

pictures  
I  
took  
recently



# COMP1511/COMP1911

## Week 9!

M13B: 1pm – 4pm || T11X: 11am – 2pm

Tutors: William (me!) + Jason || Daniel

# My GitHub:

---



[https://github.com/william-o-s/unsw\\_comp1511\\_tutoring](https://github.com/william-o-s/unsw_comp1511_tutoring)

# simon says put your hand up if...

---

LOOKED  
at assignment 2

STARTED  
stage 1

FINISHED  
stage 1

FINISHED  
stage 2

# Tutorial Agenda:

---


Part 1

Part 2

Part 3

Let's remove heads/tails (in a C context) and explain:


```
dcc  
-leak-check  
program.c -o  
program
```



Let's Kahoot!

# Kahoot!

Let's remove heads/tails (in a C context) and explain:



# Let's remove heads/tails (in a C context) and explain:

---

```
dcc  
--leak-check  
program.c -o  
program
```



Let's Kahoot!

---

**Kahoot!**

# Let's remove heads/tails (in a C context) and explain:

## List evens

```
int list_evens(struct node *head1, struct node *head2)
```

Given two linked lists:

- return 0, if neither list contains even numbers.
- return 1, if one list contains even numbers, but the other does not.
- return -1, if both lists contain even numbers.

## List ordered insert

```
struct node *list_ordered_insert(struct node *head, int data)
```

Given a linked list that is ordered in ascending order and a value to insert, insert the value into the list that will allow the list to remain in ascending order.

## List delete smallest

```
struct node *list_delete_smallest(struct node *head)
```

Given a linked list, remove the node with the smallest value from the linked list and return the new head of the list.

## List copy

```
struct node *list_copy(struct node *head1)
```

Given a linked list, make a copy of the list and free the old list and return the new head of the list.

## List append

```
struct node *list_append(struct node *head1, struct node *head2)
```

Given two linked lists, append `list2` to `list1`.

## List reverse

```
struct node *list_reverse(struct node *head)
```

Given a linked list, reverse the list and return the new head of the list.

## Find intersection

```
struct node *list_find_intersection(struct node *head1, struct node *head2)
```

Given two linked lists, return a new list that is constructed of nodes containing any values that appear in both lists.

## Count occurrences

```
int list_count_occurrences(struct node *head, int data)
```

Given a linked list and a value, count the number of times that value appears in the linked list.





# VSCode Shortcuts

---

- Start with Ctrl+Shift+P
  - "Toggle Multi-Cursor Editor"
  - Convert text casing: (highlight text) → Ctrl + Shift + P → "Transform to ..."
- Multiple Cursors: Ctrl + Click anywhere
  - Cursor over multiple lines vertically: Shift + Alt + Click on line
- Duplicate Line: Ctrl + Shift + Alt + Up/Down Arrow
- Move Lines: Alt + Up/Down Arrow
- Change All Occurrences: Ctrl + Shift + L or Ctrl + D
- Indentation: (Highlight line/lines) → Ctrl + Left/Right Square Bracket
- Find and Replace: Ctrl + F → (click dropdown) → Replace next