COMP1511 PROGRAMMING FUNDAMENTALS

LECTURE 13

Time to delete from a linked list

LAST WEEK...

- Linked Lists -
  - creating a list
  - inserting nodes at the head
  - traversing a list
  - inserting nodes at the tail
  - inserting a node between two other nodes

TODAY...

- Linked Lists -
  - deleting nodes in a list
    - at the head
    - at the tail
    - in the middle
    - with only one item in a list
  - harder example of linked lists

“

WHERE IS THE CODE?

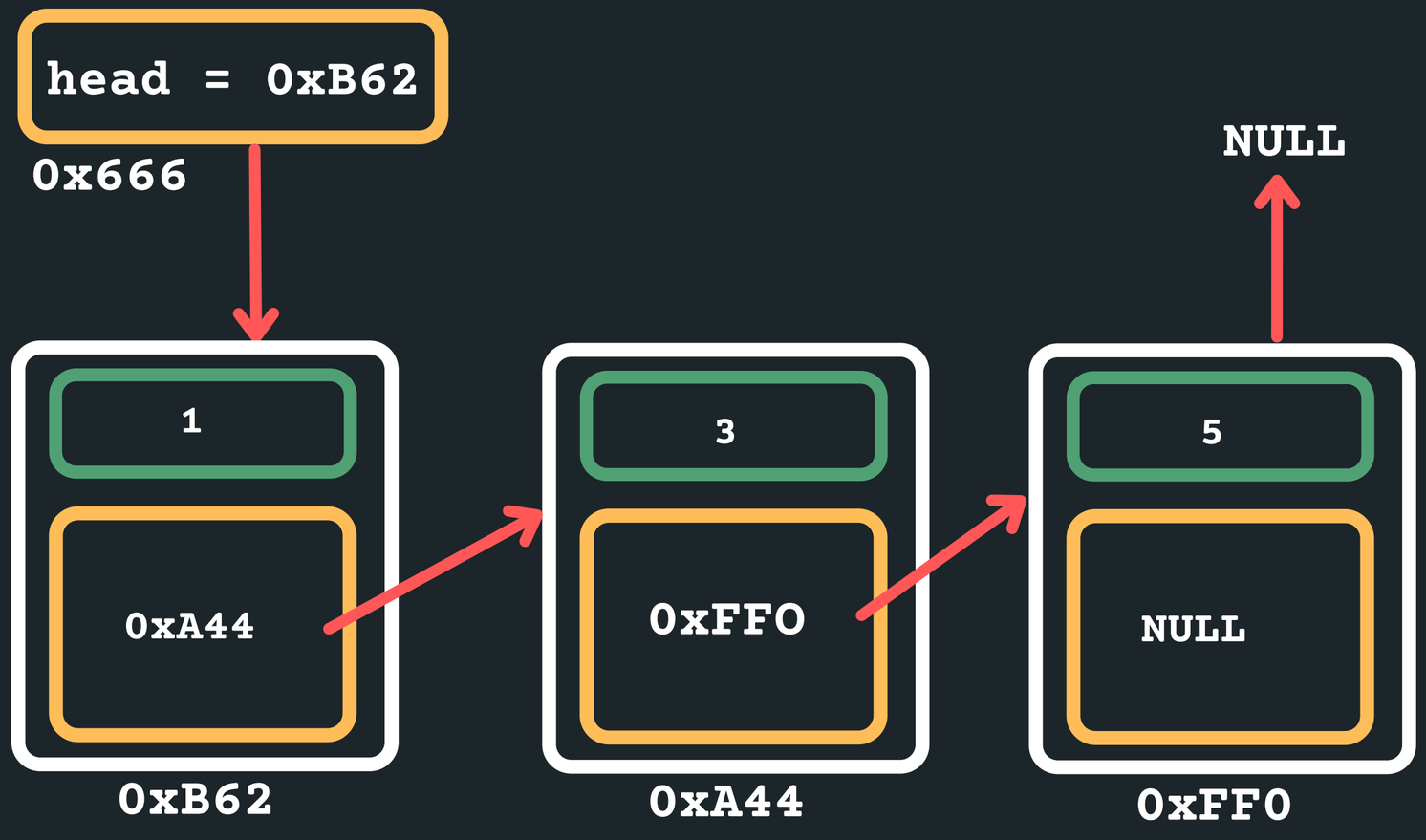
Live lecture code can be found here:

