Today - Lecture 13 - CS162

1) Remember Linear Linked Lists (LLL)

2) How to Create a LLL

   a) special cases

   b) insert at the beginning

   c) add at the end

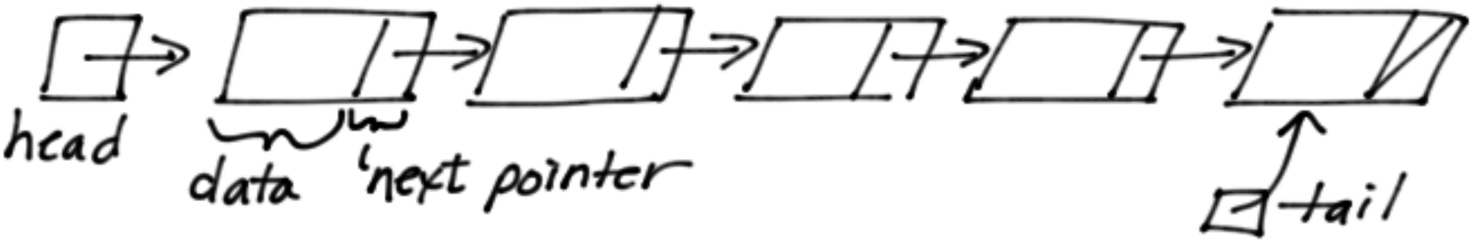
3) Demonstrating LLL in code (.h and .cpp)

4) Next... insert in sorted order

Announcements:

* PRACTICE!

Review of LLL



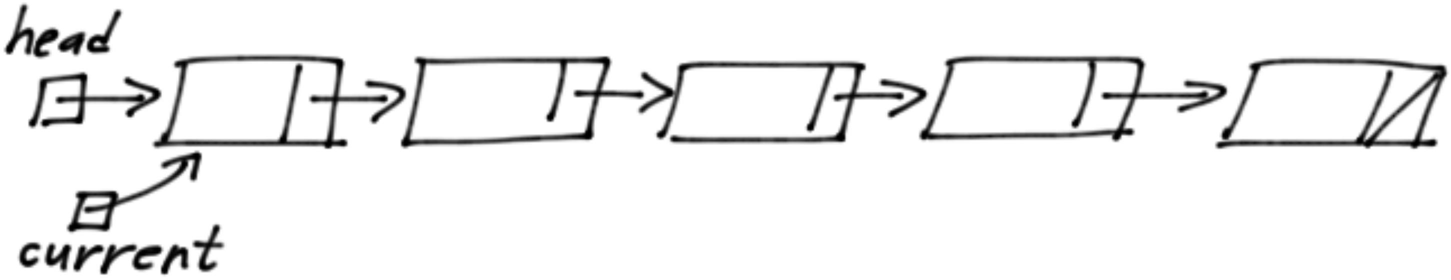
```
struct node
{
    video show;  // data
    node * next;  // a pointer to the next
                  // node; it is NULL if this
                  // is the last node
};
```

```
node *head;  // mandatory!
node *tail;  // possibly
```

To Traverse we need temporary pointer variables

```
node * current = head;  // starts off at the
                        // beginning of the list
```

Traversal



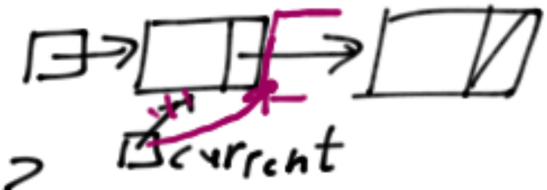
a) current = current→next;

(*current).next

address of the next node

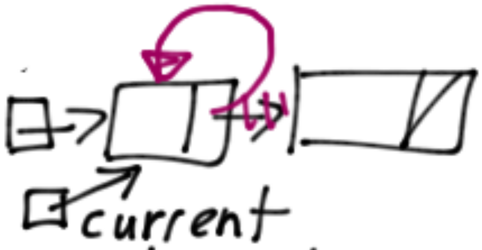
why not:

b) ++current ?



