

# CS 423UG Fall 2005: Midterm Exam Solutions

## 1 The RedGene Puzzle

1. (a) (per-quantum) P1,P2,P3,P4,P5,P1,P3,P5,P1,P5,P1,P5,P1,P1,P1,P1,P1. (b) P2,P4,P3,P5,P1.  
(c) P2,P5,P1,P3,P4.
2. RR:  $(9+1+5+3+9)/5=5.4$  tu's. SJF:  $(9+0+2+1+4)/5=3.2$  tu's. Prio:  $(6+0+16+18+1)/5=8.2$  tu's. SJF has the shortest average waiting time (3.2 tu's).
3. Advantage: higher priority processes can be given more time quanta than lower priority processes. Disadvantage: higher priority processes can hog the CPU (by having too many pointer entries in the ready queue).

## 2 The MicroDead Puzzle

1. The Resource Allocation graph has one node for each process and for each resource in the system. It has an edge from a resource  $R_i$  to a process  $P_j$  if  $P_j$  currently holds  $R_i$  (8 such edges), and an edge from a process  $P_j$  to a resource  $R_i$  if  $P_j$  is waiting for resource  $R_i$  (8 such edges). The Wait-For graph has only processes as its nodes. It has an edge from process  $P_i$  to process  $P_j$  if  $P_j$  is currently holding some resource for which  $P_i$  is waiting (8 such edges).
2. There are 2 cycles in the Wait-For graph: P3-P4-P5 and P1-P2-P5-P3. Thus there are 5 deadlocked processes: P1,P2,P3,P4,P5. No points deducted if P6 also included in this list (although P6 not deadlocked, it is still waiting for P1).
3. The minimum-sized set has exactly one process - either P3 or P5. Killing P3 (only) will release both deadlock cycles. Alternately, killing P5 (only) will release both deadlock cycles.

## 3 The Mutexoogole Puzzle

1. The code has three problems: it can deadlock, it fails to restore s1, and it has unmatched downs and ups.

```

2. void atomic_swap(Stack *s1, Stack *s2) {
    Item *item1;
    Item *item2; // items being transferred
    Stack *tmp;
    if(s1->id > s2->id) {
        // impose ordering on down operations
        tmp = s1;
        s1 = s2;
        s2 = tmp;
    }
    down(s1->mutex);
    down(s2->mutex);
    item1 = pop(s1);
    if(item1 != NULL) {
        item2 = pop(s2);
        if(item2 != NULL) {
            push(s2, item1);
            push(s1, item2);
        }
        else
            push (s1, item1);
    }
    up(s2->mutex);
    up(s1->mutex);
}

```