COMP1511/1911 Programming Fundamentals

Week 9 Lecture 1

Linked Lists
A larger Application

Announcements

- **Assignment 1 Marks**:
  - Out early to mid week
- **My Experience**:
  - Keep an eye on your UNSW email for it we would love your feedback
- **Revision Sessions:**
  - Last set of revision sessions on next week
  - Look out for announcement and sign ups on the Ed forum soon

Week 10 Practice Exams

- Held in Labs
- This is how you get lab marks for week 10
  - Marks are based on attempting it.
- If you are in an online tut-lab
  - you can sign up for an in-person lab for week 10
  - sign up details coming soon on Ed forum.
- Don’t miss this chance to see what the exam environment is like and get used to it.

Last Week

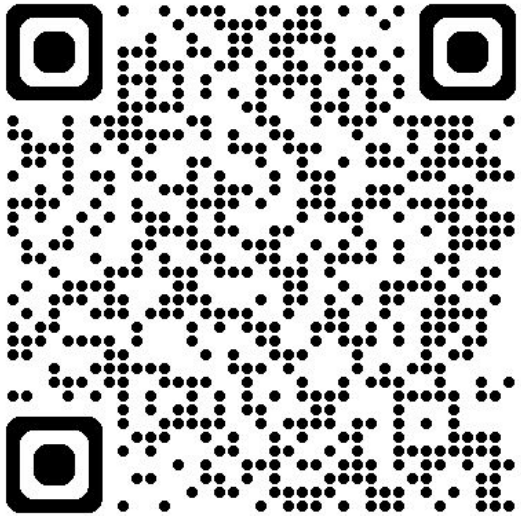
- Inserting Nodes anywhere
- Deleting Nodes
  - From the start of the list
  - Freeing all nodes
  - Search and Delete Approach 1

Today's Lecture

- Recap:
  - Linked List deletion First Node
  - Free all nodes
  - Linked List Search and delete approach 1
- Linked List Search and Delete
    - Second implementation
    - Extending first implementation to delete all occurrences
- Linked Lists a Larger Application.
  - Linked Lists with complex data (other than just int)
  - Multi-file Linked Lists
  - Helpful for assignment 2

