2) (10 pts) DSN (Sorting)

For this problem, fill in the blank to finish the stable (elements with the same values are kept in their original order) quicksort on a linked list. We are using the head of the linked list as a pivot. You can assume that the following linked list functions have all been implemented and take O(1) operations.

Note: Each blank is worth one point and involves either making calls or filling in parameters to the functions whose prototypes and descriptions are given below.

```c
typedef struct Node {
    int value;
    struct Node * next;
} Node;

typedef struct List {
    Node* front;
    Node* back;
} List;

void addToTail(List * list, Node * node); // Add to tail

// Returns a list that is the combination of 2 given lists.
List * merge(List * front, List * back);

Node* getAndRemoveHead(List * list); // Removes and returns the head
List* createEmptyList(); // Returns dynamically allocated empty List
int isEmpty(List * list); // Returns 1 if empty and 0 otherwise

// Sort code on next page
```

```
List * sort(List * lst) {

    if (_____(lst)) return lst;

    Node * pivot = getAndRemoveHead(lst);

    List * first = _____();

    List * last = _____();

    List * middle = createEmptyList();

    addToTail(middle, pivot);

    while (!isEmpty(lst)) {

        Node * cur = _____(lst);

        if (cur->value < pivot->value)

            addToTail(_____, cur);
        else if (cur->value == pivot->value)

            addToTail(_____, cur);
        else

            addToTail(_____, cur);
    }

    first = sort(_____);

    last = sort(_____);

    first = merge(first, middle);
    first = merge(first, last);
    free(middle);
    free(last);
    free(lst);

    return _____;

}
```