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/22T1/LIVE/WEEK08/

RECAP OF LINKED LISTS

INSERTING

- Where can I insert in a linked list?
  - At the head
  - Between any two nodes that exist
  - After the tail as the last node



RECAP OF LINKED LISTS

INSERTING

- I can also have other conditions, for example insert in the middle of a list, insert after the first element of a list, etc

LINKED LISTS

DELETING

- Where can I delete in a linked list?
  - Nowhere (if it is an empty list - edge case!)
  - At the head (deleting the head of the list)
  - Between any two nodes that exist
  - At the tail (last node of the list)

LINKED LISTS

DELETING EMPTY LIST

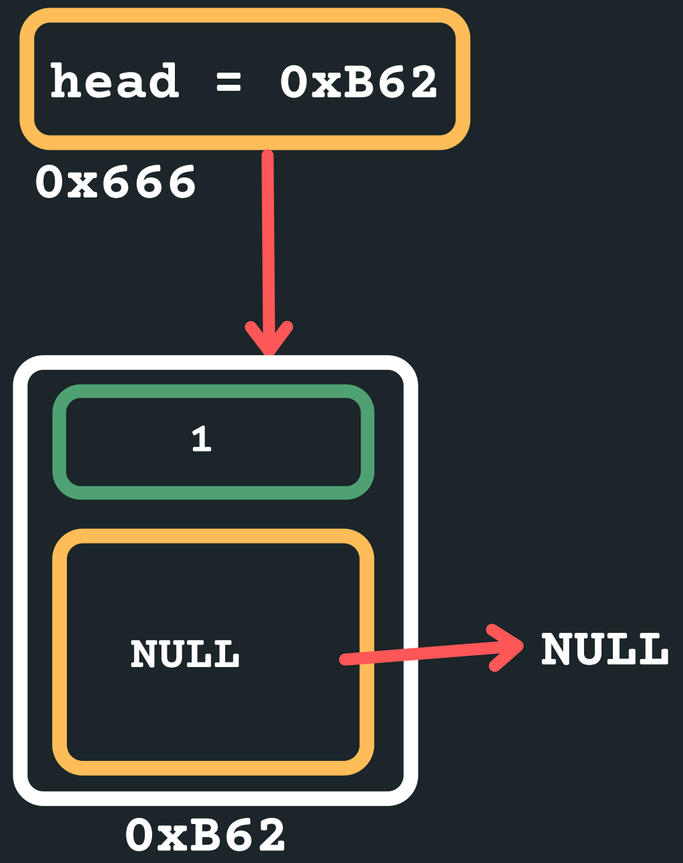
- Deleting when nowhere! (it is an empty list)
  - Check if list is empty
  - If it is - return NULL

```
struct node *current = head;
if (current == NULL){
    return NULL;
}
```

LINKED LISTS

DELETING ONE ITEM LIST

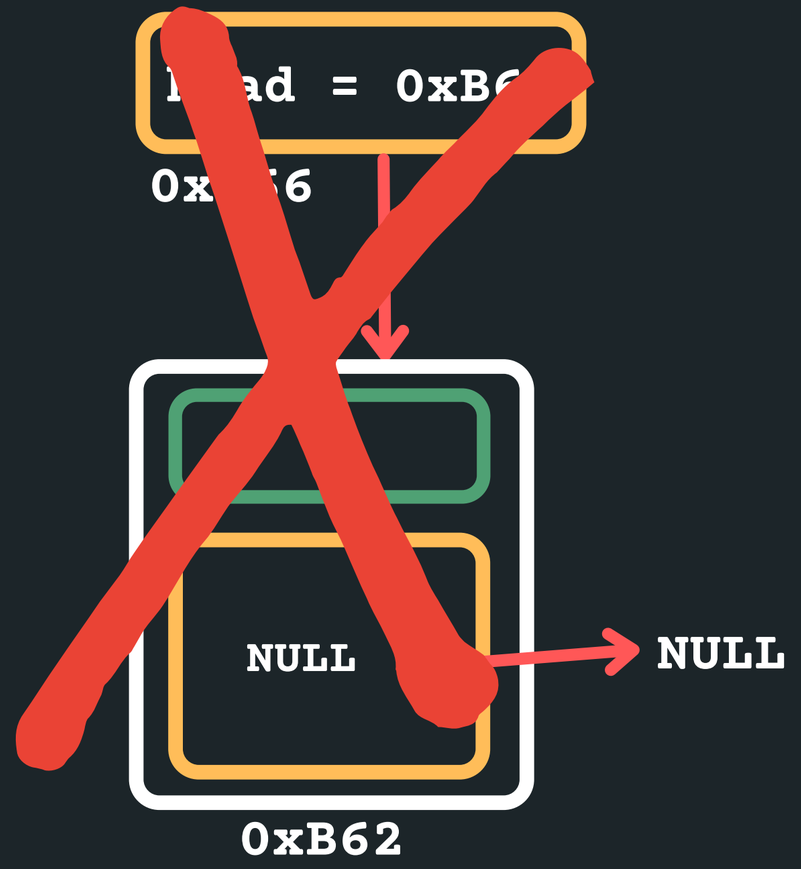
- Deleting when there is only one item in the list



