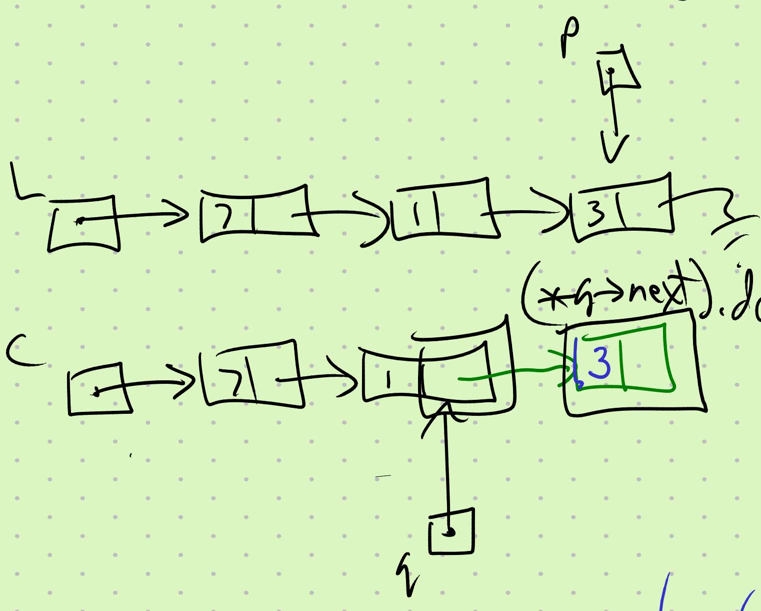
node* copy(node* L);

// copy list beginning at L.

// return pointer to beginning of copy.



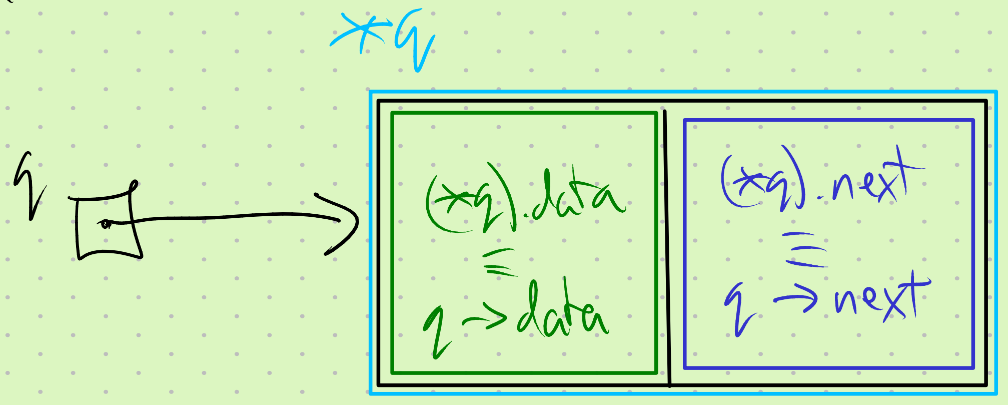
(*q->next).data ≡ q->next->data

q->next = new node;

q->next->data = p->data;

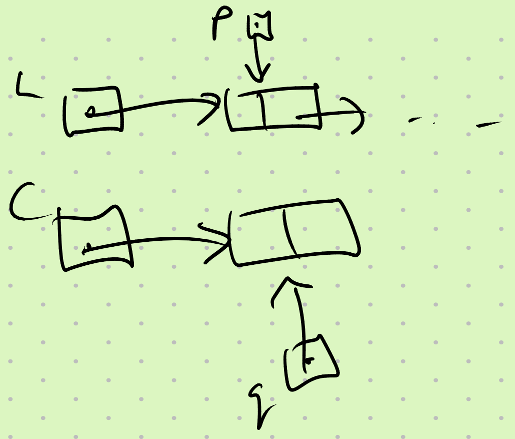
(*(q->next)).data

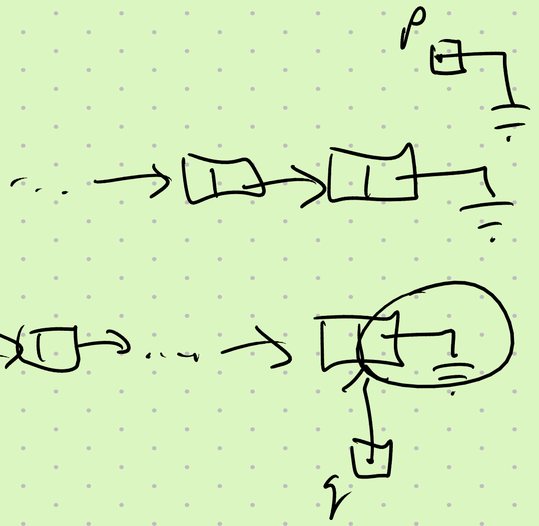
Remember:

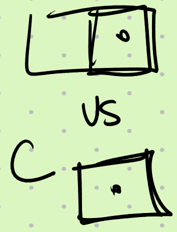


Details for copy function:

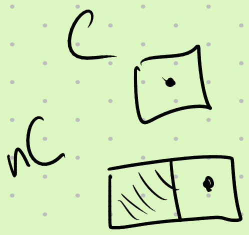
```
node* copy(node* L)
{
    node* C = NULL; // will point to beginning of copy
    node* p = L; // next node to be copied
    node* q = NULL; // last node in copy so far.
    if (L==NULL) return C;
    // set up first node...
    q = C = new node;
    C->data = p->data;
    p = p->next;
    while (p) {

        q->next = new node;
        q = q->next;
        q->data = p->data;
        p = p->next;
    }
    q->next = NULL;
    return C;
}
```







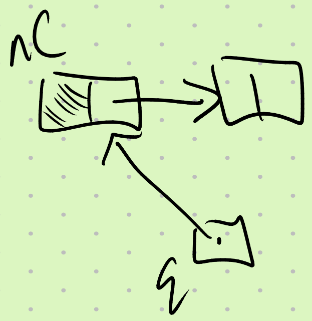
Note: in some sense, we need the special case at the beginning because C, L are NOT part of any node, unlike all other pointers in the list.



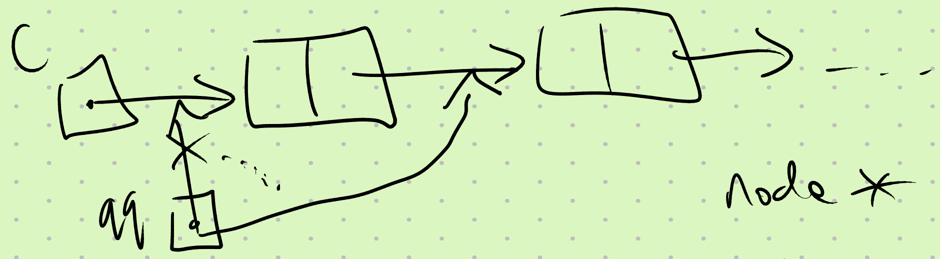
idea: put a "fake" node at the beginning...

Alternate version:

```
node* copy (node* L)
{   node nC;  // Not a pointer!
    node * p = L;
    node* q = &nC;
    while (p) {
        q->next = new node;
        q = q->next;
        q->data = p->data;
        p = p->next;
    }
    q->next = NULL;
    return nC.next;
}
```



One more approach: use pointers to pointers...



```
node * C; ...
node** q = &C;
```