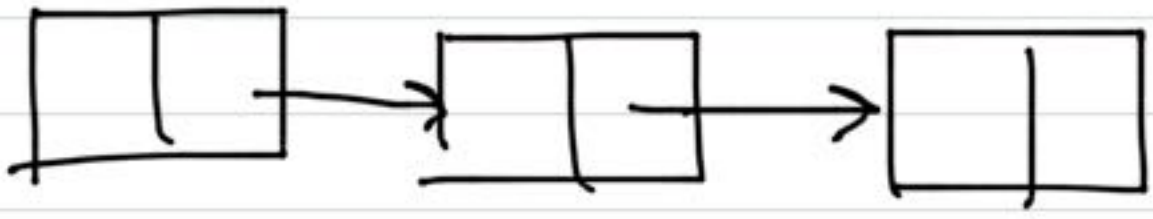
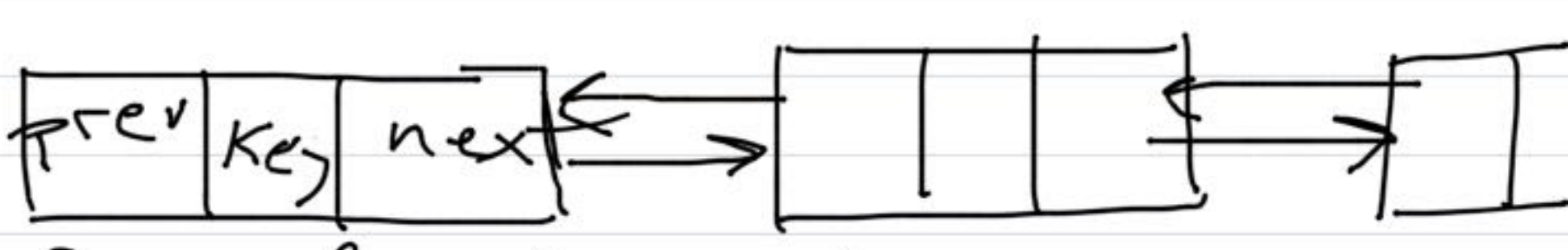
With single-linked list, next pointer only



With double-linked list, there is a next and previous pointer

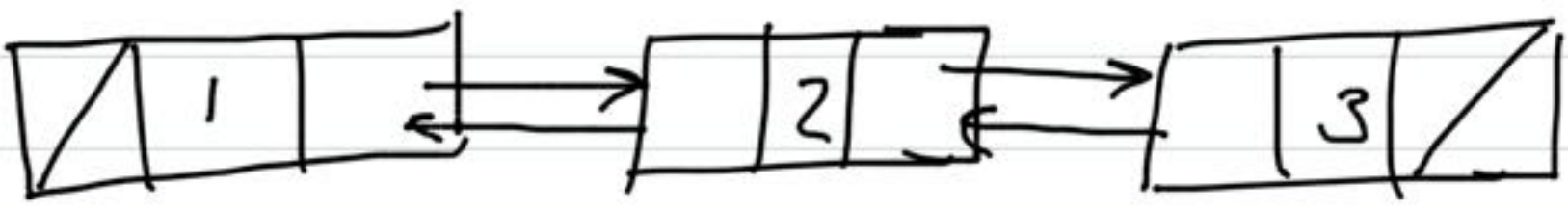


for head node

previous = NULL

for tail node

next = NULL



In code, we can use the previous pointer to traverse list in other direction.

Code example on Moodle Lecture9

```
struct node{
    int key
    node *next
    node *previous
}

node *head
head->key = 1
head->previous = NULL
head->next = NULL
```

We change next when we create a new node.

```
node *n1 = new node

n1 -> key = 2
n1 -> next = NULL
n1 -> previous = head
head -> next = n1
```