LINKED LISTS

DELETING ONE ITEM LIST

- Deleting when there is only one item in the list
  - free the head!

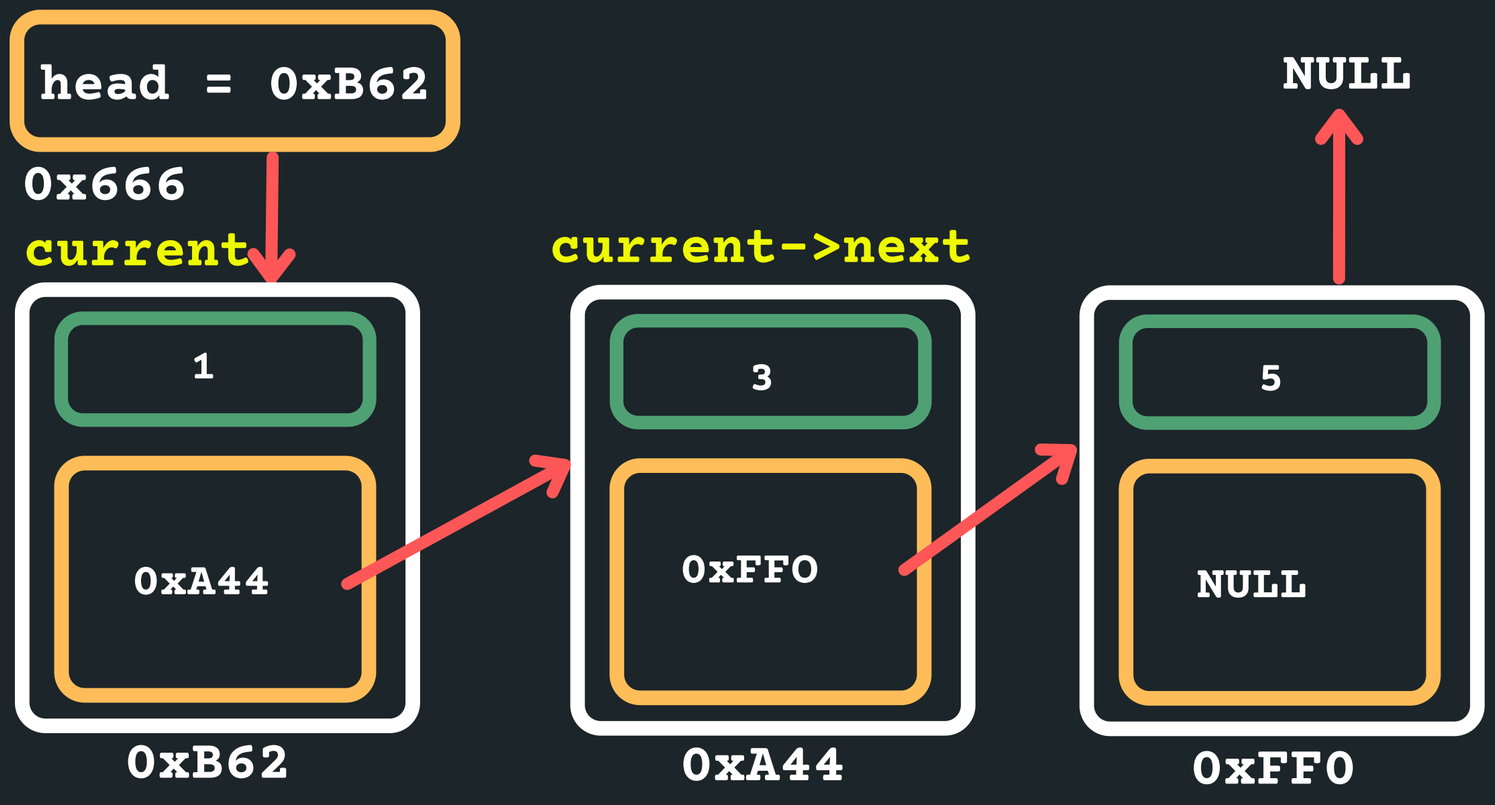


LINKED LISTS

DELETING THE HEAD WITH OTHER ITEMS

- Deleting when at the head of the list with other items in the list
  - Find the node that you want to delete (the head)

```
struct node *current = head
```

