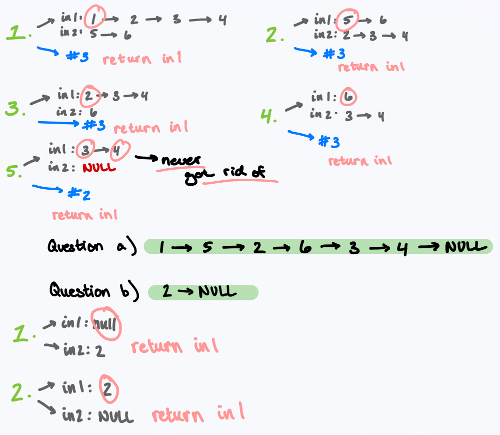
All of the points for this problem will be assigned based on your explanation, since we have full faith in your ability to run this program and copy down the answer.

```
struct Node {
    int val;
    Node*  next;
};

Node* llrec(Node* in1, Node* in2)
{
    if(in1 == nullptr) {
        return in2;
    }
    else if(in2 == nullptr) {
        return in1;
    }
    else {
        in1->next = llrec(in2, in1->next);
        return in1;
    }
}
```

CALL #
LIST(s)
RETURNED VALUE
IF STATEMENT #



Question a: What linked list is returned if llrec is called with the input linked lists in1 = 1,2,3,4 and in2 = 5,6?

Question b: What linked list is return if llrec is called with the input linked lists in1 = nullptr and in2 = 2?

To show work, you can draw a call tree or box diagram of the function calls using some simplified substitution of your choice rather than pointer values (e.g. "p3" for a pointer to a node with value 3). Submit your answers as a Markdown or PDF file showing your work and derivations supporting your final answer. You must name the file **q4_answers.[pdf,md]**. As before you must push this file into your github repo to receive credit.

When the recursive calls go back up the call list, it is changing the very last node each time (because the function has *in1->next* being set to the very next return statement which doesn't occur until the very end). So, our result is that, starting from the last element of the redefined *in1*, the nodes are changing as it goes back up the call list to whatever the correspond *in1* or *in2* object is (depending on which if statement the call is in). **Call work is shown above.**