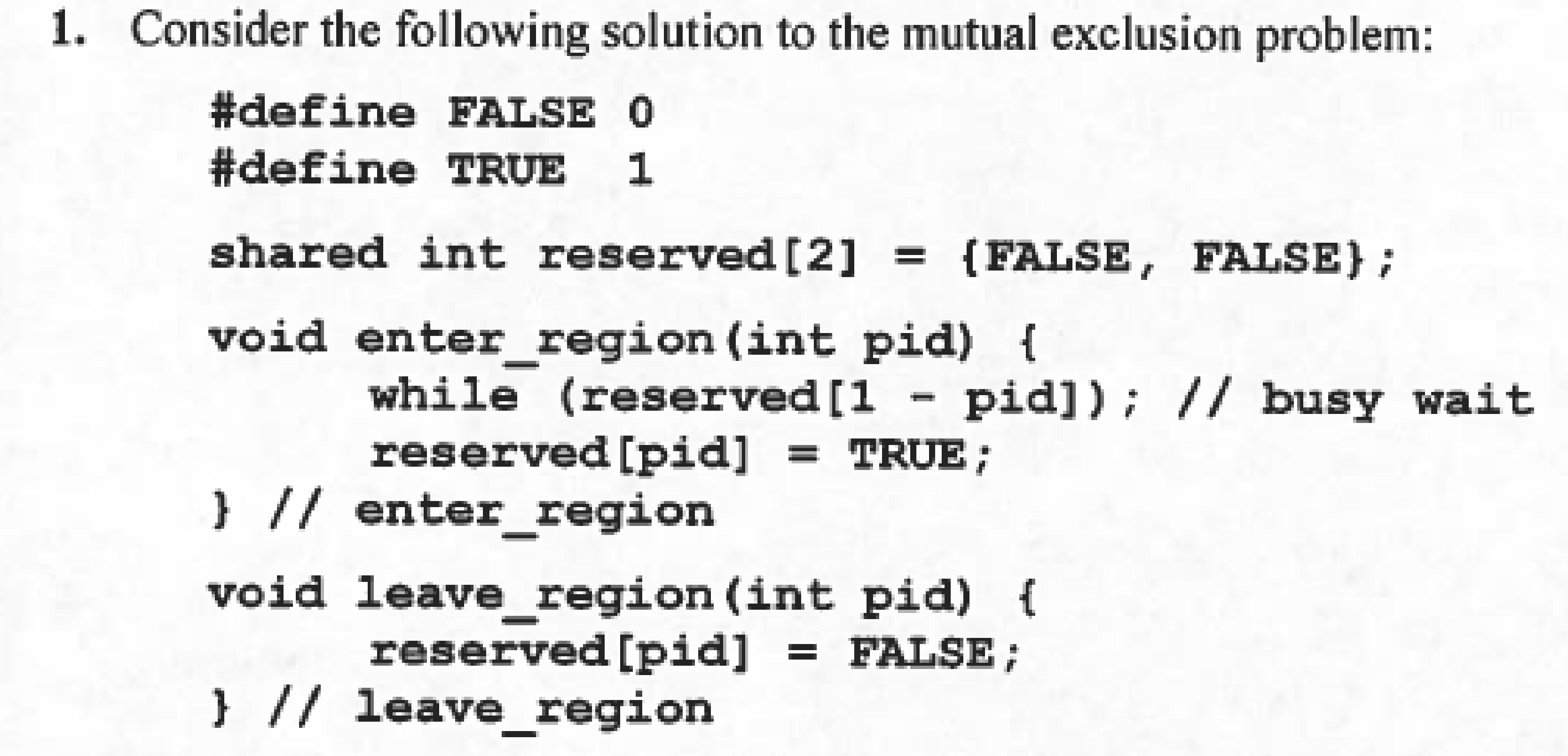
Solutions to the second midterm

COSC 4330/6310

Summer 2012

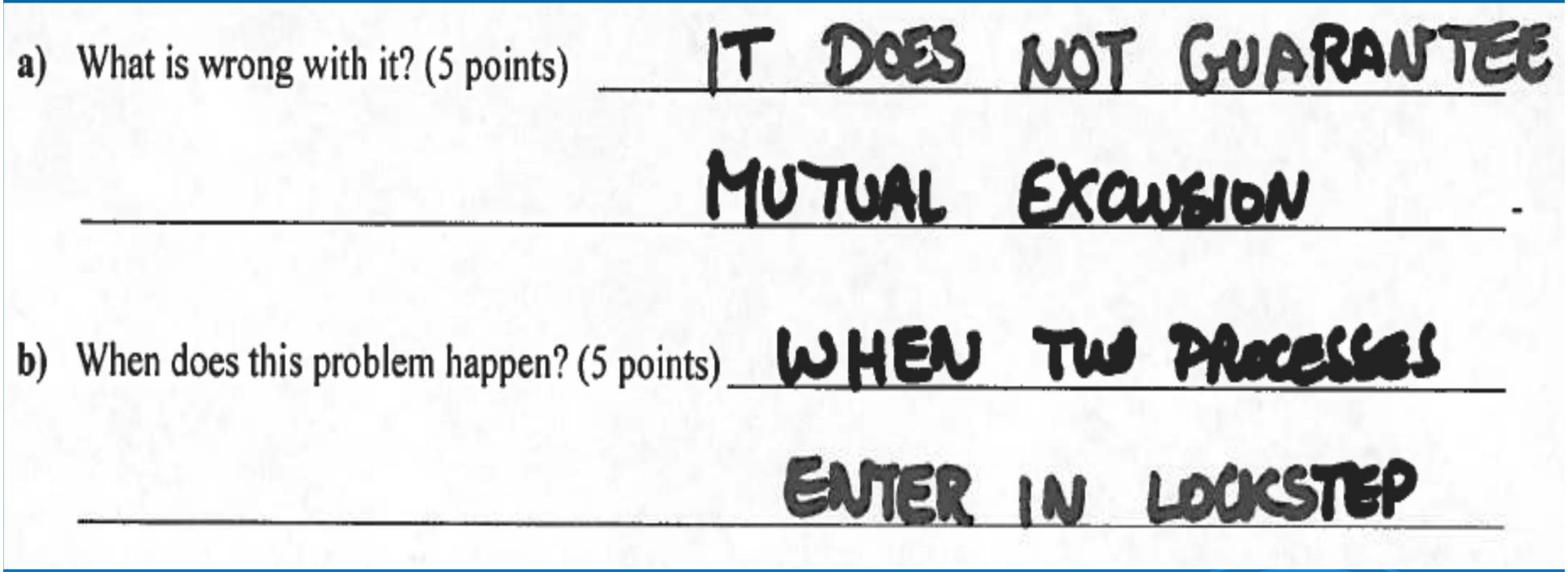
First question



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

```
#define FALSE 0
#define TRUE  1

shared int reserved[2] = {FALSE, FALSE};