Link to Week 9 Live Lecture Code

https://cgi.cse.unsw.edu.au/~cs 1511/24T3/live/week_9/

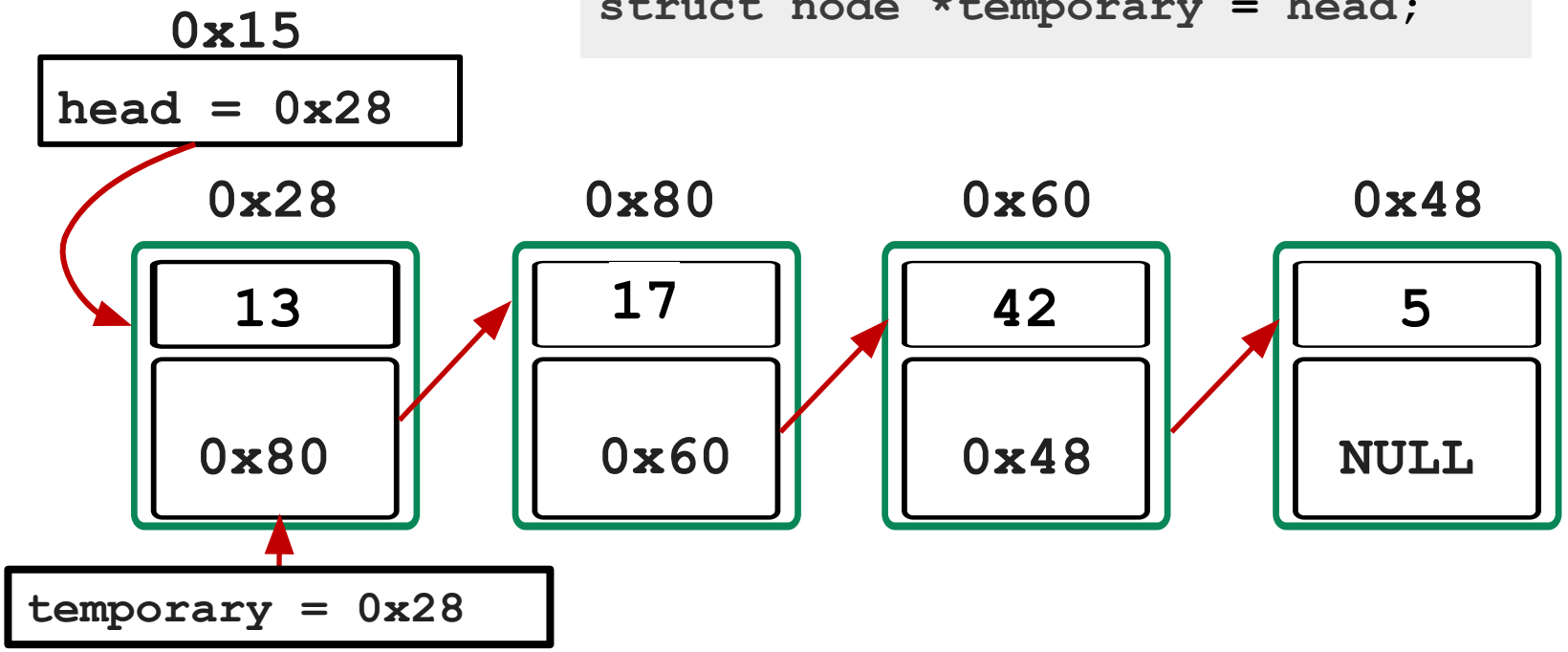


Deletion Recap

Deleting the First Node in a Linked List

Let's create a pointer to the first node

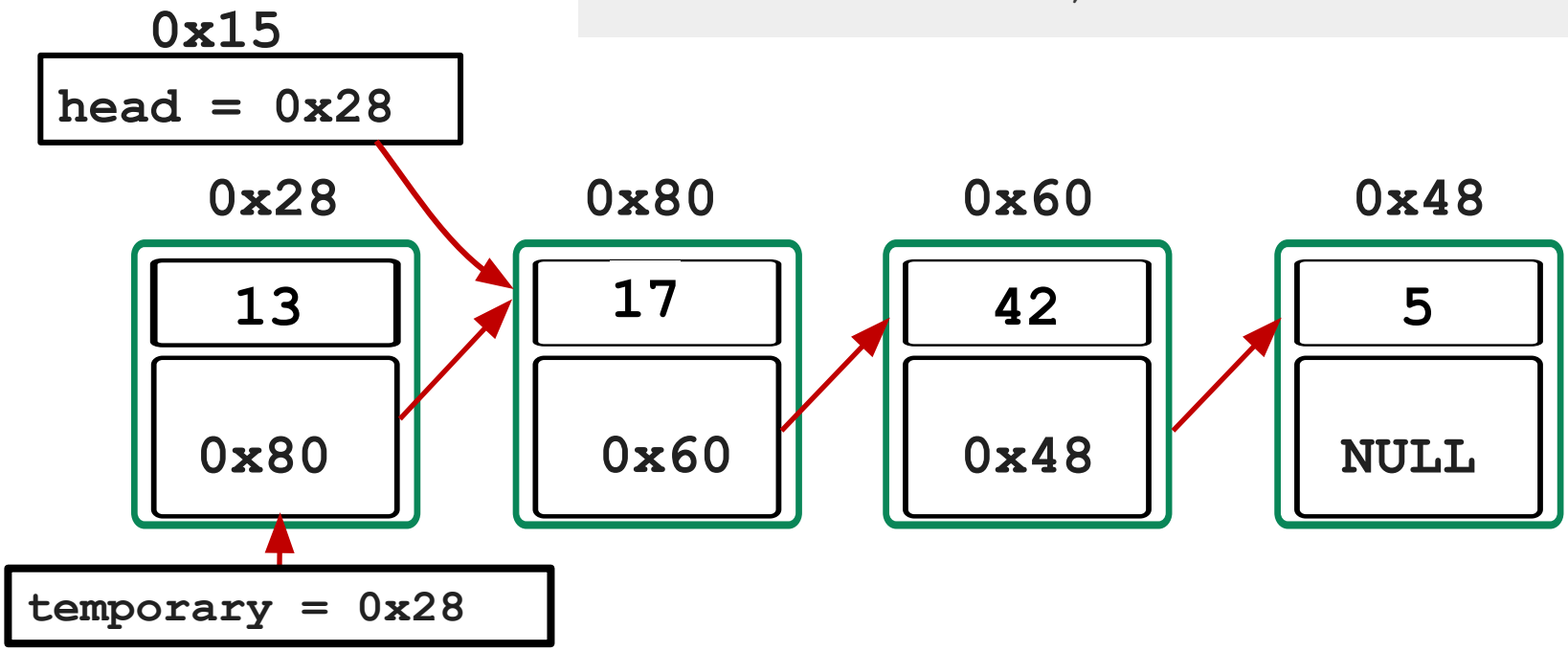
```
struct node *temporary = head;
```



Deleting the First Node in a Linked List

Now we can update head

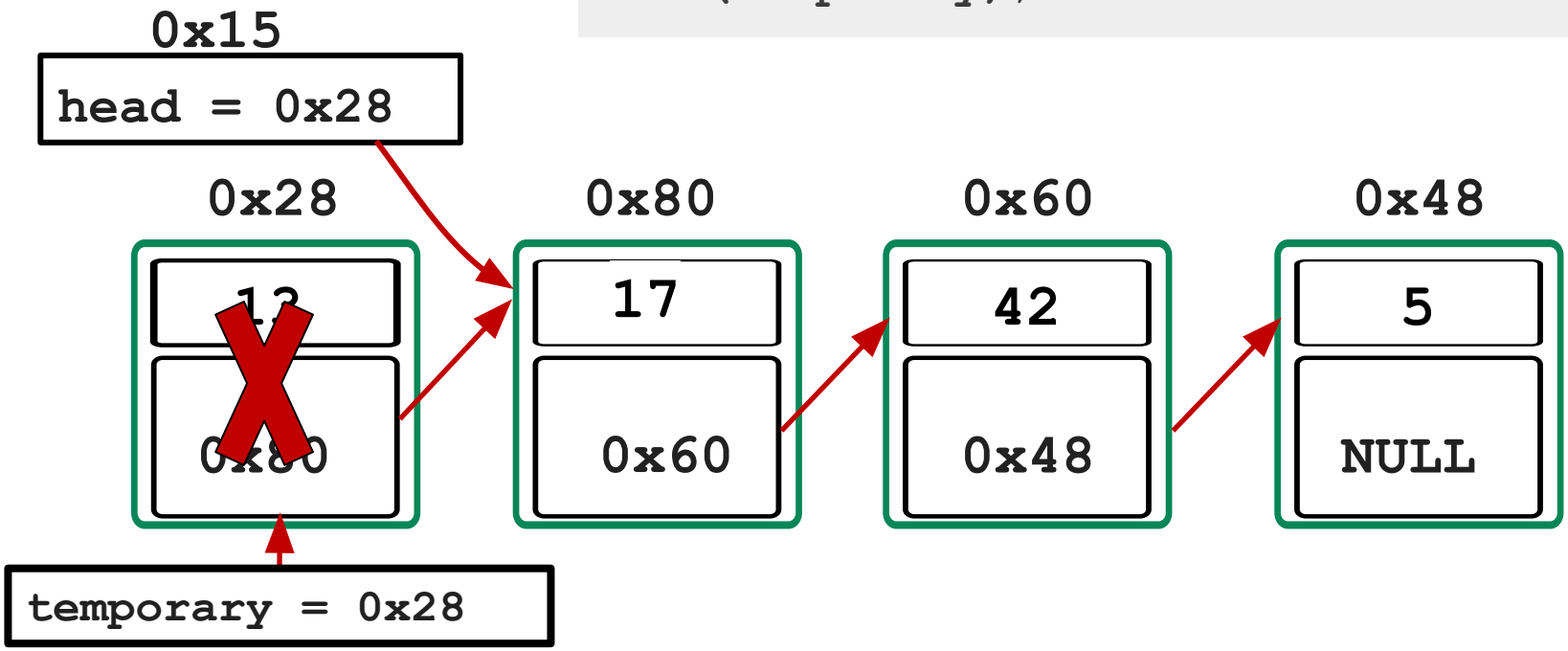
```
head = head->next;
```



