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

1. **Questions with short answers:** (4×5 points)
   
a) What would be the major disadvantage of using *non-blocking receives* to build a server?

   The server will waste cycles doing busy waits between requests.

   
b) What is the easiest way to implement the *at most once semantics* in remote procedure calls?

   Attach a serial number to each request and instruct the server to reject duplicates of previous requests.

   
c) What is the difference between *virtual circuits* and *streams*?

   Virtual circuits preserve message boundaries; streams do not.

   
d) When are *busy waits* the best choice?

   In multicore/multiprocessor architectures when a process waits for a process running on another processor and the wait is expected to be short.

2. Consider the instruction **TSET R7, LOCK** and assume it is used to ensure mutual exclusion within a critical section. Assuming that the variable **LOCK** can only be equal to zero or one, what are the two possible values for **R7** after the instruction is executed and their meanings? (2×5 points)

   a) If **R7** equals **0** then the process can enter the critical section.

   b) If **R7** equals **1** then the process cannot enter the critical section.
3. For each of the statements below, indicate in one sentence whether the statement is true or false (2 points), and why (3 points).

a) Making all remote procedures idempotent greatly simplifies the task of the RPC server.

   TRUE, we do not have to worry about multiple executions of the same procedure call.

b) A blocking send is the same as a buffered send.

   FALSE, a buffered send is the same as a non-blocking send.

c) The all or nothing semantics guarantees that all remote procedure calls will be executed at least once.

   FALSE, the all or nothing semantics guarantees that all remote procedure calls will be executed exactly once or not at all.

d) We can simulate a blocking receive with a non-blocking receive inside a busy wait loop.

   TRUE, think of while (non_blocking_read(...) == NO_MESSAGE);

e) Peterson’s algorithm requires busy waits.

   TRUE, it contains an empty while loop.

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

   FALSE, user stubs exchange messages with the server stub.
4. Complete the following template to obtain a correct solution to the mutual exclusion problem for two processes whose ID’s are either 0 or 1? (5×4 points)

```c
shared int requested[2] = {0, 0};

shared int turn;

void enter_region(int mypid) {
    requested[_____ mypid_____] = _____ 1 _____;
    turn = ___ 1 - mypid ____;
    while (requested[____ 1 - mypid_____] && turn != mypid);
} // enter_region

void leave_region(int mypid) {
    requested[______ mypid_____] = 0;
} // leave_region
```

5. Give at least one example of distributed applications

   a) That should use **streams** rather than **datagrams**. (5 points)

   *Applications requesting transfers of large amounts of data: http, ftp, ...________*

   b) That should use **datagrams** rather than **streams**. (5 points)

   *Applications requesting transfers of small amounts of data ____________________*

6. Two concurrent processes access the same shared variable **count**.

   ```
   process one {
       count++;  
   }
   process two {
       count--;  
   }
   ```

   Assuming that **count** was initially equal to 3, what values can it take after the two processes have completed? (10 points minus 5 points for each incorrect or missing answer)

   **Answers:** 2, 3 and 4