

- ). COMP1511 17s2 — Lecture 20 — The Missing Link
  - . admin: Don't panic!
  - !. morelinkedlists: more linked lists
  - !. morelinkedlists: working with multiple lists
  - !. morelinkedlists: don't forget: look out for NULL!
  - !. morelinkedlists: don't forget: look out for NULL!
  - !. morelinkedlists: don't forget: look out for NULL!
  - !. morelinkedlists: what if the list is empty?
  - !. morelinkedlists: a list can have nothing in it
- ). recursion: recursion
  - ). recursion: What is recursion?
    - . recursion: What's wrong with this code:
  - !. recursion: Classic example: factorial
  - !. recursion: Classic example: fibonacci
  - !. recursion: Two cases: base case, recursive case
  - !. recursion: Two cases: base case, recursive case
  - !. recursion: Two cases: base case, recursive case
  - !. recursion: Working with Linked Lists recursively

# COMP1511 17s2

## — Lecture 20 —

### The Missing Link

Andrew Bennett

<[andrew.bennett@unsw.edu.au](mailto:andrew.bennett@unsw.edu.au)>

more linked lists

stacks + queues

recursion

# Don't panic!

we're nearly there...

assignment 2 **in progress**

**player.c** released this week

don't forget to check **SPOTS**

**MandelbrArt** voting is up! (ends Sunday 23:59:59 this week)

# more linked lists

what if... we had **multiple** lists?

# working with multiple lists

comparing two lists?

combining two lists?

appending one list onto the other?

# don't forget: look out for NULL!

4

memory allocation can **fail**

you could be passed a **NULL pointer**

# don't forget: look out for NULL!

memory allocation can **fail**

```
Node newNode () {  
    Node new = malloc(1, sizeof (struct _node));  
    // What if malloc fails?  
    new->value = 17;  
}
```

you could be passed a **NULL pointer**

```
void someListFunction(List list) {  
    // What if list is NULL??  
    list->head->value = 17;  
}
```

# don't forget: look out for NULL!

memory allocation can **fail**

```
Node newNode () {  
    Node new = malloc(1, sizeof (struct _node));  
    // What if malloc fails?  
    if (new == NULL) {  
        // ... do something.  
    }  
    new->value = 17;  
}
```

you could be passed a **NULL pointer**

```
void someListFunction(List list) {  
    // What if list is NULL??  
    assert (list != NULL);  
  
    list->head->value = 17;  
}
```

# what if the list is empty?

7

`list == NULL` vs

`list->head == NULL`

one of these is **bad**

one of these is a **valid case** to consider

# a list can have nothing in it <sup>8</sup>

```
[ ] -> X
```

VS

```
[ ] -> 3 -> 1 -> 4 -> X
```

# recursion

in order to understand recursion,  
you must first understand recursion

# What is recursion?

in C programming: a function that calls itself

# What's wrong with this code:

```
int doSomething (int input) {  
    return doSomething (input);  
}
```

# Classic example: factorial

12

$$n! = n * n-1 * n-2 * n-3 * \dots * 2 * 1$$

# Classic example: fibonacci

13

```
fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
```

# Two cases: base case, recursive case

**base case:** when to stop

**recursive case:** how to keep going

# Two cases: base case, recursive case

**base case:** when to stop  
factorial?

$$1! = 1$$

**recursive case:** how to keep going  
factorial?

$$n! = n * n-1!$$

# Two cases: base case, recursive case

**base case:** when to stop

fibonacci?

$\text{fib}(0) = \text{fib}(1) = 1$

**recursive case:** how to keep going

fibonacci?

$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$

# Working with Linked Lists recursively

17

some things can be a lot easier with recursion