Deleting the First Node in a Linked List

Now we can free the first node

```
free(temporary);
```



Delete All Nodes the Correct Way

Let's test it and check it with dcc –leak-check

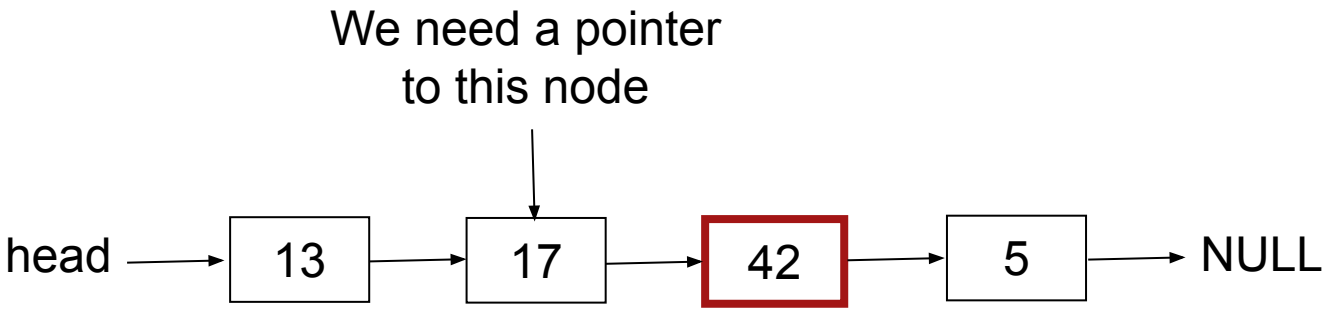
```
// Delete all nodes from a given list
void delete_all_nodes(struct node *head) {
    struct node *current = head;
    while (current != NULL) {
        head = head->next;
        free(current);
        current = head;
    }
}
```

