Linear Linked List

LLL

- head
- Node
- data
- next pointer

Insert

- Insert at beginning
- Insert at end without a tail pointer
- " " " with " " "
- Insert in sorted order
1. head = NULL
2. head = new node
3. head->next = tmp
4. "Save the data?"
5. node * tmp = head
case 1: if (!head) //empty list

head = new node

//copy your data
head->data = NULL;

for (node = head; node->next != NULL; node = node->next)

//already have a list

else

temp = node->next

dup = node->next->next

temp->next = dup

dup->next = NULL

node->next = temp

dup = NULL

dup = new node

dup->data = node->data

dup->next = NULL

node = node->next

node->next = dup

dump = node
3

```c
head = NULL;

head = data; quantity = insert quantity;
step1 (head -> data, category, insert)
step2 (head -> data, title, insert, title)

video I am adding

assume this is a

int sum (int data, title) +1

head -> data, title = new char

head = new node;

if (!head)
    exit

if (head == NULL)
    head = NULL

if (head == NULL)
    f

} // case 1
```
case 2.

else

node = new node;

while (current-next !== null)

x = current.next;

(current.next = null); // Transferring to

the new node

current = current.next; // Transferring to

if (x.node) { current = current.next; x.node; }
Insert at end with a tail pointer

\#if (head)
\#else
    head = head.next;
    head = NULL;
\#endif

node *new = malloc(sizeof(node));
new->data = topl;
new->next = NULL;

if (top->next == NULL) {
    top->next = new;
}
else {
    node *temp = top;
    while (temp->next)
        temp = temp->next;
    temp->next = new;
}

if (top) {
    top = top->next;
}

free(top);

return head;
```plaintext
if (head == NULL)
    return;

while (current != NULL)
{
    if (current->next == head)
    {
        // Figure out when it goes
        temp = next = NULL;
        temp = move_read;
        node = node + temp = new node;
    }
    current = current + next;
}
```
class List {
    private:
        node * head;

    public:
        List();
        ~List();

    int display();
    int append();
    int insert_beg();
    int insert_at();
    int delete();
}

#include <iostream>

int main() {
    List list;
    return 0;
}
int first::insert-beginnings()
{
    if (first == NULL)
    {
        first = new node;
        first->next = NULL;
        return first;
    } else
    {
        first = new node;
        first->next = first;
        return first;
    }
}

return first = top;

if (read = top)
{
    top = top->next;
    /* some io data */
    top->next = read;
    node * temp = new node;
    temp->data = some_data;
    top = temp;
}

// first = NULL;
// first = first->next;

if (first == NULL)
{
    first = new node;
    first->data = "initial node";
}

first = first->next;
```plaintext
while (current->next != NULL)
    if (current == head)
        head = temp;
    else if (node->current = head)
    {
        node = temp;
    }
```
Practice

- Display the contents of LLL
- Display every other node
- Display the last node
- Display 2nd to the last node
- Count the # nodes