1) (5 pts) ALG (Dynamic Memory Management in C)

The struct below is used to define a node in a linked list. Below it is a function, ***freeList***, that is given a pointer to the front/head of a linked list. The function is supposed to free all the dynamically allocated memory associated with the linked list that the given pointer is pointing to. Unfortunately, the function does not work. Explain why the function doesn't work and propose a fix to the function, **in words only, conceptually explaining the order in which the key steps have to be executed.**

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

void freeList(node* front) {
    if (front != NULL)
        free(front);
    if (front->next != NULL)
        freeList(front->next);
}
```

Problem(s) with current code

If front isn't NULL, the memory it's pointing to gets freed. When this occurs, the link to the memory for future nodes, if they exist, is lost and can no longer be freed.

Grading: 2 pts or 0 pts, student must clearly indicate that pointer to potential list memory is inaccessible for credit.

Proposed fixes (conceptually, in words)

If front is already NULL, no action should be taken. This must be checked and taken care of first. Next, the recursive call to free the rest of the list should occur, so long as there is a rest of the list. Finally, front should be freed, last.

Grading: 1 pt for each component, described in the appropriate order.