Search and Delete

- We want to search for a node with a particular value in it and then delete it
- Where could the item be
  - Nowhere - if it is an empty list or the list does not contain the value
  - At the head (deleting the first node in the list)
  - Between any 2 nodes in the list
  - At the tail (deleting the last node in the list)
  - There could be multiple occurrences! For now let’s just consider the first occurrence

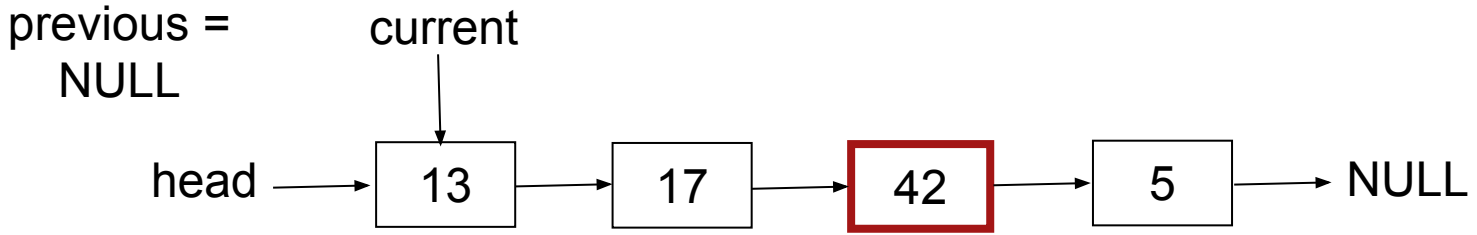
Search and delete: between 2 nodes

- To delete a node we need to link the previous node to the next node
  - If we want to delete the node with 42, we need to find the node before it



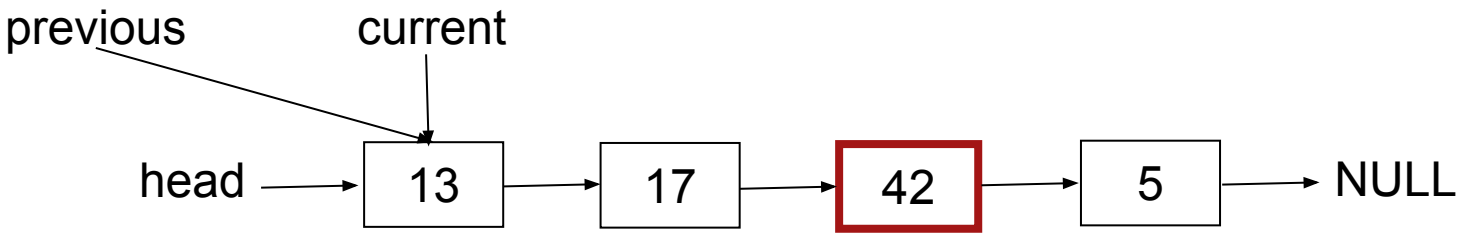
Search and delete: between 2 nodes

```
// Approach 1: Have a previous node pointer
struct node *previous = NULL;
struct node *current = head;
while (current != NULL && current->data != search_key) {
    previous = current;
    current = current->next;
}
```



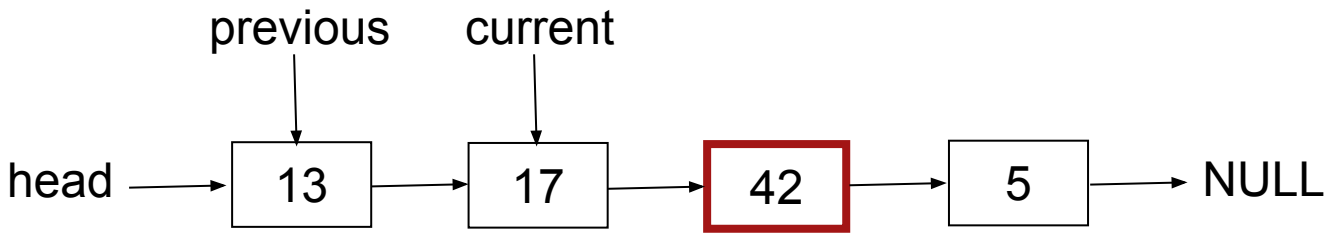
Search and delete: between 2 nodes

```
// Approach 1: Have a previous node pointer
struct node *previous = NULL;
struct node *current = head;
while (current != NULL && current->data != search_key) {
    previous = current;
    current = current->next;
}
```