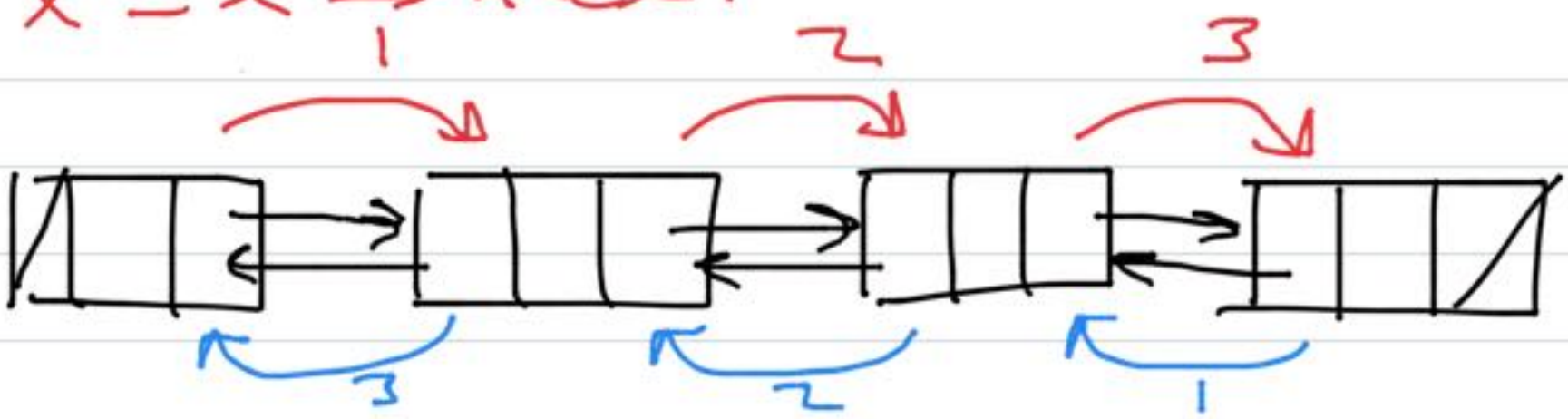
head n1

Build linked list with 5 nodes
Assume X exists

```
while(i<5)
    node *n = new node
    n -> next = NULL
    n -> previous = X
    n -> key = i
    X -> next = n
    X = n
        i++
```

The previous pointer also means we can traverse the list from last node to first node:

x = x → next



x = x → previous

Insert and Delete change pseudocode, pg. 38 CLRS book

```
List_insert (L, x)
x.next = L.head // put at front
if L.head != NIL // when is this
   L.head.prev = x   true?
L.head = x
x.prev = NIL
```

