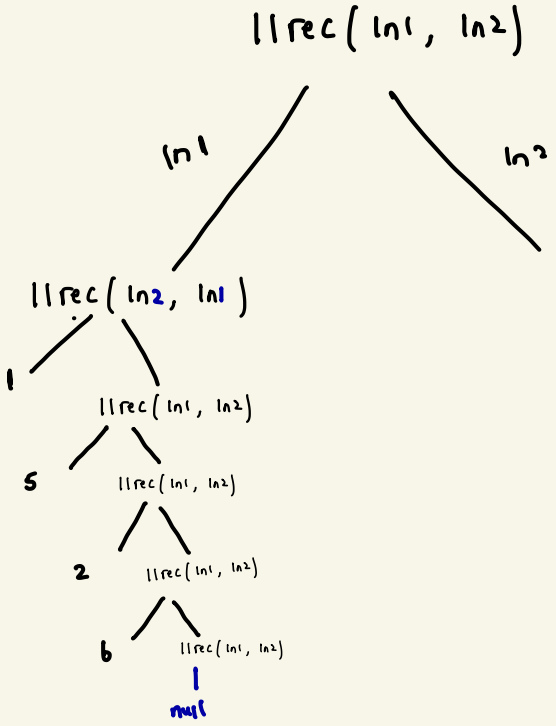
4)



1) start (1, 2, 3, 4 | 5, 6)

2) 1 — (5, 6 | 2, 3, 4)

3) 1 — 5 — (2, 3 4 | 6)

4) 1 - 5 - 2 (6 | 3, 4)

5) 1 - 5 - 2 - 6 (3, 4 | null)

6) return

displays: 1, 5, 2, 6, 3, 4

question a:
When the function is called
It first checks if in1 and in2 are null, which they aren't, so it moves to the else part of the conditional statement, where in->next is assigned to the function call, with the in1 and in2 swapped; at this point, and as shown in the diagram, in1->1

function called itself and keeps swapping the in1 and in2, as shown in the diagram I drew above, until (it hits the in2 base case) the last call is when in1->1->5->2->6->next(in2, in1),

at this point, in2=in1 with the values 3,4 and in2=nullprt

The next pointer then points to the 3,4, as the functions clear the stack from the recursion.

question b:
The linked list will return 2. because it will first check the first if function in1==nullprt, which returns in2=2 since they are no other values in the stack