2) (5 pts) DSN (Linked Lists)

Given the linked list structure named `node`, defined in lines 1 through 4, and the function named `eFunction` defined in lines 6 through 14, answer the questions below.

```
 1 typedef struct node {
 2         int data;
 3         struct node * next;
 4 } node;
 5
 6 node* eFunction(node* aNode){
 7    if(aNode == NULL) return NULL;
 8    if(aNode->next == NULL) return aNode;
 9
10    node* rest = eFunction(aNode->next);
11    aNode->next->next = aNode;
12    aNode->next = NULL;
13    return rest;
14 }
```

(a) (1 pt) Is this function recursive? (Circle the correct answer below.)

YES (1 pt) NO

(b) (2 pts) What does the function eFunction do, in general to the list pointed to by its formal parameter, aNode?

This function reverses the list originally pointed to by aNode and returns a pointer to the new front of the list. (**Grading:** 1 pt for reverse, 1 pt for return pointer to reversed list.)

(c) (2 pts) What important task does line 12 perform?

The last node in a linked list must have its next pointer point to NULL. That is how most linked list functions detect the end of the list. Line 12 does this since aNode ends up point to the last node in the list. After the reversal is complete, it's necessary to make sure that the next pointer of the last node in the resulting list is pointing to NULL because before line 12 it's not. (It's pointing to the second node in the original list, which is the last node in the list pointed to by rest.)

Grading: Most of this detail is unnecessary. Full credit for noting that the line sets the last node's next pointer of the resulting list to NULL. Give partial as needed.