COSC 4330 SECOND MIDTERM JULY 1, 2004

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

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

a) The Round-Robin scheduling algorithm does not differentiate between CPU-bound and I/O-bound processes.

TRUE, all processes have the same priority.

b) All UNIX message passing primitives use ***direct naming.***

FALSE, they use ports, which allows a process to wait for messages from different senders.

c) Peterson's algorithm assumes the existence of ***shared variables***.

TRUE, it would not work without shared variables.

d) Reliable datagrams use ***negative acknowledgements***.

FALSE, they use positive acknowledgements.

e) Most schedulers adjust the priorities of ***real-time processes*** in order to give each process its ***fair share*** of the CPU.

FALSE, real-time processes normally have fixed priorities.

2. Consider the following System V Release 4 scheduler: (3×5 points)

```
#ts_quantum  ts_tqexp  ts_slpret  ts_maxwait  ts_lwait  LEVEL
 1000        0         1          16000       1         # 0
  500        1         2          8000        2         # 1
  200        1         3          4000        3         # 2
  100        2         3          2000        3         # 3
```

a) Which events can ***increase*** the priority of a process at level 2?

(a) The process returns from the waiting state _____

(b) It stays for more than 4000ms in the ready queue _____

b) Which events can ***lower*** it?

It returns to the ready queue after having exhausted his time slice._____

_____

3. What is the major disadvantage of the ***at-most-once*** semantics for remote procedure calls? (5 points)

It does not protect against the risk of partial execution. _____

4. Consider the following solution to the mutual exclusion problem:

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

enter_critical_section(int pid) { //pid must be 0 or 1
     must_wait[1 – pid] = 1;
     while (must_wait[pid] == 1);  // wait
} // enter_critical_section

leave_critical_section(int pid) { //pid must be 0 or 1
     must_wait[1 – pid] = 0;
// leave critical section:
in_use = 0;
```

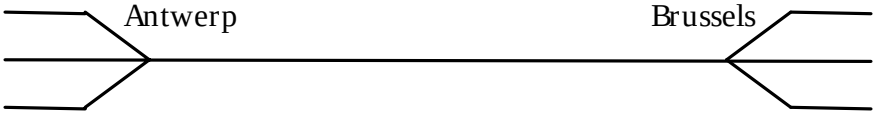
a) What is the problem with this solution? (5 points)

It can cause deadlocks. _____

b) When does this problem manifest itself? (5 points)

When the two processes enter the critical section in lockstep. _____

5. Two cities, say, Antwerp and Brussels, are linked by a single monorail track**:**



You are to synchronize train departures in a way that ensures that no trains moving in opposite directions will be simultaneously present on the track while allowing several trains moving in the same direction to share the track. The complete solution should involve four functions, namely, **leave_A( )**, **reach_B( )**, **leave_B( )** and **reach_A( )** but you will be only asked to write the two first ones since the two other ones are symmetrical. Do not worry about potential starvation conditions. (30 points)

```
semaphore can_enter = 1;
semaphore mutexa = 1;
semaphore mutexb = 1;
shared int A2Bcount = 0;
shared int B2Acount = 0;
```

```
(void) leave_A() {
P(&mutexa);
A2Bcount++;_____;
if (A2Bcount == 1_____)
     P(&can_enter); _____;
V(&mutexa);
} // leave_A
```

```
(void) reach_B() {
P(&mutexa);
A2Bcount--;_____;
if (A2Bcount == 0 _____)
     V(&can_enter); _____;
V(&mutexa);
} // reach_B
```

6. Explain why some applications are better implemented with ***streams*** and others with ***datagrams***. Give at least one example of both. (2×5 points)

Any application that sends data that cannot fit in a single message should use streams because streams guarantee that the messages will always arrive in order, without lost or duplicate messages, which is most applications need.

Applications that can fit the data they send in a single message should use datagrams to avoid the connection establishment overhead.

So a web server should use streams while a user authentication service could sue datagrams.