3) (10 pts) ALG (Stacks and Queues)

Consider the process of merging two queues, `q1` and `q2`, into one queue. One way to manage this process fairly is to take the first item in `q1`, then the first item from `q2`, and continue alternating from the two queues until one of the queues run out, followed by taking all of the items from the queue that has yet to run out in the original order. For example, if `q1` contains 3, 8, 2, 7 and 5, and `q2` contains 6, 11, 9, 1, 4 and 10, then merging the two queues would create a queue with the following items in this order: 3, 6, 8, 11, 2, 9, 7, 1, 5, 4, and 10. Assume that the following struct definitions and functions with the signatures shown below already exist.

```
typedef struct node {
    int data;
    struct node* next;
} node;

typedef struct queue {
    node* front;
    node* back;
} queue;


// Initializes the queue pointed to by myQ to be an empty queue.
void initialize(queue* myQ);

// Enqueues the node pointed to by item into the queue pointed
// to by myQ.
void enqueue(queue* myQ, node* item);

// Removes and returns the front node stored in the queue
// pointed to by myQ. Returns NULL if myQ is empty.
node* dequeue(queue* myQ);

// Returns the number of items in the queue pointed to by myQ.
int size(queue* myQ);
```

On the following page, write a function that takes in two queues, `q1` and `q2`, and merges these into a single queue, emptying out `q1` and `q2` in the process and returning a pointer to the resulting queue.

```
queue* merge(queue* q1, queue* q2) {

    queue* res = malloc(sizeof(queue));
    initialize(res);

    int list = 0;

    while (size(q1) > 0 || size(q2) > 0) {

        if (list == 0 && size(q1) > 0)
            enqueue(res, dequeue(q1));
        else if (list == 1 && size(q2) > 0)
            enqueue(res, dequeue(q2));

        list = (list+1)%2;
    }

    return res;
}
```

Grading: 2 pts for properly creating an empty list to return (1 pt malloc, 1 pt init)
2 pts for toggling mechanism between lists
2 pts for handling unequal list sizes somehow
1 pt dequeing
2 pts enqueing item into their created queue
1 pt return