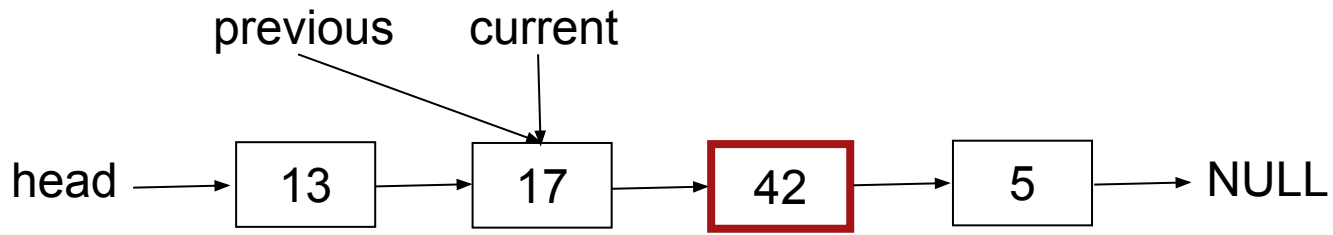
Search and delete: between 2 nodes

```
// Approach 1: Have a previous node pointer
struct node *previous = NULL;
struct node *current = head;
while (current != NULL && current->data != search_key) {
    previous = current;
    current = current->next;
}
```



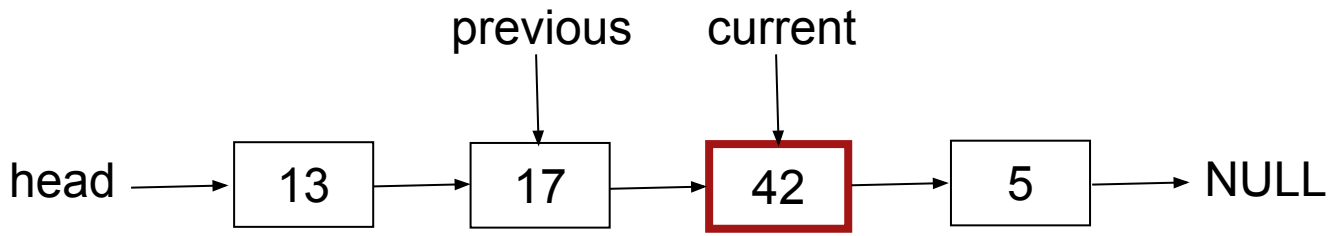
Search and delete: between 2 nodes

```
// Approach 1: Have a previous node pointer
struct node *previous = NULL;
struct node *current = head;
while (current != NULL && current->data != search_key) {
    previous = current;
    current = current->next;
}
```



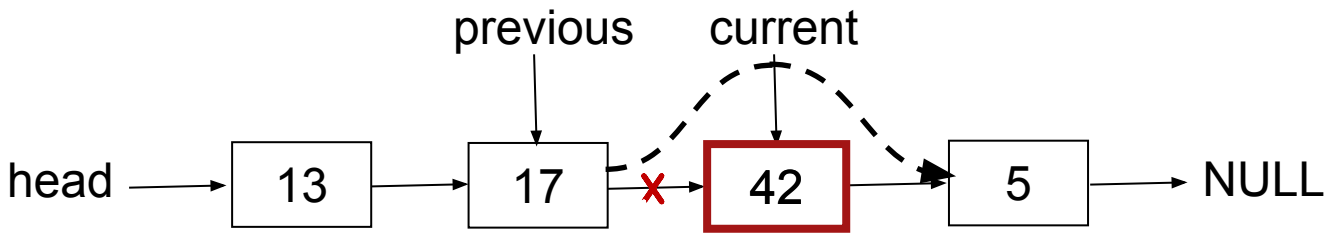
Search and delete: between 2 nodes

```
// Approach 1: Have a previous node pointer
struct node *previous = NULL;
struct node *current = head;
while (current != NULL && current->data != search_key) {
    previous = current;
    current = current->next;
}
```



Search and delete: Approach 1

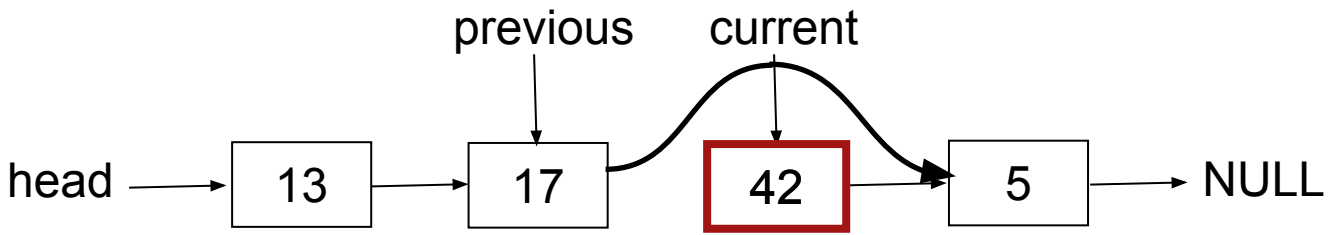
Then we need to connect current node to the one after the one we are deleting.



Search and delete: Approach 1

Then we need to connect current node to the one after the one we are deleting.

