UMPIRE Practice- Copy List with Random Pointer

U: Understand

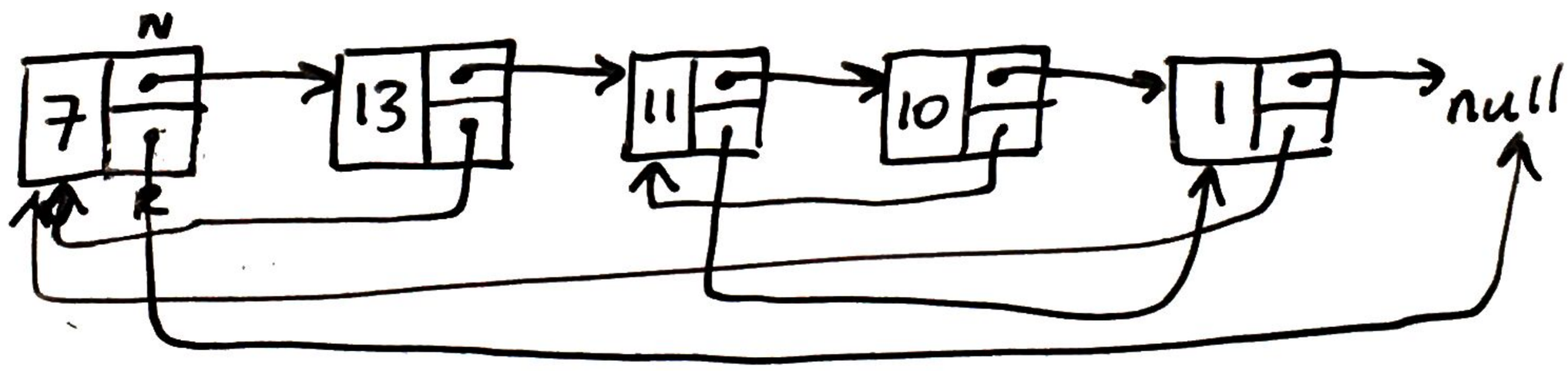
- deep copy → must create new objects
- numbers are not necessarily unique
- list could be null
- random may be any item in list OR null

M: match

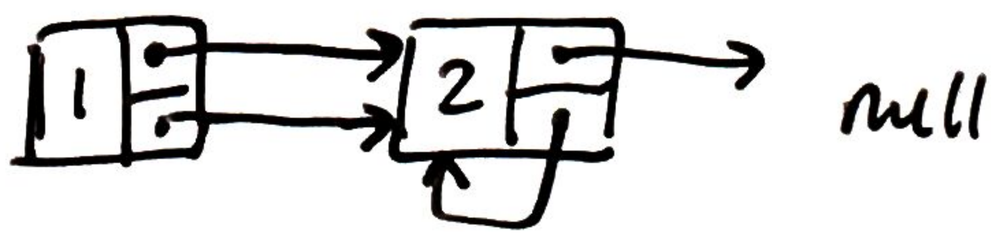
- linked list! that's a start!
- pointer bookkeeping? (i.e. make pointers to point at things I want to remember)
- dummy head?
- multipass?
- two pointers- no guarantee that there's a cycle (but could help if there is one)

P: plan

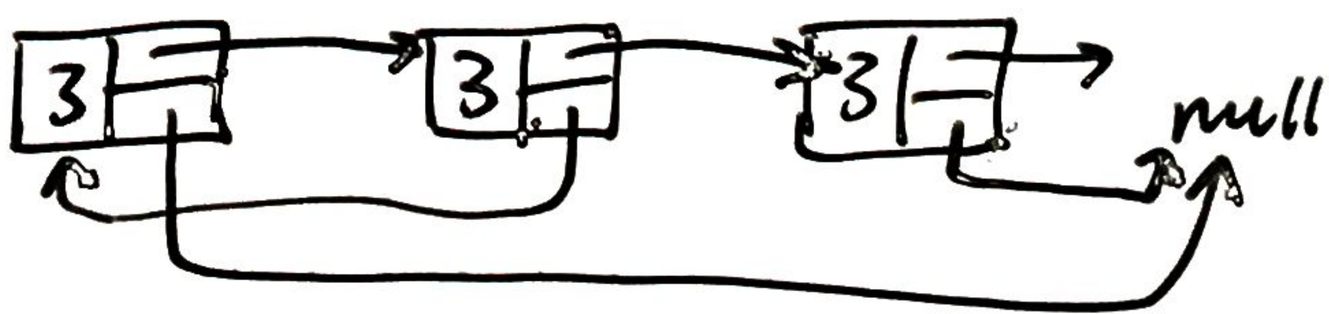
* oops I forgot to make test cases
  - happy path:



- edge case: null
- self cycle:



- same values:

