COSC 4330 SECOND MIDTERM APRIL 2, 2001

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

1. Consider the following system V scheduler:

```
#ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait LEVEL

    800        0        1        4000        1   #  0
    400        0        2        2000        2   #  1
    200        X        M        1000        T   #  2
    100        Y        N         500        U   #  3
```

Give correct values for the four following parameters? (4×5 points)

X = _1__ M = _3__ T = _3__ U = _3__

2. Explain why some applications are better implemented with *virtual circuits* and others with *datagrams*. Give at least one example of both. (10 points)

Virtual circuits are better for transferring relatively large amounts of data that cannot fit in a single packet for it guarantees that the data will arrive as they were sent. Hence they are the best choice for FTP and HTTP protocols. Datagrams are better for sending and receiving small amounts of data that can fit in a single packet.

3. What is the major disadvantage of *busy waits*? (5 points) How can we to avoid them? (5 points)

Busy waits waste CPU cycles and cause too many context switches. The best way to avoid them is to use kernel-supported synchronization primitives are they put waiting processes in the waiting state.

4. Two concurrent processes access the same shared variable **`count`**.

```
process one {
   count += 3;
}
```

```
process two {
   count -= 5;
}
```

Assuming that **`count`** was initially equal to 20, what values can it take after the two processes have completed? (10 points)

Answers: _15, 23 and 18, the sole correct answer_____

5. Fill in the blanks to obtain a monitor implementation of a *binary semaphore* (4×5 points)

```
Class binary_semaphore {
     private condition go;
     private int value;

     public void synchronized V() {
          value = 1___;
          go.signal___;
     } // V

     public void synchronized P() {
          if (_value == 0_)
               go.wait__;
          value = 0;
     } // P

     semaphore(int initial_value) {
          value = initial_value;  // must be 0 or 1
     } //constructor

} // Class binary_semaphore
```

6. Fill in the blanks to obtain a solution to the mutual exclusion problem for two processes whose ID's can be 0 or 1 (4×points)

```
int no_wait;
int status[2] = {0, 0};

void enter_region(int pid) {
     status[pid] = _1_________;
     no_wait = _1 - pid____;
          while (status[_1 - pid_] && no_wait != pid);
} // enter_region

void leave_region(int pid) {
     status[pid] = _0_;
} // leave_region
```

7. How can you explain the fact that the throughput of a computer using a round-robin scheduling policy often goes down when the number of users goes up? (10 points)

When the number of users go up, most round-robin scheduling policy will reduce the duration of time slices in order to maintain the same customer waiting time. Unfortunately, this increases the context switch overhead.