EXAM ON PROCESSES AND SCHEDULING – SPRING 2007
CSC 262 – OPERATING SYSTEMS
NICHOLAS R. HOWE

1. **Vocabulary** (18 points). Give a one or two sentence definition/description of the following terms.

a). MIC chip

b). O(1) Scheduler

c). Atomic operation

d). Peterson Algorithm

e). TSL operation

f). Two Generals Problem

g). Pipe

h). Multicast

i). Dutch Banker

2. **Process Scheduling** (24 points). Consider the process table below.

Process ID	0	1	2	3	4	5	6
Queue	0	0	1	2	1	2	2
Priority	0	1	3	1	4	3	5

Draw a Gantt chart showing which processes would execute when under the following scheduling policies over the next 1000 ms. You may ignore time spent on context switching overhead, and assume that each process runs for its full quantum without blocking or interrupts. However, a process that is allowed to run for 300 ms in total will terminate and be removed from the process table, perhaps allowing other processes to go.

a. Scheduling with multiple queues. Use round-robin scheduling between queues (all assumed to be equal priority; start in numerical order from 0) and strict priority scheduling within a queue (high numbers are high priority). Quantum is 100 ms.

b. Scheduling with multiple queues. Queue 0 is lowest priority, queue 2 is highest. Use strict priority scheduling between queues, and simple round-robin scheduling within a queue. (Assume that the process with the highest "priority" has been waiting longest for a turn.) Quantum is 150 ms.

3. **Threads** (10 points). Which of the following will be shared by two related lightweight processes?

Code Segment -- Call Stack -- Data Segment (Heap) -- Status Registers -- Resources

4. **Deadlock** (24 points). The municipality of Deadlock City has a number of railroads passing through the town. Each railroad crosses the others at multiple points, and the trains passing though are long enough that they sometimes extend past several crossings at a time.

a. If multiple trains enter the city at once, it has been found that sometimes a situation arises where no train can continue to move forward. The only solution is to force one or more trains to reverse slowly out of the city to let the others proceed. Draw a diagram showing one possible configuration of trains matching the description above.

b. It has been proposed to require all trains to "reserve" all the crossing points they wish to use before their entry into the city. Once a train has reserved a crossing point, no other train can reserve it until the original train has passed through. Trains unable to reserve all the crossings they need will have to wait outside the city until their crossings are free. Will this solve the problem? Comment on any disadvantages of this approach.

c. An alternate solution is proposed. Each train would be given a unique priority number. Trains would declare the crossings they intend to use on entry to the city as described above. However, instead of being required to wait, trains can enter the city and proceed until they come to a crossing that has been declared on the itinerary of a train with higher priority, at which point they will have to wait until the higher-priority train passes through. Will this solve the problem in (a)? Comment on the advantages/disadvantages relative to (b).

d. List Coffman's necessary conditions for deadlock, and identify any connections the the scenarios described above.

[Note: this problem was inspired by the city of Pine Bluff, Arkansas.]

5. **Classic Problems** (24 points). Consider the following proposed solution to the east-west bridge problem.

```
semaphore mutex(1), queue(0);
int crossing(0), waiting(0);
enum {east,west} dir = east;

void entryEast() {
  down(mutex);
  if (crossing > 0)
      && (dir == west) {
    waiting++;
    up(mutex);
    down(queue);
    waiting--;
  }
  dir = east;
  crossing++;
  up(mutex);
}

void exitEast() {
  down(mutex);
  crossing--;
  if (crossing == 0)
      && (waiting > 0) {
    up(queue);
  } else {
    up(mutex);
  }
}
```

```
void entryWest() {
  down(mutex);
  if (crossing > 0)
      && (dir == east) {
    waiting++;
    up(mutex);
    down(queue);
    waiting--;
  }
  dir = west;
  crossing++;
  up(mutex);
}

void exitWest() {
  down(mutex);
  crossing--;
  if (crossing == 0)
      && (waiting > 0) {
    up(queue);
  } else {
    up(mutex);
  }
}
```

a). Does this protocol ensure mutual exclusion between eastbound and westbound processes? Explain why/why not.

b). Does the protocol prevent starvation in all cases? If yes, explain why. If not, explain how to fix it so that starvation cannot occur.

c). Contrast the protocol above with the protocol we developed on the homework for the north-south bridge, in terms of its efficiency.

d). Contrast the protocol above with the much simpler protocol below, again in terms of its efficiency.

```
semaphore bridge(1);

void entryEast() {
  down(bridge);
}

void exitEast() {
  up(bridge);
}
```

```
void entryWest() {
  down(bridge);
}

void exitWest() {
  up(bridge);
}
```