void enter_region(int pid) {
        while (reserved[1 - pid]); // busy wait
        reserved[pid] = TRUE;
} // enter_region

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

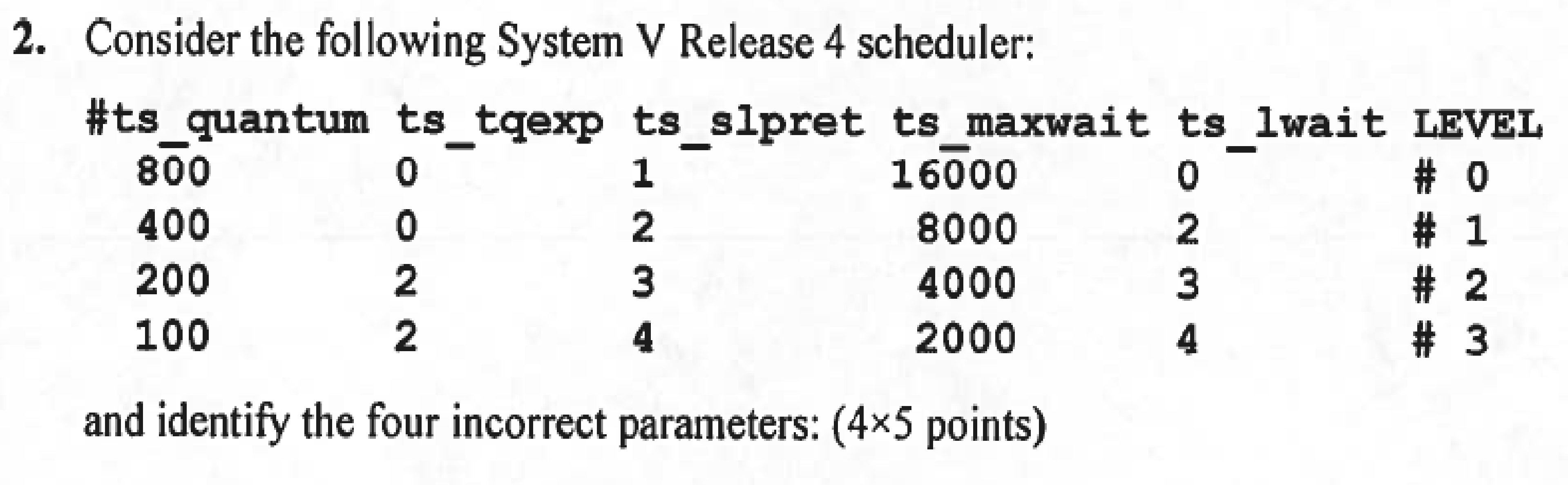
Answer



a) What is wrong with it? (5 points) IT DOES NOT GUARANTEE MUTUAL EXCLUSION.

