2) (10 pts) DSN (Linked Lists)

Write a **<u>recursive</u>** function that takes in the head of a linked list and frees all dynamically allocated memory associated with that list. You may assume that **<u>all</u>** the nodes in any linked list passed to your function (including the head node) have been dynamically allocated. It's possible that your function might receive an empty linked list (i.e., a NULL pointer), and you should handle that case appropriately.

Note that your function must be recursive in order to be eligible for credit.

The linked list node struct and the function signature are as follows:

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

void destroy_list(node *head) {
     if (head == NULL)
          return;

     destroy_list(head->next);
     free(head);
}
```

Grading: Award 3 pts for correctly handling the base case. Award 3 points for a correct recursive call. Award 4 points for freeing *head* <u>after</u> the recursive call.

Note: Some students might set head->next to NULL before freeing *head*., which causes some unnecessary computation, but doesn't render their solution incorrect. You can ignore that.

However, if they set *head* itself to NULL before freeing it, they should not be eligible to receive the 4 points for freeing *head* after the recursive call.