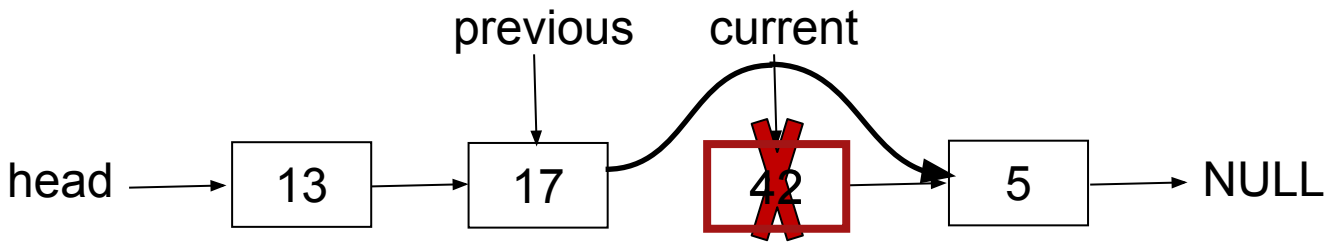
```
previous->next = current->next;
```



Search and delete: Approach 1

Now we can free the node we want to delete

```
free(current);
```



Coding

Let’s code up the second approach.

Let’s extend our first approach to delete all occurrences.

Email Management System

Email Management System

- Files/code provided (4 files):
  - email_management_system.c (TODO)
  - email_management_system.h (PROVIDED)
  - main.c (PROVIDED)
  - test_main.c (PROVIDED)
- Complete all `TODO` function definitions in email_management_system.c

Before you start coding

- Understand the Problem
  - what the provided code is doing
  - how it all fits together
  - how to compile it and run the code
- Draw diagrams
  - do this before/while coding each function too!
- Think about different test cases
  - do this before/while coding each function too!

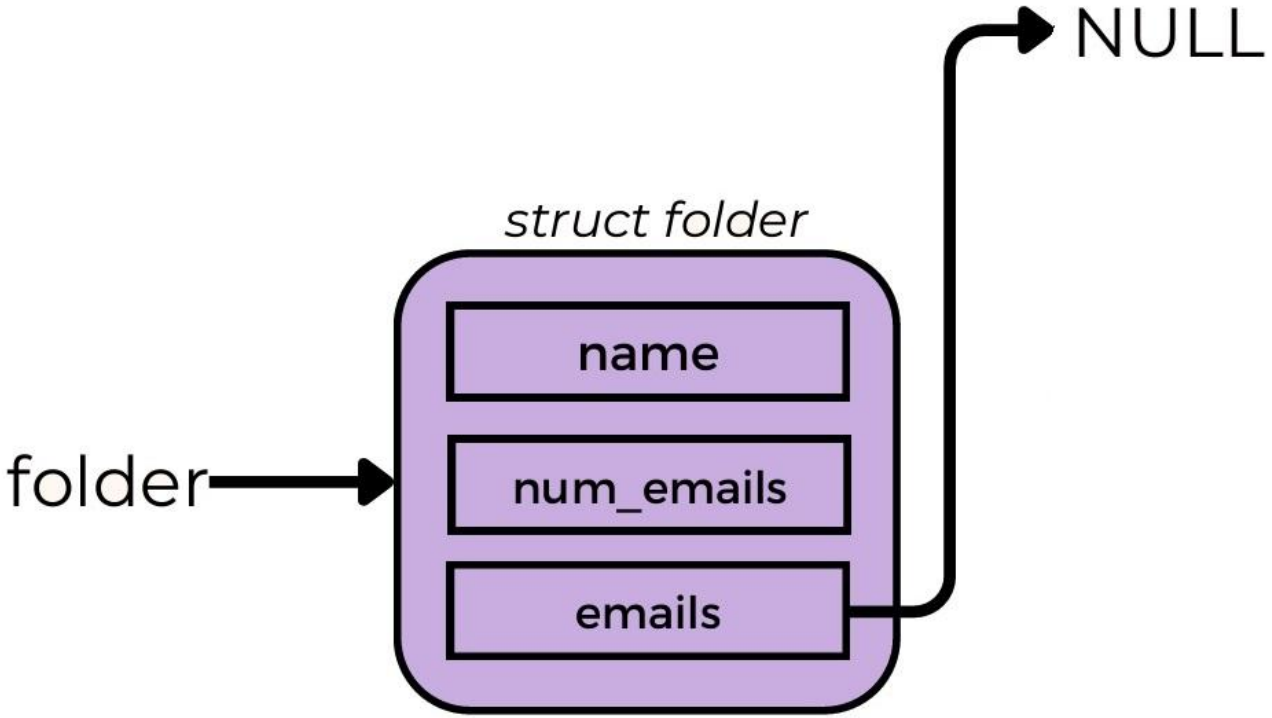
structs

```
struct folder {
    char name[MAX_LEN];
    //to use later :)
    //int num_emails;
    struct email *emails;
};
```