c) current→next; ?

d) current→next = current;

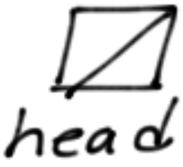


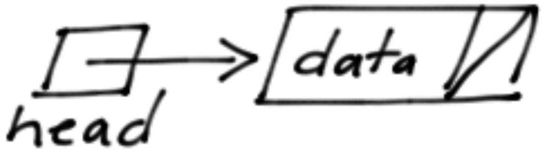
These compile but... draw the pointer diagram

Creating a LLL - special cases

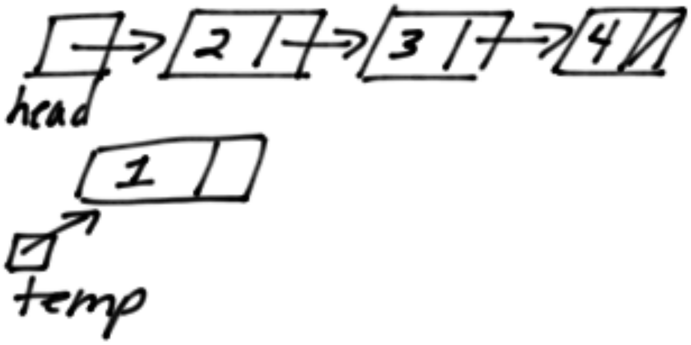
Before | After

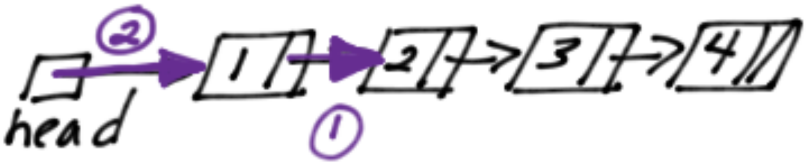
① Empty List



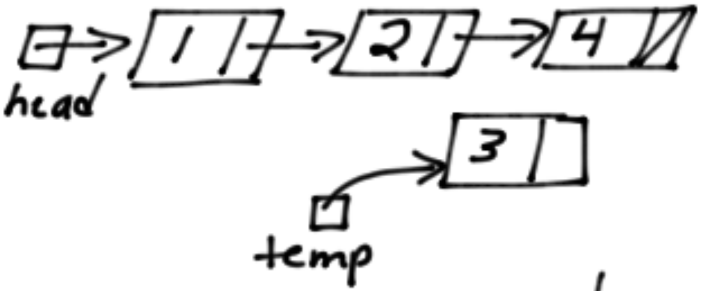


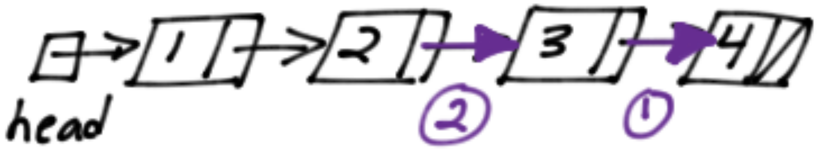
② Add at beginning



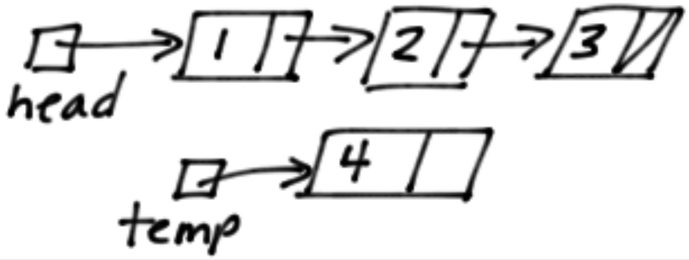


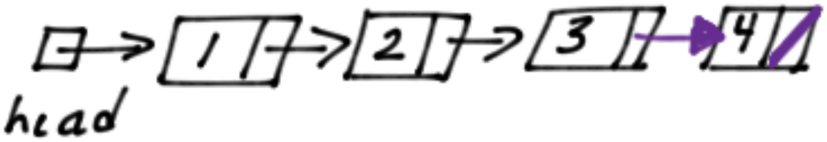
③ Add in the middle



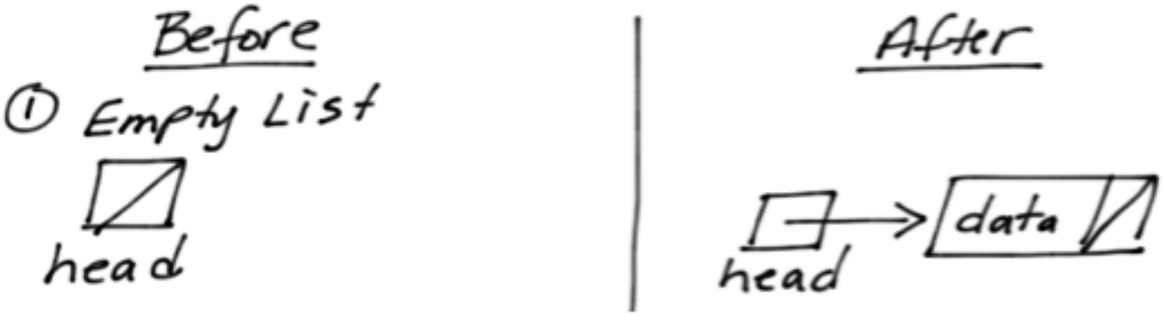


④ Add at the end





Inserting - into an Empty List



1) Detecting this case:

```
if (head == NULL)
```

watch out!

// OR

```
if (!head)   // true when head is NULL
```

2) Inserting:

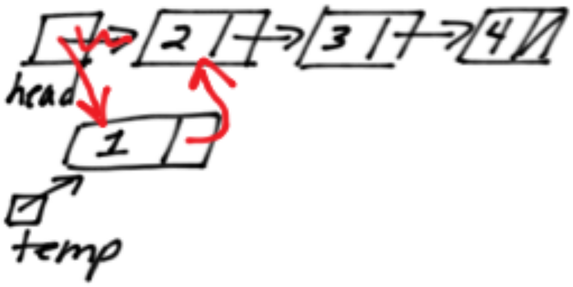
```
head = new node;
//store the data
head->next = NULL;   // important!
```

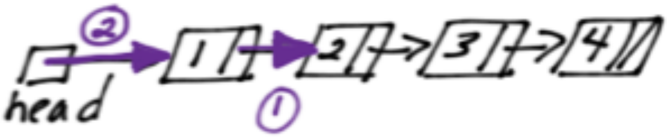
Inserting – at the beginning of a non-empty List

1) Detecting:

$$\text{if}(\underbrace{\text{head}}_{\text{true when head is }\underline{\text{NOT}}\text{ NULL}}\ \&\&\ \underbrace{\text{data\_being\_add} < \text{head}\to\text{data}}_{\text{conceptually}})$$

② Add at beginning





2) Inserting:

a) why not : head = new node ?

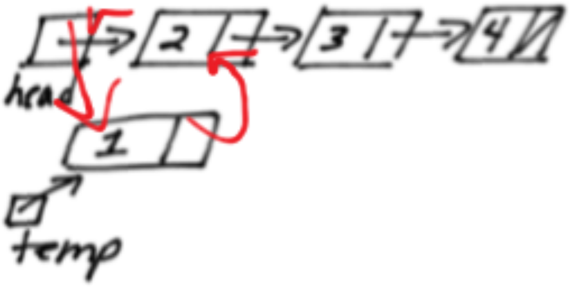
b) why not : temp = new node;
head = temp; ?

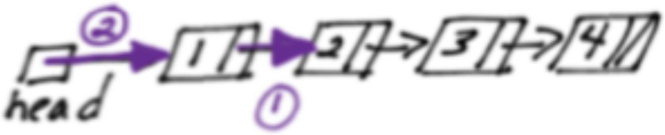
} List is Lost!!

c)
```
temp = new node;
//store the data
temp -> next = head;
head = temp;   // order is important
```

Use Caution !