b) When does this problem happen? (5 points) WHEN TWO PROCESSES ENTER IN LOCKSTEP

Second question

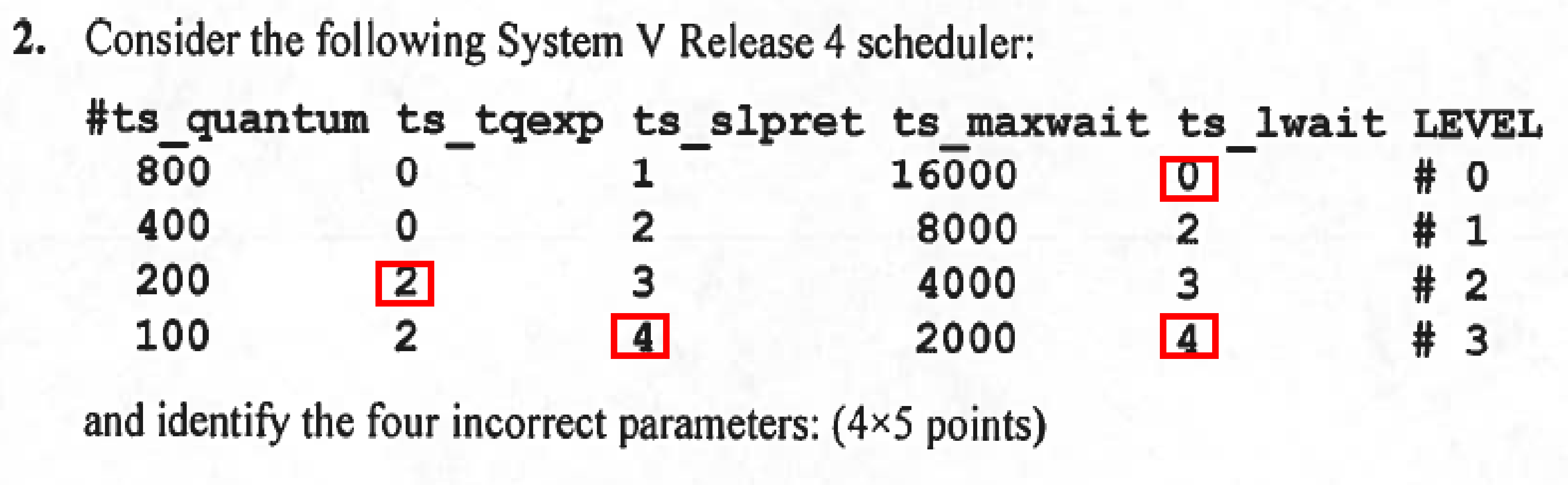


2. Consider the following System V Release 4 scheduler:

```
#ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait LEVEL
  800         0         1        16000       0        # 0
  400         0         2         8000       2        # 1
  200         2         3         4000       3        # 2
  100         2         4         2000       4        # 3