Differences between single and double LL for insert operation

Set L.head.prev to x
Set x.prev to NIL

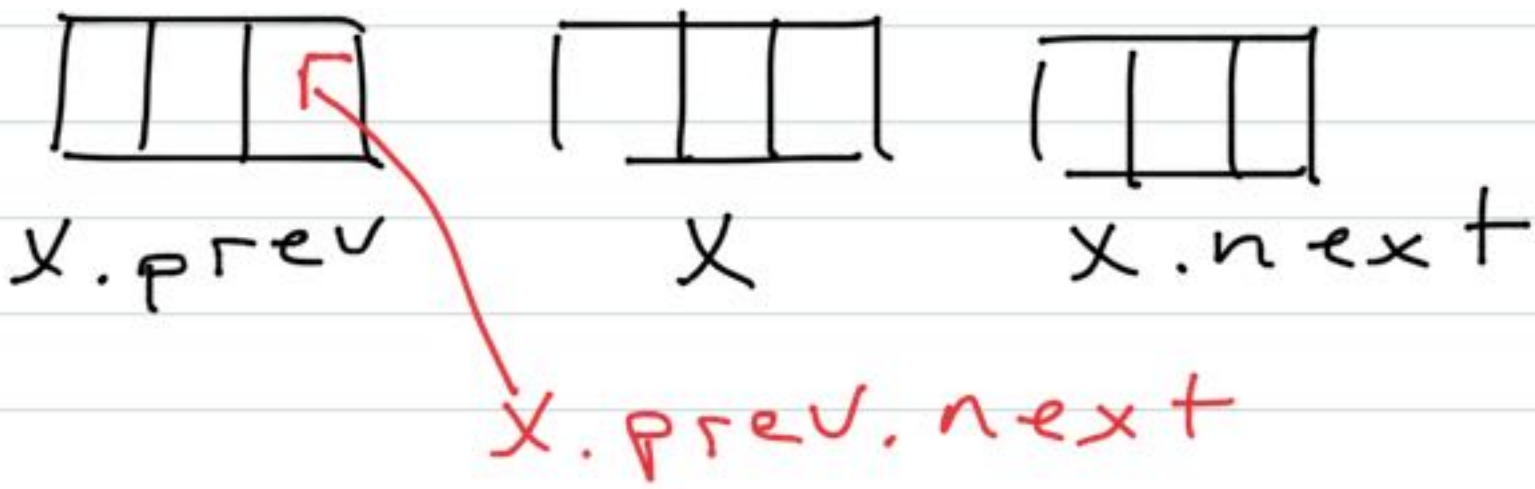
Read L.head.prev as
(L.head).prev

Delete operation
pseudocode: doesn't include memory management

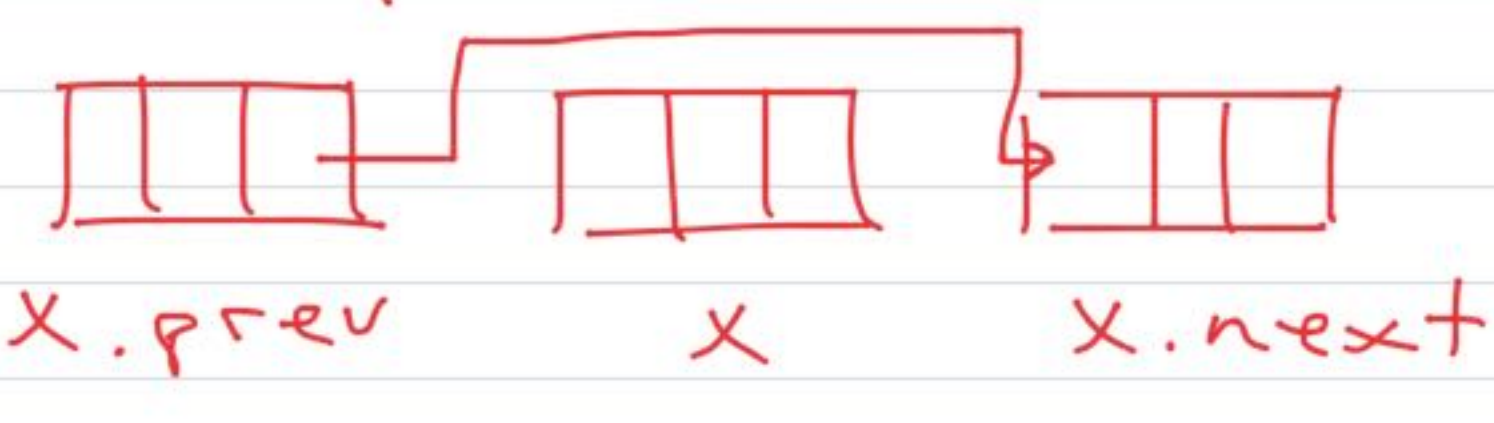
```
List_delete(L, x)
1 if x.prev != NIL   // head of list
2     x.prev.next = x.next
3 else
4     L.head = x.next
5 if x.next != NIL
6     x.next.prev = x.prev
```

Line 1: true when not head of list

Line 2: x.prev.next

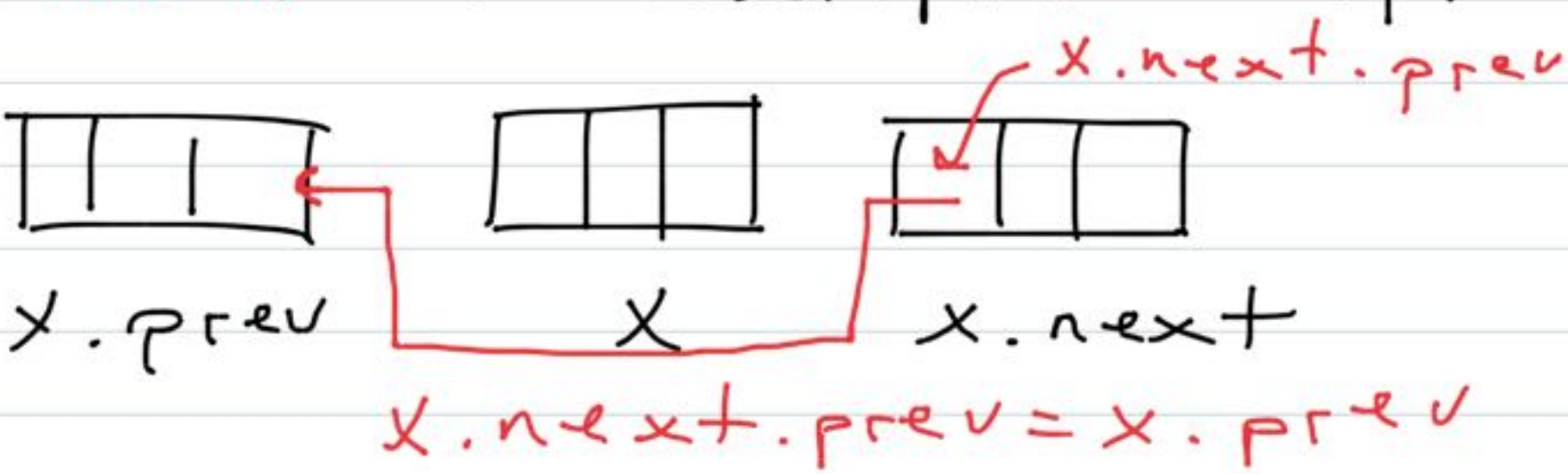


x.prev.next = x.next



Line 5: if x.next != NIL
true when not the last node in the list.

Line 6: x.next.prev = x.prev



What is a node?

So far, we've created variables of type node using a struct.

The nodes are simple, with pointers and one other variable, such as key or name.

But, the linked-list structure can have more complex nodes.

Consider this:

In this week's assignment you're adding cities to a CC.

What if cities could be no more than 50 miles apart when creating network.

What does the node look like?

What additional logic added?

Node needs to have position, such as (x,y) or (lat, long)

Adding node needs to check that

distance(city1, city2) < 50

Add city looks like:

get city1 coordinates

get city2 coordinates

calc distance

if distance < 50

add city

else

don't add city.

That would work, but it gets messy. More common is a more complex definition of node using classes instead of structs. Logic of checking distances hidden in class. You still write it, its just cleaner.

How classes work.

Define data types using class, (similar to structs)

Variables are private to within the class. In structs, variables are public.

We control access to private variables using public methods. Instead of setting variable directly, we call a method to set it. Error checking happens here.