Wednesday, January 30, 2019 12:44 PM

Today:

- Finish Class w/ multi-file compilation example (Time12 class)

- The Node Structure

- Linked List
  - ↳ simple LL
  - ↳ LL class

Motivation:

Say we have an array w/ 1000 of its elements populated (Length 1001)

The array is sorted,

[1, 3, 4, 7, 12, ... 450]

n = 1001

We want insert an element, and keep array sorted. Say, new element value = 2.

How many memory copy operations are needed?

1000 !

Very inefficient

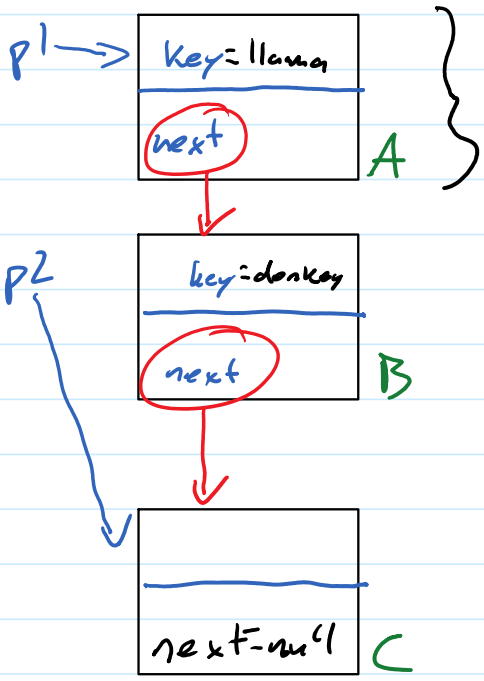
The Linked List Node

```
struct Node {
    string key;
    Node *next; // self referential pointer
};              // Point to its own type.
```

Recall that can have a pointer to a struct type:

```
Node *p1;
```

```
int main() {
    :
```



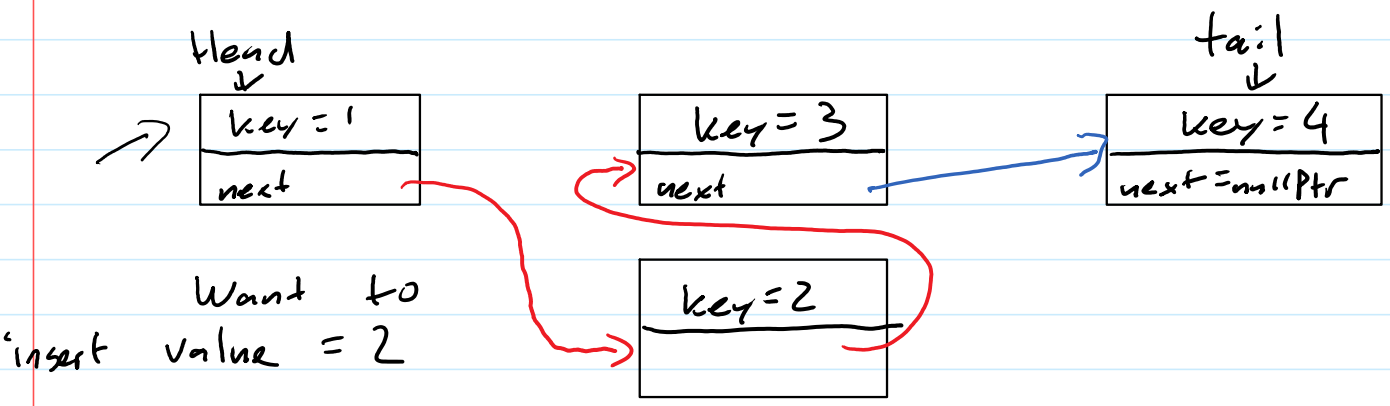

```
Node *p1 = new Node;
Node *p2 = new Node;
p1->key = "llama";
p2->key = "donkey";
p1->next = p2;
p2 = new Node;
cout << p1->next->key << endl;
```

result: donkey

```
// link up B and C
p1->next->next = p2;
```

Singly linked list:

node only knows about next node



Don't have to copy (move) all the elements!

The Linked List Class

- Define a struct for Node (singly linked)
  two members ① key
  ② self ref. pointer

- Define SLL class

  private data members:
  1) head
  2) tail

  public functions:
  constructor - no parameters
  - set pointers to null

  destructor - gets called when

object goes out of scope
- use it to de-allocate all the linked nodes

Node * search (string skey)
- locate a node w/ given value
- return pointer

void appendNode (string newItem)
- add new node to the end (tail)

void insert (string afterMe, string newItem)

void displayList ( )
- start at head
- traverse list and display every node key