```

and identify the four incorrect parameters: (4×5 points)

Answer

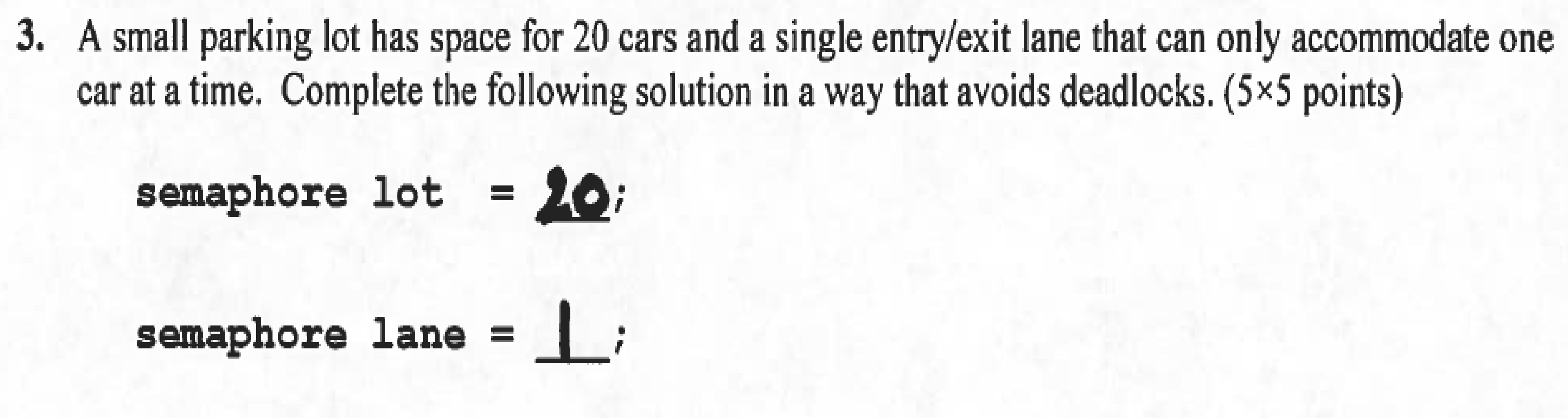


2. Consider the following System V Release 4 scheduler:

```
#ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait LEVEL
   800          0        1        16000       0       # 0
   400          0        2         8000       2       # 1
   200          2        3         4000       3       # 2
   100          2        4         2000       4       # 3
