CLOSED BOOK. YOU CAN HAVE ONE SHEET OF NOTES.

1. True or False (2×10 points)

T ___ F X _ Monitor conditions are normally initialized to zero. (They have no values.)

T X _ F ___ Mutex semaphores are normally initialized to one.

2. What is wrong with the following solution to the mutual exclusion problem? (20 points)

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

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

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

Mark the correct answer(s):
- It does not guarantee mutual exclusion under all circumstances.
- It may occasion deadlocks.
- It will sometimes prevent a process to enter the critical section when the other process is not inside.

3. A small parking lot has space for 35 cars and a single entry/exit point that can only accommodate one car at a time. Complete the following solution in a way that avoids deadlocks. (40 points)

```c
semaphore spaces = _____35___;
semaphore green = _____1____;

enter_lot()
{
    P(&spaces); P(&green);
    go_through_exit();
    V(&green); V(&spaces);
} // enter_lot
```

leave_lot()
{
    P(&green);
    go_through_exit();
    V(&green);
} // leave lot

4. Why do most good programmers avoid putting signal calls inside their critical sections? (20 points)

Most good programmers avoid putting signal calls inside their critical sections because any monitor procedure signalling a given condition will lose control of the monitor if there is another monitor procedure waiting on that condition.