② Add at beginning





What will this do?

```
node * temp;

temp = new node;
//store the data
temp -> next = head;
head = temp;

delete temp;
```

← WRONG

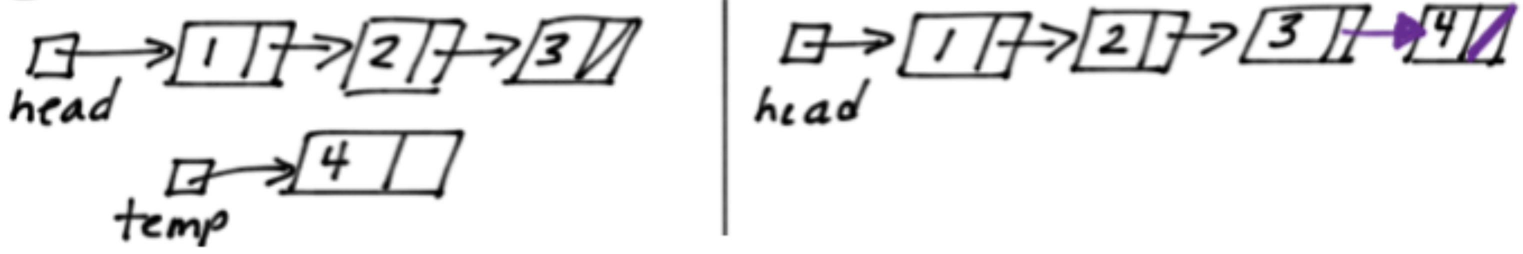
Release the memory that temp is pointing to!!!

Rule

ONLY USE DELETE WHEN **REMOVING**
not when you are adding!

Inserting – At the End

④ Add at the end



Traversal

1) Keep head locked on the first node

2) Use another local variable to assist with traversal

```
node * current = head;
```

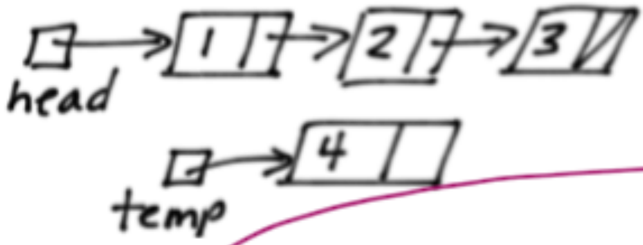
3) Traverse until:

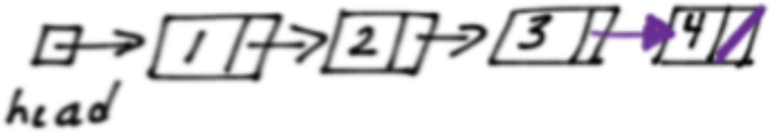
current → next $\boxed{\text{is}}$ NULL

4) Why not stop when current is NULL instead?

Inserting – At the End

④ Add at the end





```
node *current = head;
while (current != NULL)
      current = current->next;
```

current = NULL;

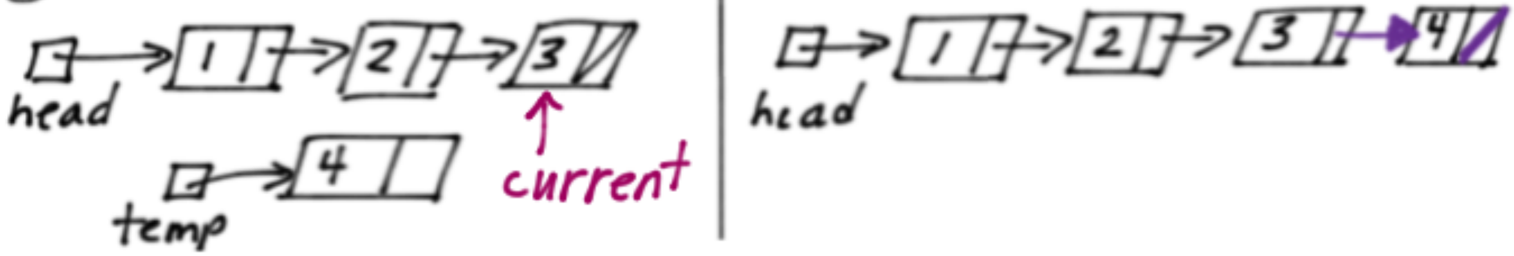
```
current->next = temp;
temp->next = NULL;
```

◁— SEG FAULT

Too Far!