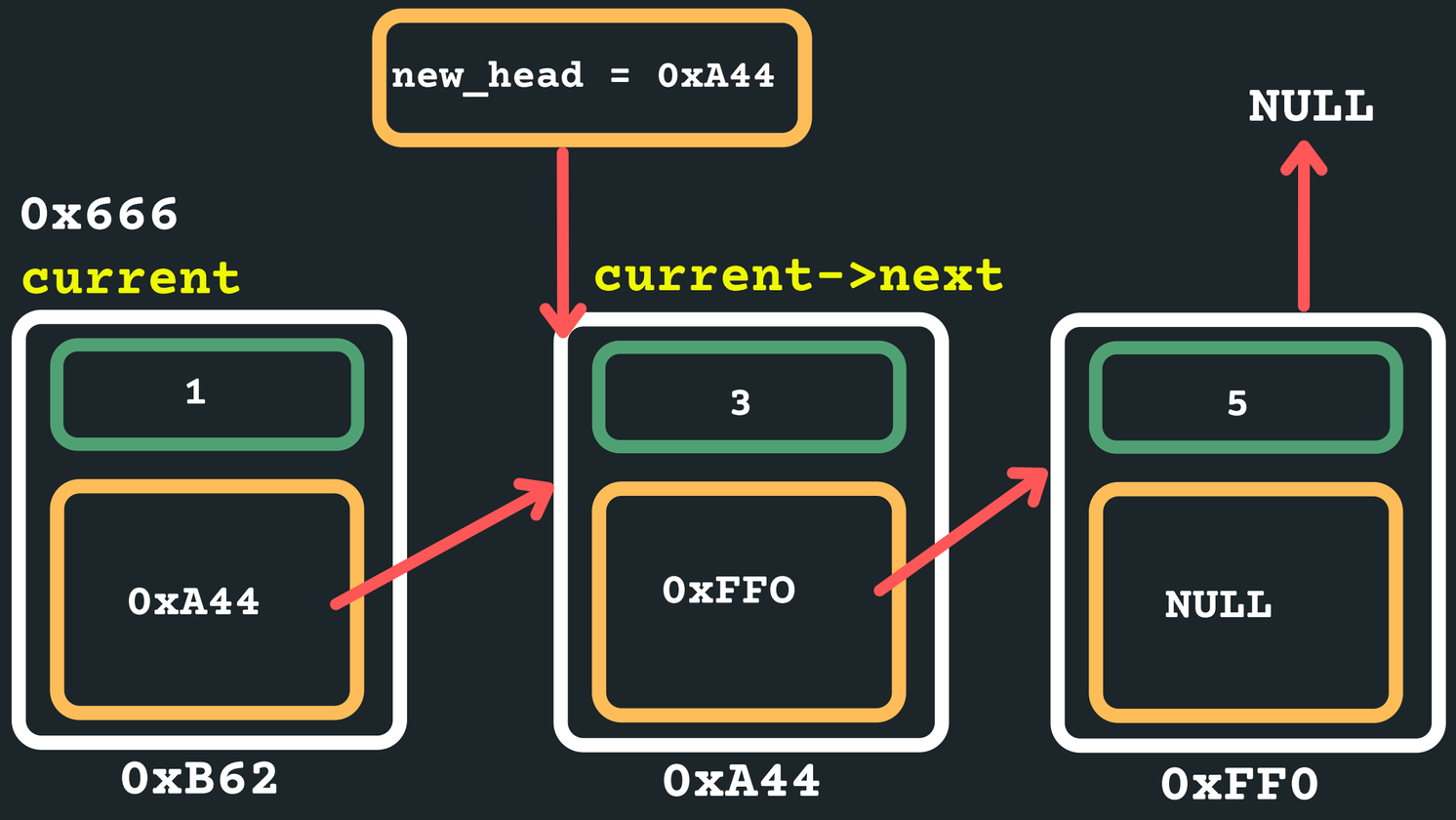


LINKED LISTS

DELETING THE HEAD WITH OTHER ITEMS

- Deleting when at the head of the list with other items in the list
  - Point the head to the next node

```
