Monday lec:
Security club on campus email Professor Long;

typedef struct node NODE

```
struct node {
        char *word;
        NODE *next, *prev;
}

p = newNode("foo")
```

Make sure to call malloc in newNode
Also read the whole assignment again. Just do it.

q -> next = list
q ->next->prev =q
q->prev = NULL
(doubly linked, I think I should do it this way)
list = the first node

strcmp(p.word,text) == 0
this will tell you if they are the same

FIND

```
while(p != NULL){
        if(strcmp(p->word,text) == 0){
                return p;
        }else{ //is this me being a bad javascript developer??
                p = p->next
        }
}
return NULL
```

Functions I'm going to want
newNode
findNode
addNode

delete:
gotta do three things

p->next->prev = p->prev
p->prev->next = p->next
free(p)
also make sure to check you aren't at the front or back of the list

nanosecond 10^-9 sec
gigaHz 10^9 things per sec on a cpu

p,q > 2^1024
pxq ≈ 2^2048

| time complexity | | sec | min | hr |
|---|---|---|---|---|
| c | constant | Notated O(1) | | |
| n | linear | 1000 | 6*10^4 | 3.6*10^6 |
| nlogn | | 140 | 4893 | 2*10^5 |
| n^2 | squared | 31 | 244 | 1897 |
| n^3 | cubed | 10 | 39 | 153 |
| 2^n | exponential | 9 | 15 | 21 |

there are worse things farther down

Church Turing thesis, any function that you can compute, (partally recursive function) can be done by a Turing machine.
Your computer is a universal machine

Algorithms matter because you want your code to finish running before you die.

an invariant is something you know is true.

Darrell saved IBM from the tyranny of Bubble Sort

f(n) = O(g(n))  (big O)
if for some n>N, f(n) ≤ c*g(n)
c is constant

If c is big you may want to use a smaller c agorithm on small input and the "better" algorithm on big inputs.

Talked about unordered search
Talked about binary searches log2n
Can we do better than a binary search? (talk about later, but I'd guess you need more information than just that it's sorted)
Quick sort -> binary = O(nlogn)+(logn)

Binary tree
If the tree is balanced then the tree is log(n) deep.
The tree is by definition ordered, so looking through it with checks is basically a binary search.

adversary is an algorithm's worst case scenario.
Binary tree's adversary is a sorted or reverse sorted list.
random is an average case and is pretty good.

"Optimize for the common case" David Sheridan
	Move to the front rule

C++ was once c with classes

| C++ | C |
|---|---|
| class | struct |
| new | malloc |
| delete | free |

```
typedef struct node NODE;
struct node {
	char *text;
	NODE *next;
}
typedef struct ll linkedList;
struct ll{
	NODE *ll;
	linkedList *self;
	void(*insert)(char*)
	etc...
}
```

```
#include “ll.h”
linkedList *newList(){
	linkedList * L = (linkedList)malloc(sizeof(linkedList))
	l->self = l
	l->list = (void*)0;
	l->
}
```