function

```
void doublesquare(int *pa, int *pb) {
     *pa *= *pa; *pb *= *pb;
} //  doublesquare
```

and assume the following calling sequence:

```
alpha = 3; doublesquare (&alpha, &alpha);
```

What will be the value of **alpha** ***after the call*** assuming that

a) the call was a ***conventional procedure call***? (5 points) **Answer: alpha =** eighty-one

b) the call was a ***remote procedure call***? (5 points) **Answer: alpha =** nine

4. Give an example of an application where

a) Datagrams are more indicated than streams. (5 points)

Short requests from a client to a server

b) Streams are more indicated than datagrams. (5 points)

Transferring a large file

5. What is the major disadvantage of *busy waits*? (5 points) What can we do to eliminate them? (5 points) Are there cases where it is better to keep them? (5 points)

a) Busy waits waste CPU cycles and cause unnecessary context switches.

b) We should use instead kernel supported solutions that move the waiting processes to the waiting state.

c) It is better to keep them in multiprocessor architectures whenever the cost of the wait is less than the cost of the context switches required to bring the waiting process to the waiting state and back.

6. A classroom has two doors and one switch at each door to turn the lights on and off. Complete the following template to ensure that (a) the room will never contain more than 40 people and (b) the light switches will always be on when there are people in the room and off when the room is empty. (5 points per correct line for a total of 25 points)

```
shared int people_count = 0;
semaphore mutex = 1;
semaphore room  = __40__;
room(){
    P(&mutex); ____________________________________________
    people_count++;
    if(people_count == 1) toogle(switch);
    V(&mutex); P(&room); ____________________________________
    stay_in_room();
    P(&mutex); ____________________________________________
    people_count--;
    if(people_count == 0)
    V(&mutex); P(&room); ____________________________________
} // room
```

Note:

The problem is a variant of the readers and writers problem discussed in class. Here too, we use a mutex to ensure that people enter or leave the classroom one by one. The big difference is that we do here a P(&room) each time a person enter and a V() each time a person leaves the room in order control the number of people in the room.