

```

void enQ(int n) {
    x = new Item(n);
    if (head==NULL) head=x;
    else tail->next=x;
    tail=x;
}

```

```

int deQ() {
    x=head;
    yield();
    if (x==NULL) return -1;
    if (x==tail) tail=NULL;
    head=x->next;
    v=x->value;
    delete x;
    return v;
}

```

1. Two threads are started, with the list containing 3 items, values 1 (in head->value), 2, and 3 (in tail->value). Each thread executes deQ() operations until the list is empty.
 - a. [20 pts] Present a scenario (interleaving) which causes some erroneous results to occur. Explain what the erroneous actions or conditions are.
 - b. [20 pts] Complete the definition of the List class, showing all variable declarations, and adding P() and V() operations to the methods which ensure that each deQ() and enQ() operation functions atomically. Place the semaphore operations so that as few instructions as possible are executed in the critical regions. The “new” and “delete” statements are guaranteed to run as atomic operations. However, the scheduler uses pre-emptive scheduling, so that a thread switch can occur between any two instructions. Explain why your solution works, and why even more instructions cannot be removed from the critical regions.