1. (10%) What are the three contexts in which concurrency arises?

2. (10%) List three degrees of awareness between processes and briefly define each.

3. (10%) List the three control problems associated with competing processes and briefly define each.

4. (10%) What operations can be performed on a semaphore? Briefly explain each.

5. (20%) Consider the following program:

   ```java
   boolean blocked [2];
   int turn;
   void P (int id)
   {
     while (true) {
       blocked[id] = true;
       while (turn != id) {
         while (blocked[1-id])
           /* do nothing */;
         turn = id;
       } /* critical section */;
       blocked[id]=false;
       /* remainder */;
     }
   }
   void main()
   {
     blocked[0] = false;
     blocked[1] = false;
     turn = 0;
     parbegin (P(0), P(1));
   }
   ```

   This is a software solution to the mutual exclusion problem for two processes. Find a counterexample that demonstrates that this solution is incorrect.
6. (25%) Consider the following definition of semaphores:

```c
void semWait (s) {
    if (s.count > 0) {
        s.count--;
    }
    else {
        place this process in s.queue;
        block;
    }
}
void semSignal (s) {
    if (there is at least one process blocked on semaphore s) {
        remove a process P from s.queue;
        Place process P on ready list;
    }
    else
        s.count++;
}
```

(a) Compare this set of definitions with the one we introduced in class. Is there any difference in the effect of the two set of definitions when used in programs? That is, could you substitute one set for the other without altering the meaning of the program?

(b) Write definitions for semWaitB and semSignalB for a binary semaphore, by changing the definition given above as appropriate.

7. (15%) What output can be produced by the following processes P0, P1 and P2 that are started concurrently. We use labels A, B, C, D to identify statements in the programs. For simplicity, we assume that each individual statement A, B, C, D is atomic.

```plaintext
boolean cond1 = false, cond2 = false; //initial values for cond1 and cond2

Process P0 | Process P1
------------- | -------------
[A] cond1 = true; | [B] cond2 = cond1;

Process P2
-----------
[C] if (cond1==true) print("Toronto ") else print("Montreal ");
[D] if (cond2==true) print("Hamilton ") else print("London ");
```

**Complete the following table.** One entry is already filled in. Show all the other possible outputs and for each output one execution sequence.

<table>
<thead>
<tr>
<th>OUTPUT PRODUCED</th>
<th>ONE EXECUTION SEQUENCE LEADING TO THIS OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toronto Hamilton</td>
<td>A B C D</td>
</tr>
</tbody>
</table>