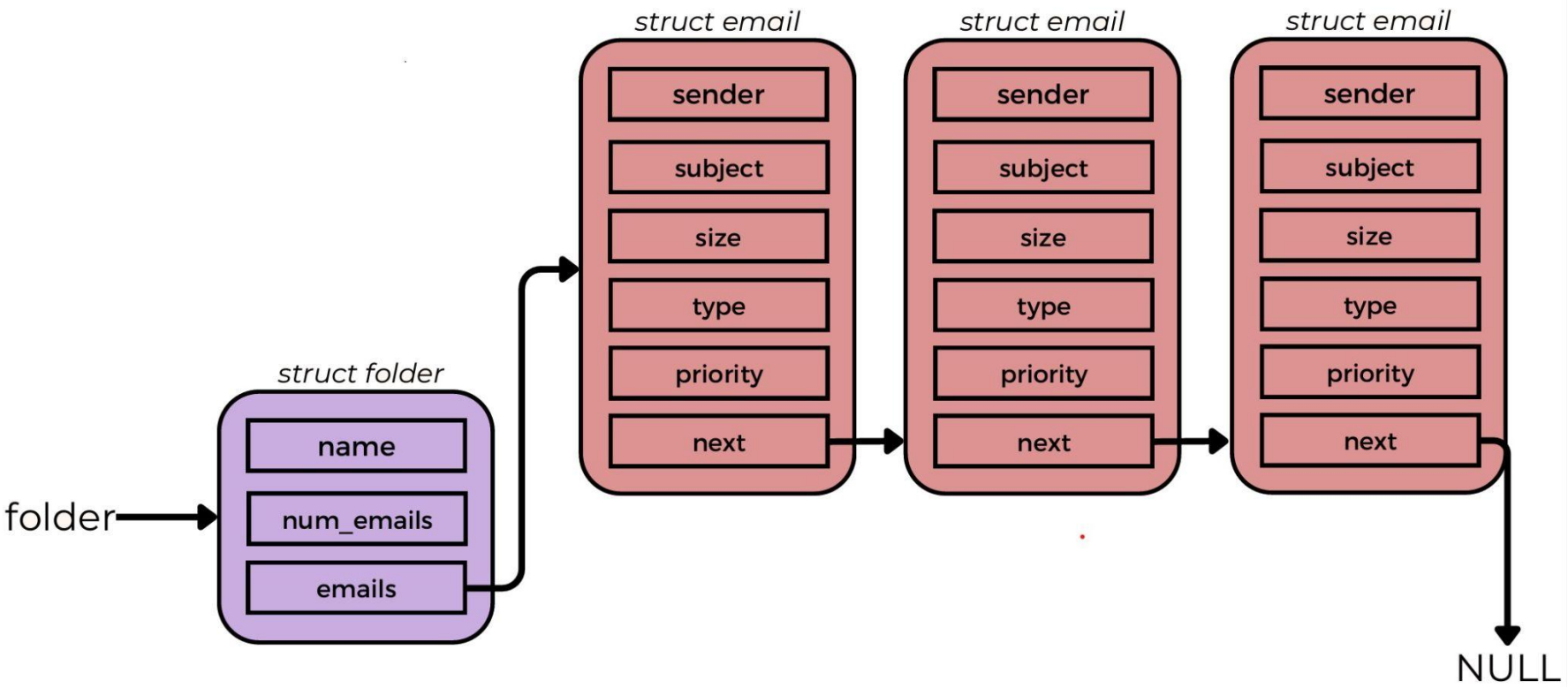
```
struct email {
    char sender[MAX_LEN];
    char subject[MAX_LEN];
    double size;
    enum email_type type;
    enum priority_type priority;
    struct email *next;
};
```

Which struct represents a linked list?

