Problem 4 — Linked List Recursion Tracing

CSCI 104 — Devin C. Martin

Question a:

```
Node* llrec(Node* in1, Node* in2)
```

in1: 1 → 2 → 3 → 4

in2: 5 → 6

Call 1

First Recursive call - llrec(1, 5):

1 → llrec(5, 2)

↓

Call 2

Second Recursive Call - llrec(5, 2):

5 → llrec(2, 3)

Call 3

Third Recursive Call - llrec(2, 3):

2 → llrec(3, 6)

Call 4

Fourth Recursive Call - llrec(3, 6):

3 → llrec(6, 4)

Call 5

Fifth Recursive Call — llrec(6, 4)

6 → 4

(this call will return node 4)

Final State

as we back trace through the calls, we update the arrows to reflect:

- after llrec(6,4), 6 points to 4
- 3 points to 6, forming: 3 → 6 → 4
- 2 points to 3, extending to: 2 → 3 → 6 → 4
- 5 points to 2, resulting in: 5 → 2 → 3 → 6 → 4
- finally 1 points to 5 creating the final list:

1 → 5 → 2 → 6 → 3 → 4

final list: 1 → 5 → 2 → 6 → 3 → 4

each arrow represents the 'next' pointer of a node being set to the head of the list returned by the recursive call, resulting in the final order

Question b below

Question b:

Node* llrec(Node* in1, Node* in2)

in1: nullptr

in2: 2

Call 1

```
if (in1 = nullptr) {
    return in2;
}
```

llrec(nullptr, 2)

In this instance, the initial call has in1 pointing to nullptr. As a result, the first condition is true

returns in2:

"2"

This ends the function and the resulting linked list is

2 → nullptr