struct node *new_head = current->next;
```

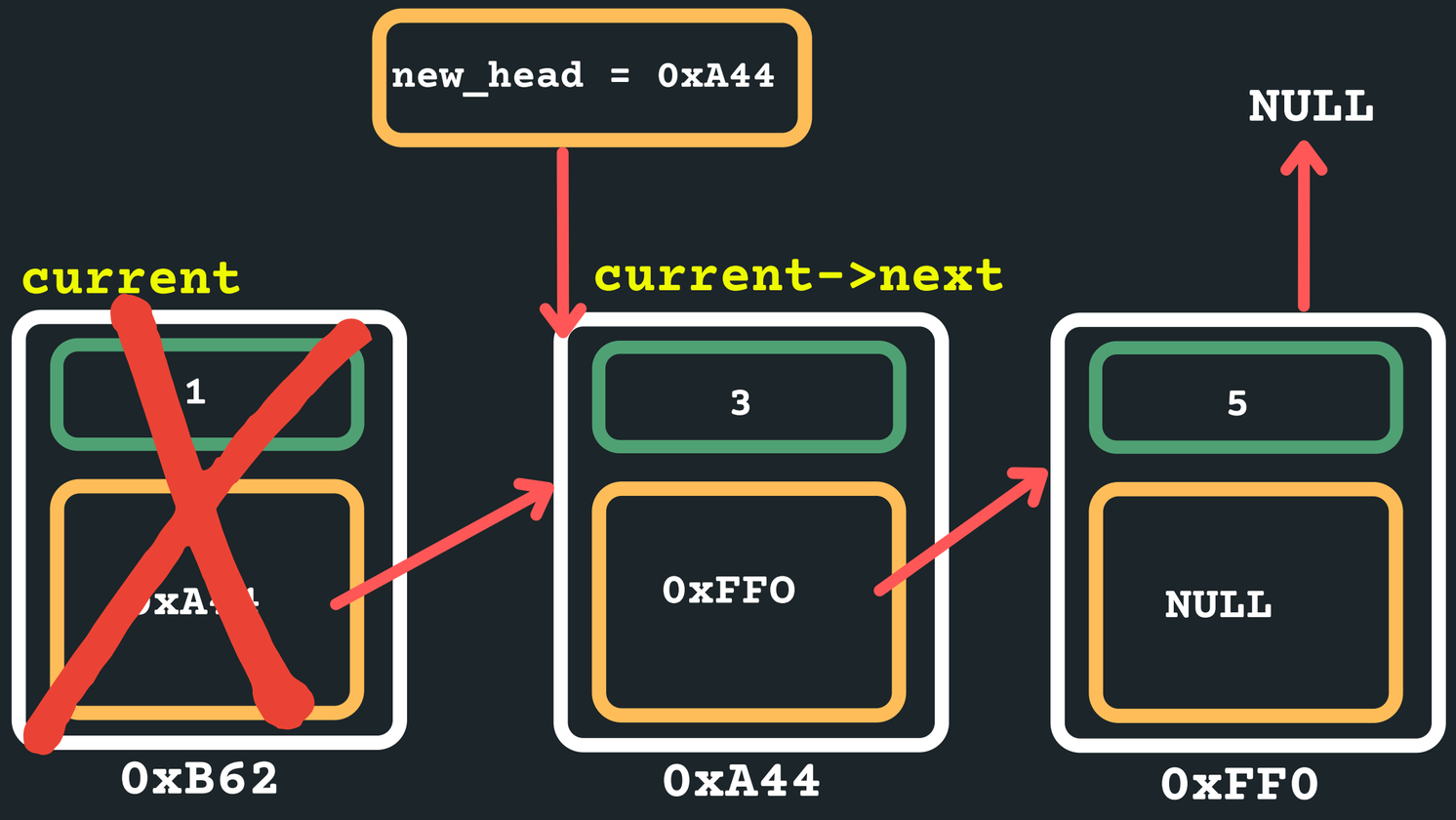


LINKED LISTS

DELETING THE HEAD WITH OTHER ITEMS

- Deleting when at the head of the list with other items in the list
    - Delete the current head