Inserting – At the End

④ Add at the end



Inserting

1) Once traversal ends with current→next being NULL

2) Connect up the nodes by

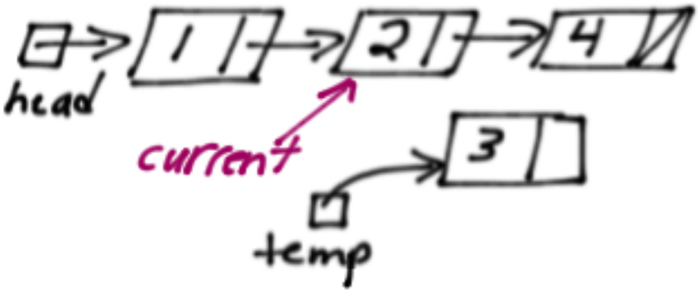
either order here is fine

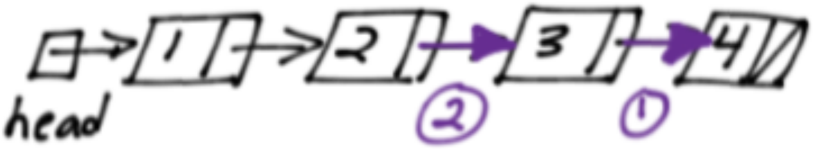
```
current→next = temp;
temp→next = NULL;
// save the data into temp
```

3) Vital that current not be NULL to begin

Inserting – in the midst

③ Add in the middle





1) First, make sure head is not NULL

2) Traverse to the right spot ...

3) Connect up the nodes (order is important)

Question

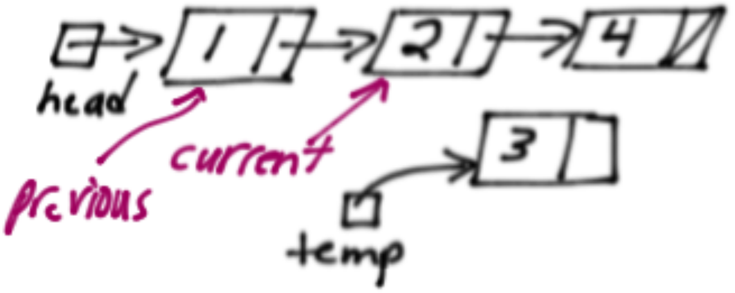
How do we know that it is time to insert?

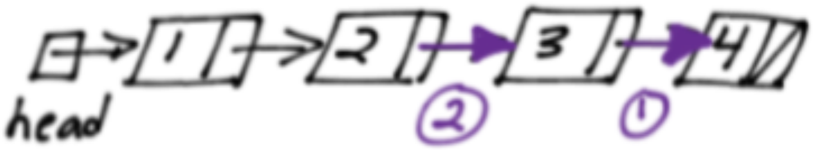
a) data is between current and current→next
"look ahead"

b) or, drag a previous pointer one node behind

Inserting – in the midst (with a previous pointer)

③ Add in the middle





Traversal

make sure we don't dereference a NULL ptr

```
while (current && not time to stop)
{       previous = current;
        current = current -> next;

}
```

Connect up

```
previous -> next = temp;
temp -> next = current;
```

Sample Problems from Lecture #13:

1. write the code to display JUST the last node's data in a linear linked list.

2. write the code to display EVERY OTHER node's data in a linear linked list

3. write the code to insert at the SECOND node (not at head ... but at head->next)

4. write the code to insert right BEFORE the last node.