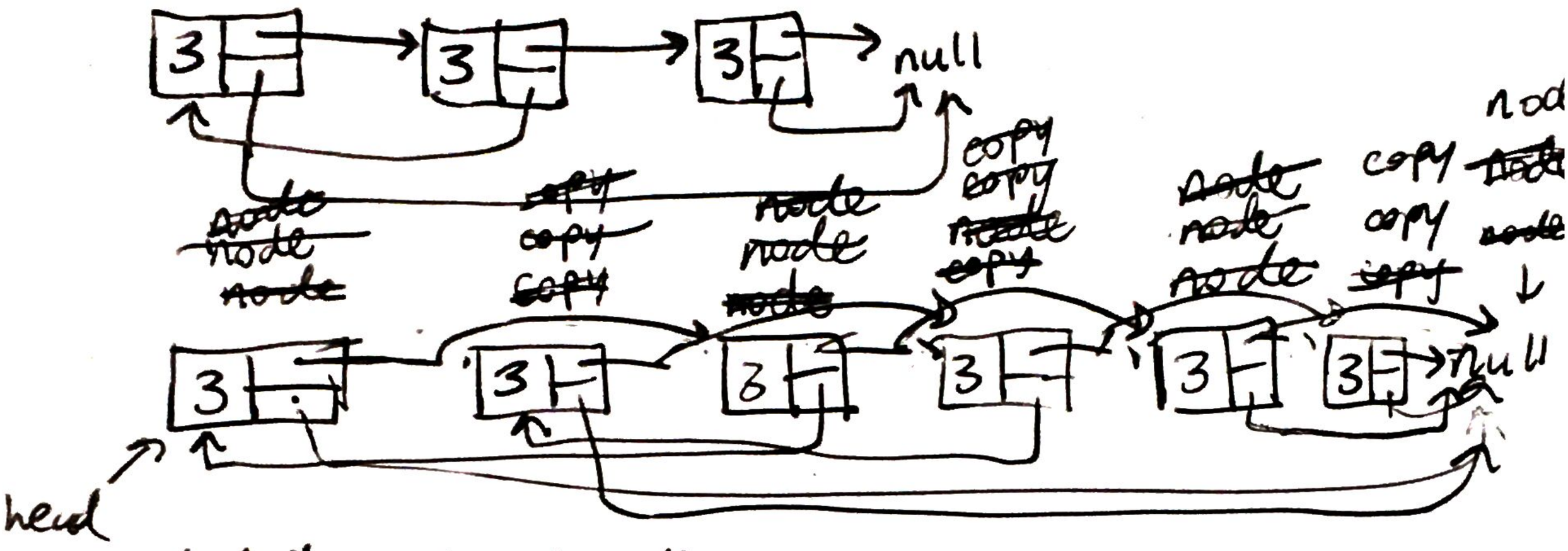


no cycle:

P: plan

1.) If head = null, return null.

2.) Else, list has lt nodes. First, create a copy of each node interweaved in the list:



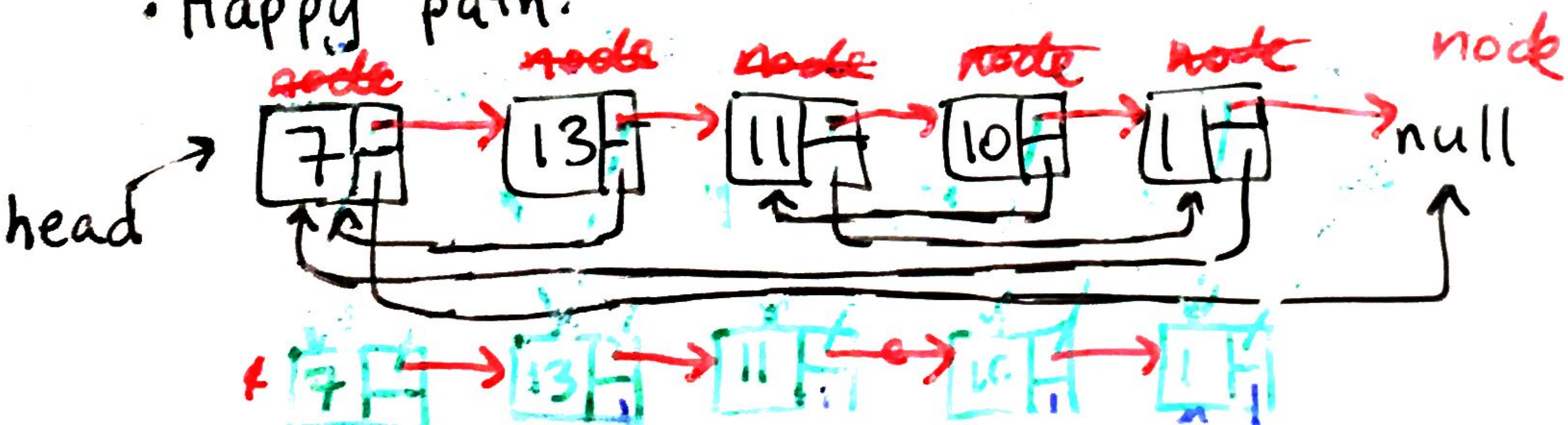

```
Node* node = head;
while (node != nullptr)
{
    // make a copy of the current node &
    // insert it into list
    Node* copy = new Node(node->val,
                          node->next);

    node->next = copy;

    // increment node to next item in list
    node = copy->next;
}
node = head;  Node* copy = node->next;
// now, iterate through list & copy random pointers
while (node != nullptr)
{
    if (node->random != nullptr)
    {
        copy->random = node->random->ne
    }
```

```
                    //move to next item
if (node != nullptr)  node = copy -> next;
                      copy = node -> next;
        }

        //finally, separate the interleaved lists
        // into two separate lists
        node = head;
        Node* new_head = node -> next;
        copy = node -> next;
        while (node != nullptr)
        {
            node -> next = copy -> next;
            node = node -> next;

            if ( node != nullptr)
            {
                copy -> next = node -> next;
                copy = copy -> next;
            }
        }

        return new_head;
```

I: implement - see LeetCode or attached file

R: review

- Happy path:



- Null: ✓
- Self cycle: ✓
- Same value: ✓
- No cycle: ✓

E: evaluate

- Time: $O(3N) \Rightarrow O(N)$
- Space: $O(1)$
- Improvements: use $O(N)$ space to improve runtime by using an array to keep track of indices of random pointers