CSI 310: Lecture 13

More visualization of singly and doubly linked list operations.
```c
if (HEAD == TAIL) {
    TAIL = t;
    if (TAIL == TAIL->TINK) {
        t->TINK = NULL;
        t->data = 33.3;
        node * t = new node;
        ... }
    }

    void set_tink(node * p) {
        node * tink = p;
        if (tink == TAIL) {
            TAIL = NULL;
            TAIL = tink;
            node * t = new node;
            ... }
        }

        node * p = NULL;
        node (const val-typ = val-typ);
        typedef double val-typ;
    }

    public:
    class node {
    ... }

    private:
    ... }

    Classical C Style

    OOP/Modern C++ Style

    Adding an item to the end: Requires modifying last node.

    University at Albany Computer Science Dept.
```
TAIL before dereferencing it, to set the last ink

// woe is you if you lose your
2. Rule of C/C++: The NULL pointer value is converted to boolean false

I. Since the last node of the original list must be modified (to replace the

II. When isn't there a last node?"

MUST be conditional on whether there IS a last node,

inside conditionals: all non-NULL values convert to true there. So,

... if (ptr == NULL) ... can be shortened to if (!ptr) ... , etc. For

Remarks:
node *t = new node(33.3);
if(TAIL)
    TAIL->set_lk(t);
TAIL = t;
if(!HEAD) HEAD = TAIL;

"does nothing here."
TAIL = t;
{
    HEAD = t;
    /* list is empty */ assert (HEAD);
    else
    {
        TAIL->LINK = t;
        /* list is not empty */ assert (HEAD);
        if (TAIL)

    Alternative (nicer) code:

University at Albany Computer Science Dept.
pointer to (addr of) a char

pointer to (addr of) a node

NULL

Another cut

This line's been cut

First main list text line

The third

0

2nd line

This line's been cut

0

MEDITCORE

We design your cursor and any others
pointer to (addr of) a char

pointer to (addr of) a node

This line's been cut

Another cut

First main list Text line

And line

The third

pointer to (addr of) a char

pointer to (addr of) a char

pointer to (addr of) a char

pointer to (addr of) a char

pointer to (addr of) a char
between TWO and TH3 ?

how can you insert the C-string containing "Dogs"?

Given this doubly linked list of 3 C-string addr's,
Given the original data structure, how can you change it to this one?
Determine char array length needed is 5, for "dogs"
2 Allocate char array (for C-string) saving its address.
3 Copy C-string into it.
4 Allocate a dnode, saving its address.
5 Copy char array's address into it.
6 Which means (tpn*) . data = tpc!
7 Copy char array's address into it.

```c
int main(){
    char *tpc = new char[5];
    tpc = new dnode;
    tpc->data = tpc;
    return 0;
}
```
Given the original data structure, how can you change it to this one???
Remember, we want to insert Two Dogs between the 2 circled nodes.

So we can modify their values. We NEED the addresses of Two and Three.

NULL
NULL
struct dnode {
    dnode* pfore; // The field (data member)
    dnode* pback; // names will be omitted
    char* data; // in our data structure
    // pfore==addr of the next dnode if any.
    // pfore==NULL if this is the last dnode.
    // pback==addr of the previous dnode if any.
    // pback==NULL if this is the first dnode.
    // data==addr of a C-string.
    // data!=NULL.
};

// the first dnode.
// the last dnode.
// previous dnode if any.
// next dnode if any.

The field (data member) names will be omitted in our data structure diagrams.
Alternatively, if we know the address of the right node, we can program: 

\[
\text{tpLeft} = \text{tpRight} \rightarrow \text{pback};
\]

Therefore, if we know the address of the circled node, we can calculate the address of the left node:

\[
\text{tpLeft} = \text{tpRight} \rightarrow \text{pfore};
\]

\[
\text{tpRight} = \text{tpRight} \rightarrow \text{pback};
\]
(1) Starting from value of head or tail traverse

(2) Use the value in a "cursor" variable.

or search the linked list forward or backward.

But, how can we obtain ANY one of these addresses?

(C++ declarations for these variables)

node * tpleft;
node * tpright;
Now a new dnode containing the addr. of the C-string is accessible via `tpn` AND 2 variables hold the addresses of dnodes we will put it between. HOW CAN WE CONNECT IT?

Relevant data str. is circled:
(1) Set 2 link fields in each of tpleft and ttright:
tpleft->pfore = tpright;
tpright->pback = tpleft;

(2) Set 1 link field in each of:
tpleft->pfore = tpn;
tpright->pback = tpn;

Only 2 more steps remain to connect *tpn (which is the new dnode) into the linked list:
*tpn : NULL
The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.

The result of step 1, tpn->tpnext=tpnext.
The result of step 1:

`tpn = tpn`.

The result of step 2:

`tpn = preore`. 

Volta! IS
The temporary pointer variables are not needed any more. (out of the picture)

The big picture after the changes:
A bigger picture after the changes:

Possible cursor feature:

MyeditorCore

Dogs

THis line's been cut\0

Another cut\0