```

and identify the four incorrect parameters: (4×5 points)

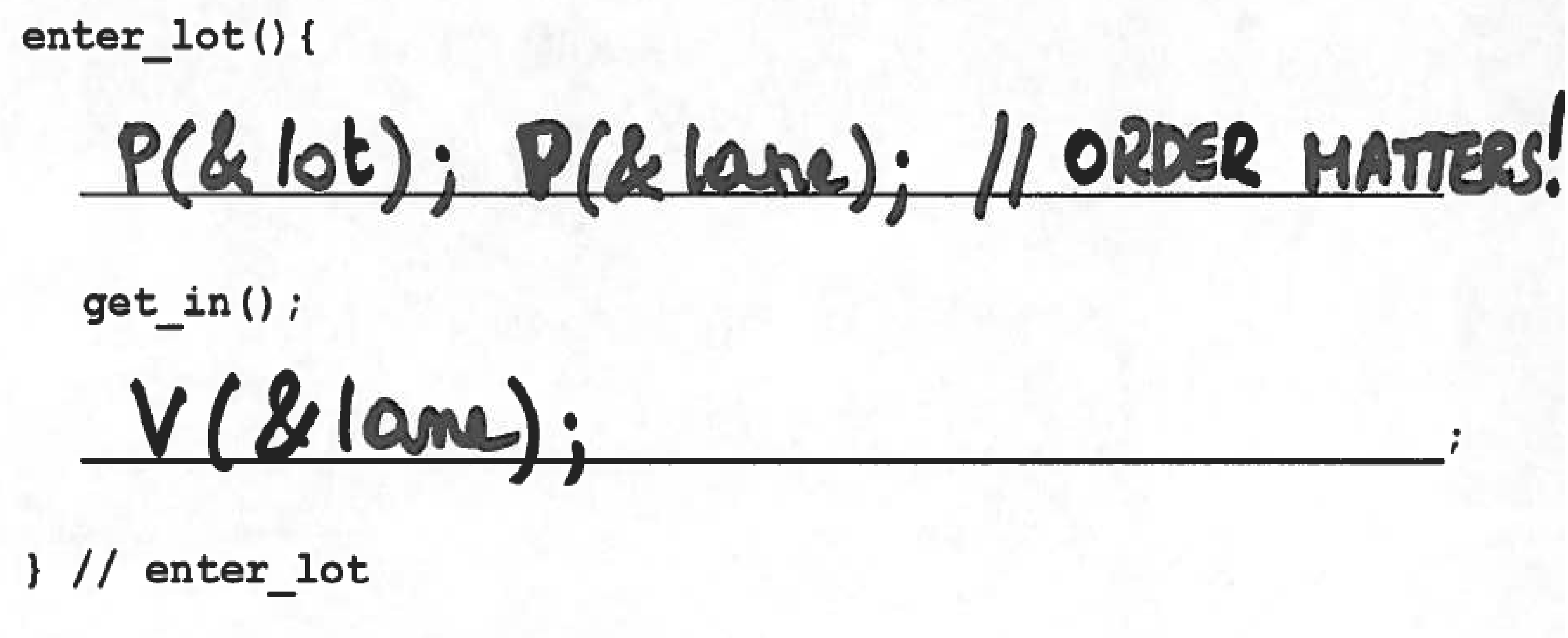
Third question



3. A small parking lot has space for 20 cars and a single entry/exit lane that can only accommodate one car at a time. Complete the following solution in a way that avoids deadlocks. (5×5 points)

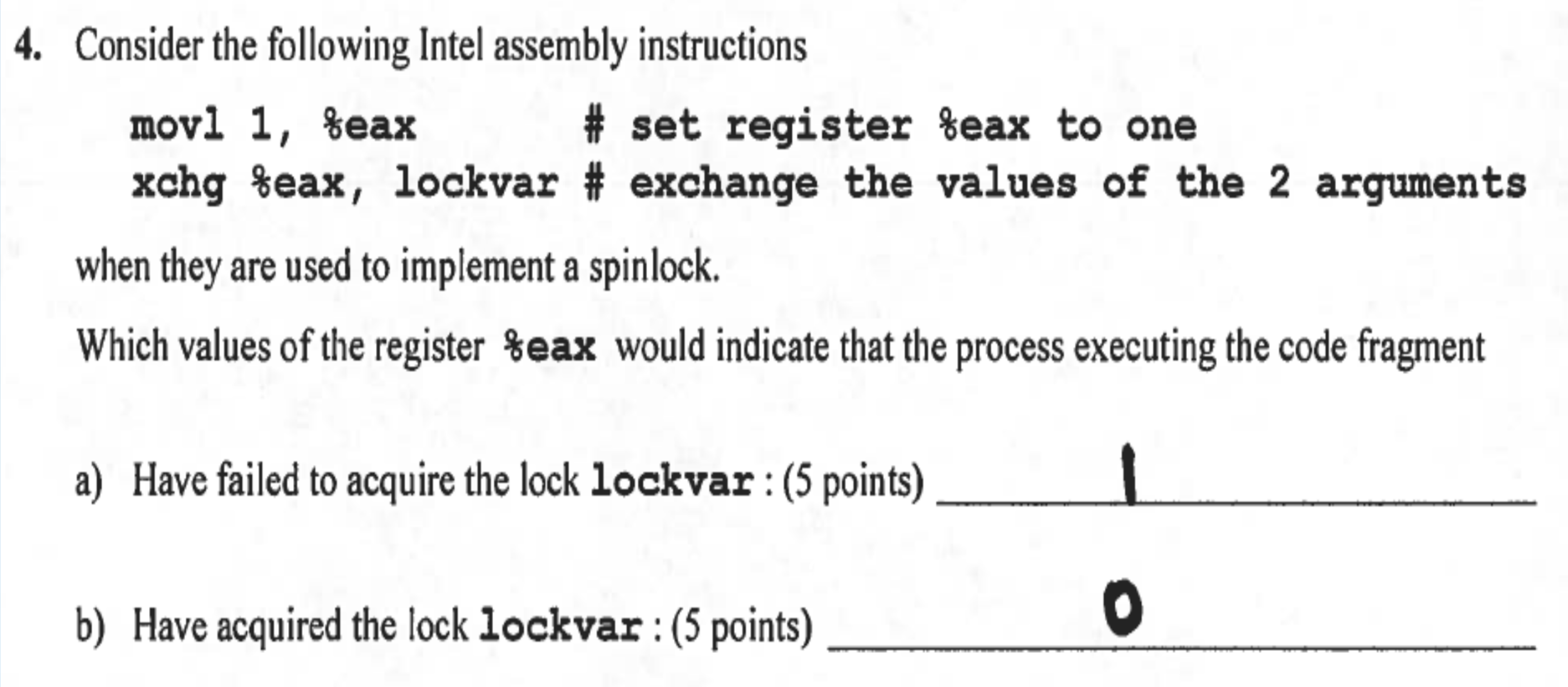
```
semaphore lot  = 20;

semaphore lane = 1;
```

Rest of answer



```
enter_lot(){
    P(&lot); P(&lane); // ORDER MATTERS!