Visualisation of the system



Visualisation of the system



structs

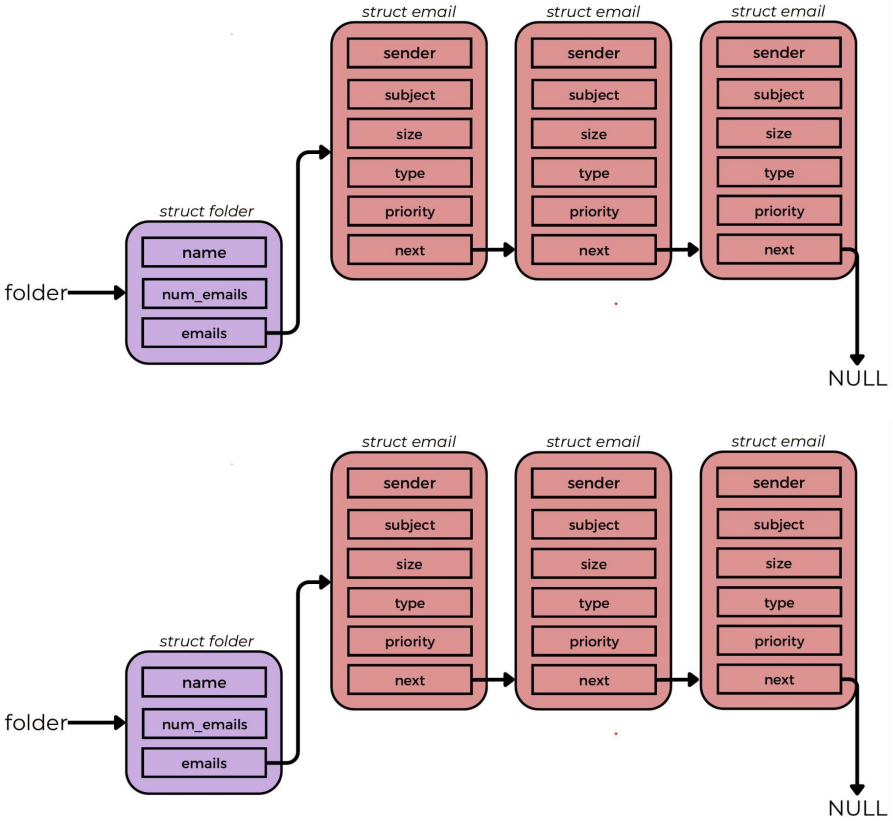
```
struct folder {
    char name[MAX_LEN];
    //to use later :)
    //int num_emails;
    struct email *emails;
};
```

```
struct email {
    char sender[MAX_LEN];
    char subject[MAX_LEN];
    double size;
    enum email_type type;
    enum priority_type priority;
    struct email *next;
};
```

If I have a pointer to a struct folder named my_folder, how could I access the head of the linked list?

Visualisation of the system

We can create many folders, each containing linked lists of emails.



Compiling and running the code

- We have 2 files that contain main functions.
- We can only have 1 main function per program.
- We can compile and run the first program as follows:

```
dcc -o test_main test_main.c email_management_system.c
./test_main
```

- We can compile and second program as follows:

```
dcc -o main main.c email_management_system.c
./main
```

Functions to Write

- Stage 1
  - create_folder
  - insert_email_at_head
  - search_email
  - clear_folders
- Stage 2
  - delete_email_of_priority
  - merge_folders
  - split_folder

Extensions

- Modify the implementation so that finding size of list is more efficient by storing the size in the folder as a member.
  - You will need to make sure you update this value whenever you add delete emails in the folder
- Sorting email lists
  - Sorting algorithms are not required knowledge for this course. However you could take every node from one list and insert it into a new list in order to get a sorted list.
- Creating an email management system struct that contains multiple folders

What did we learn today?

- Recap
  - Linked List Deletion
  - Implement search and delete approach 2
  - Extend approach 1 to delete all occurrences
- Larger Linked List Application
  - Multi-file program
  - Linked lists used inside of other structs
  - Linked lists containing complex data

Feedback Please!

Your feedback is valuable!

If you have any feedback from today's lecture, please follow the link below or use the QR Code.

Please remember to keep your feedback constructive, so I can action it and improve your learning experience.

https://forms.office.com/r/xdhUUfVSN7

Next Lecture: Your choice

- Revision OR
- Non-examinable topic recursion: poll

Reach Out

Content Related Questions:
Forum

Admin related Questions email:
cs1511@unsw.edu.au

Don’t forget to attend Help Sessions
if you need one on one help

Struggling with non-course specific issues?

My Feelings and Mental Health
Managing Low Mood, Unusual Feelings & Depression

- **Mental Health Connect** — student.unsw.edu.au/**counselling** — Telehealth
- **Mind HUB** — student.unsw.edu.au/**mind-hub** — Online Self-Help Resources

In Australia Call Afterhours
UNSW Mental Health Support Line — 1300 787 026, 5pm-9am

Outside Australia Afterhours
24-hour Medibank Hotline — +61 (2) 8905 0307

Uni and Life Pressures
Stress, Financial, Visas, Accommodation & More

Student Support
Indigenous Student Support
– student.unsw.edu.au/**advisors**
– nura-gili-centre-indigenous-programs

Reporting Sexual Assault/Harassment

Equity Diversity and Inclusion (EDI)
– edi.unsw.edu.au/**sexual-misconduct**

Educational Adjustments
To Manage my Studies and Disability / Health Condition

Equitable Learning Services (ELS)
– student.unsw.edu.au/**els**

Academic and Study Skills

Academic Skills
– student.unsw.edu.au/**skills**

Special Consideration
Because Life Impacts our Studies and Exams

Special Consideration
– student.unsw.edu.au/**special-consideration**