2) (10 pts) ALG (Linked Lists)

Suppose we have a queue implemented as a doubly linked list using the structures shown below with head pointing to node at the front of the queue and tail pointing to the node at the end of the queue.

```
struct node {
    int data;
    struct node *next, *prev;
}

struct queue {
     int size;
     struct node *head, *tail;
}
```

Write an enqueue function for this queue. If the queue is already full, return 0 and take no other action. If the queue has not been created yet, return 0 and take no other action. If the queue isn't full, enqueue the integer `item` into the queue, make the necessary adjustments, and return 1. Since there is no fixed size, the queue will be considered full if a new node can't be allocated.

```
int enqueue(queue *thisQ, int item) {
     struct node *newNode = malloc(sizeof(struct node)) ;    // 1 pt
     if(thisQ == NULL) return 0;
     if(newNode == NULL) return 0;

     newNode->data = item;                  // .5 pts

     newNode->next = NULL;                  // .5 pts

     thisQ->size = thisQ->size + 1;         // .5 pts

     if(thisQ->head == NULL) {
          newNode->prev = NULL;             // .5 pts
          thisQ->head = newNode;            // .5 pts
          thisQ->tail = newNode;            // .5 pts
          return 1;
     }

     newNode->prev = thisQ->tail;           // 2 pts

     thisQ->tail->next = newNode;           // 2 pts

     thisQ->tail = newNode;                 // 2 pts
     return 1;
}
```

Grading Note: Please count total points and round down to record an integer, so 8.5 gets recorded as 8, and 8.0 also gets recorded as 8.