    get_in();
    V(&lane);                                        ;
} // enter_lot
```

Fourth Question



4. Consider the following Intel assembly instructions

```
movl 1, %eax          # set register %eax to one
xchg %eax, lockvar # exchange the values of the 2 arguments
```

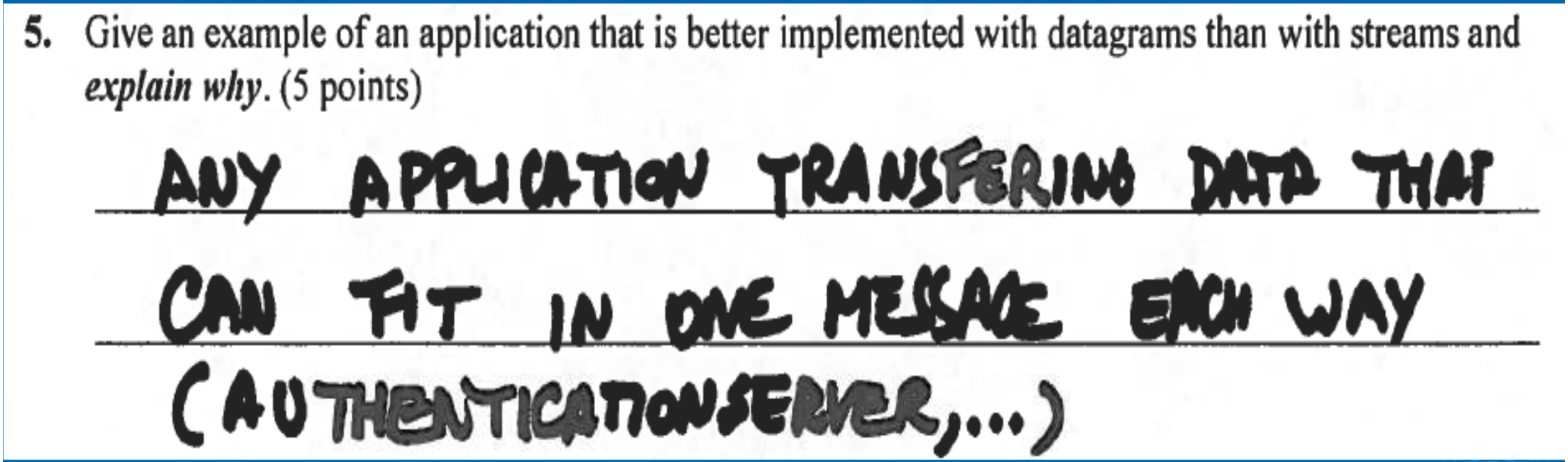
when they are used to implement a spinlock.

Which values of the register **%eax** would indicate that the process executing the code fragment

a) Have failed to acquire the lock **lockvar** : (5 points) 1

b) Have acquired the lock **lockvar** : (5 points) 0

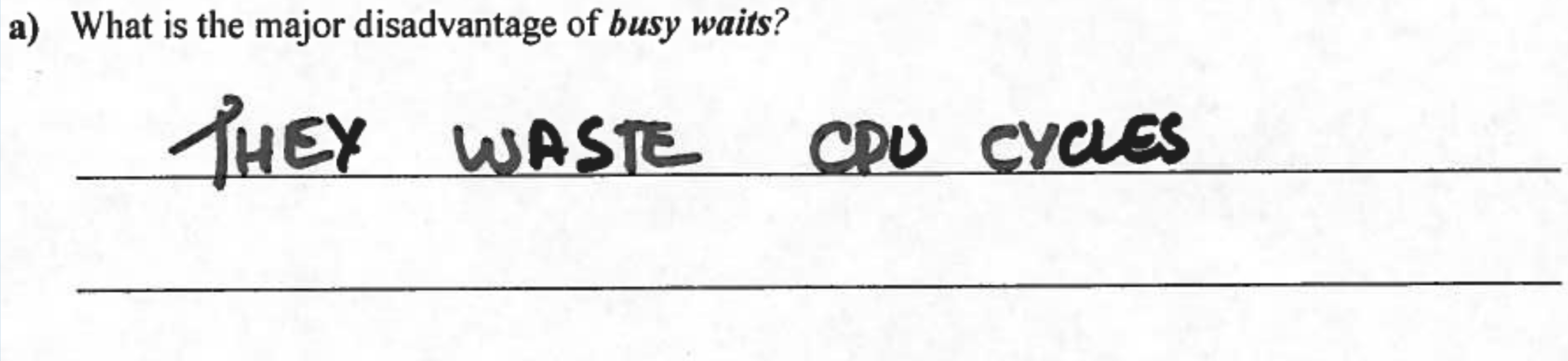
Fifth question



5. Give an example of an application that is better implemented with datagrams than with streams and *explain why*. (5 points)

ANY APPLICATION TRANSFERING DATA THAT CAN FIT IN ONE MESSAGE EACH WAY (AUTHENTICATIONSERVER,...)

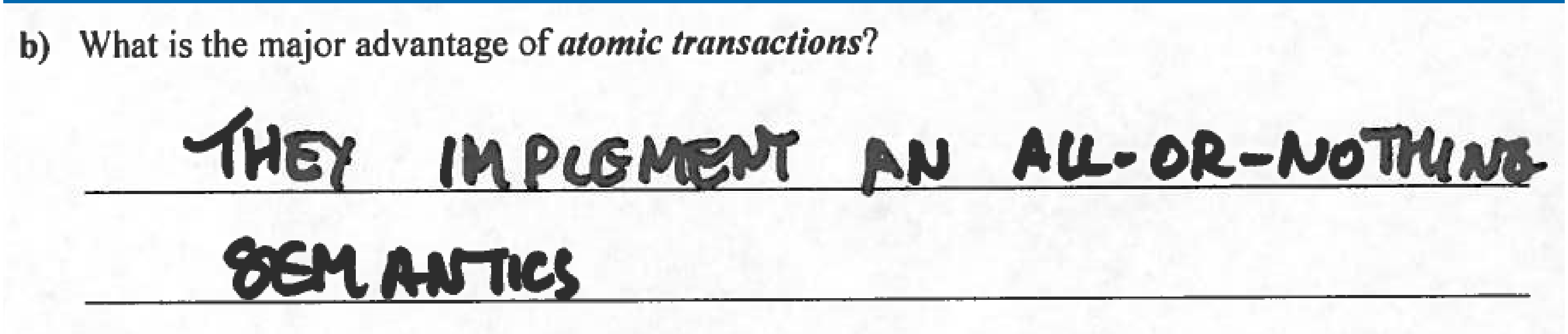
Sixth Question



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

THEY WASTE CPU CYCLES

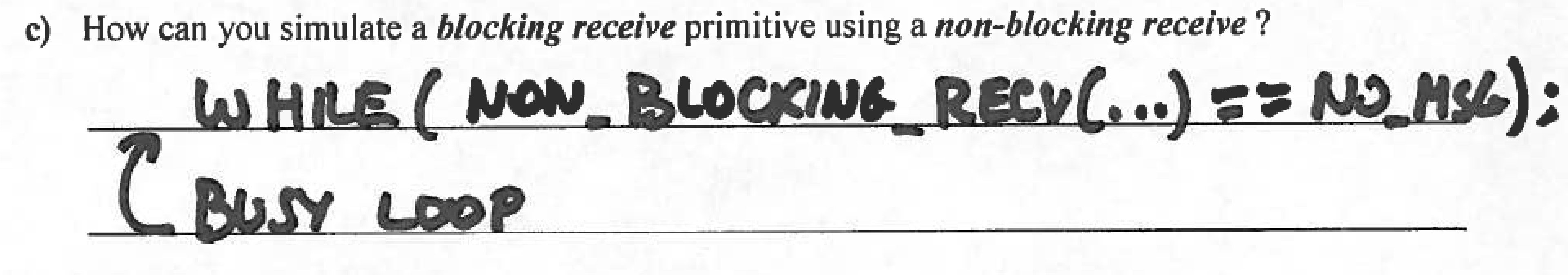
Sixth Question



b) What is the major advantage of *atomic transactions*?

THEY IMPLEMENT AN ALL-OR-NOTHING SEMANTICS

Sixth Question

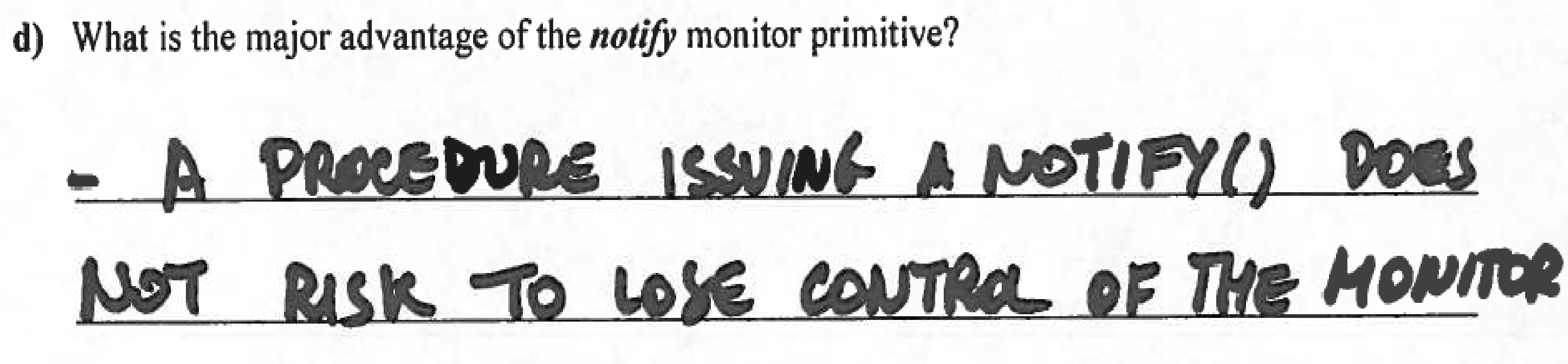


c) How can you simulate a ***blocking receive*** primitive using a ***non-blocking receive*** ?

WHILE (NON_BLOCKING_RECV(...) == NO_MSG);

↑ BUSY LOOP

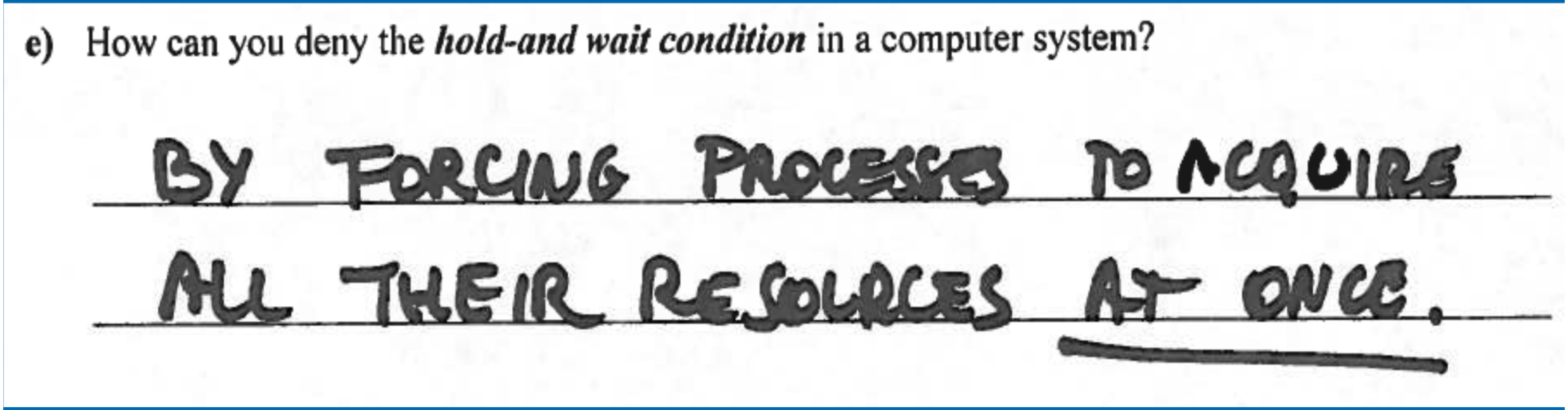
Sixth Question



d) What is the major advantage of the *notify* monitor primitive?

- A PROCEDURE ISSUING A NOTIFY() DOES NOT RISK TO LOSE CONTROL OF THE MONITOR

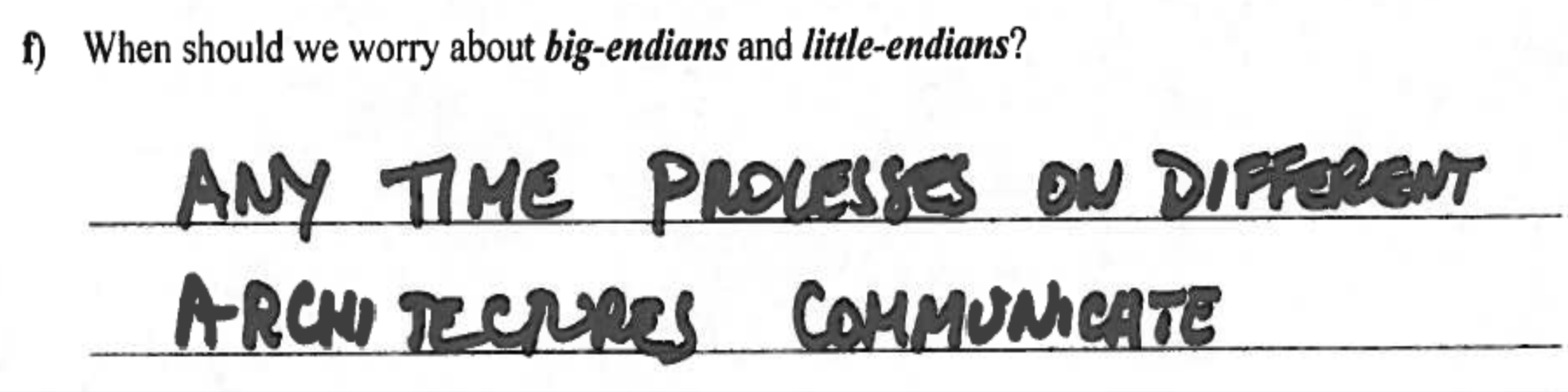
Sixth Question



e) How can you deny the ***hold-and wait condition*** in a computer system?

BY FORCING PROCESSES TO ACQUIRE ALL THEIR RESOURCES AT ONCE.

Sixth Question



f) When should we worry about ***big-endians*** and ***little-endians***?

ANY TIME PROCESSES ON DIFFERENT

ARCHITECTURES COMMUNICATE