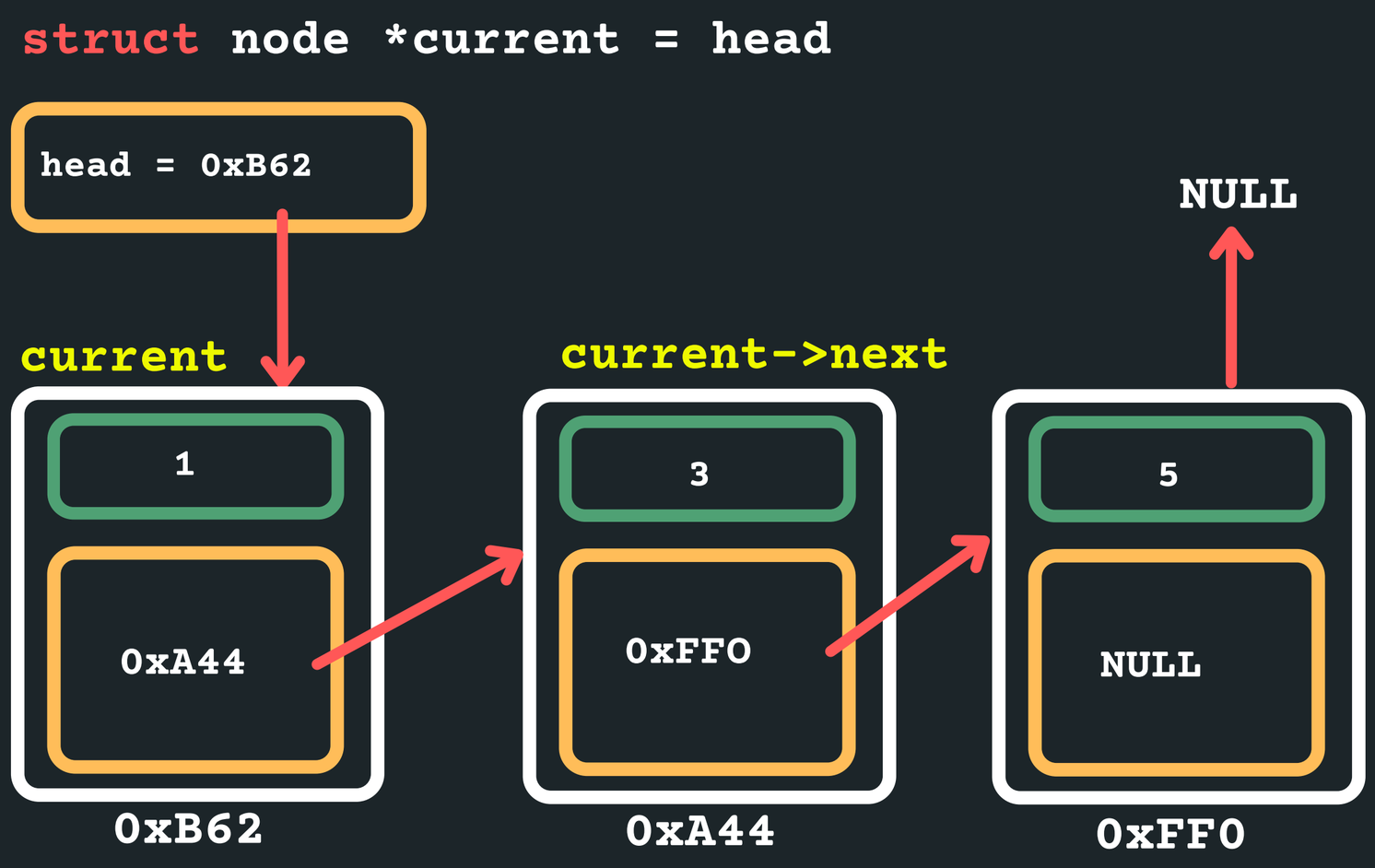
```
free(current);
```



LINKED LISTS

DELETING IN MIDDLE OF TWO NODES

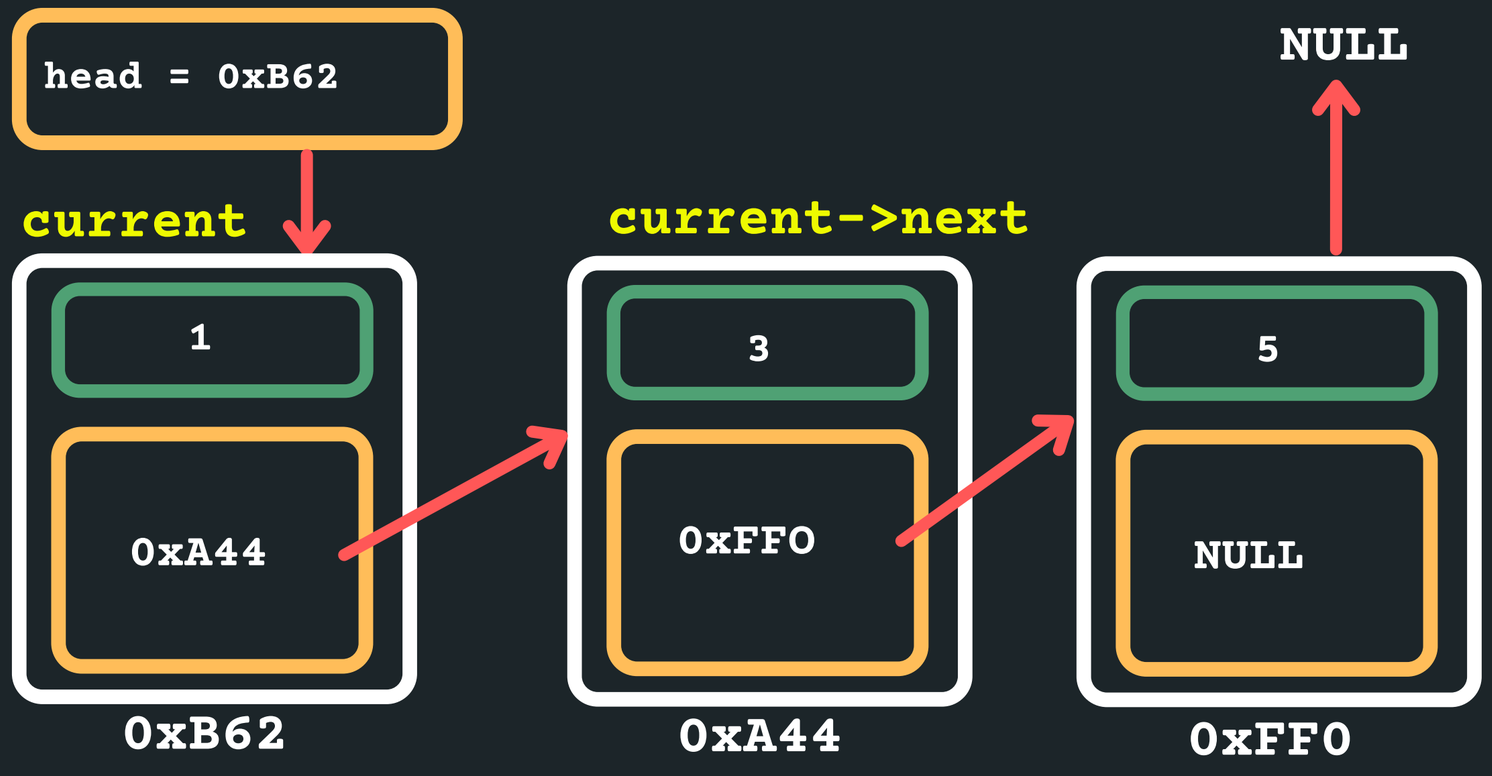
- Deleting when in the middle of two nodes (for example, node with 3)
    - Set the head to a variable current to keep track of the loop



LINKED LISTS

DELETING IN MIDDLE OF TWO NODES

- Deleting when in the middle of two nodes (for example, node with 3)
  - Loop until you find the right node - what do we think loop until the node with 3 or the previous node? Remember that onceyou are on the node with 3, you have no idea what previous node was.

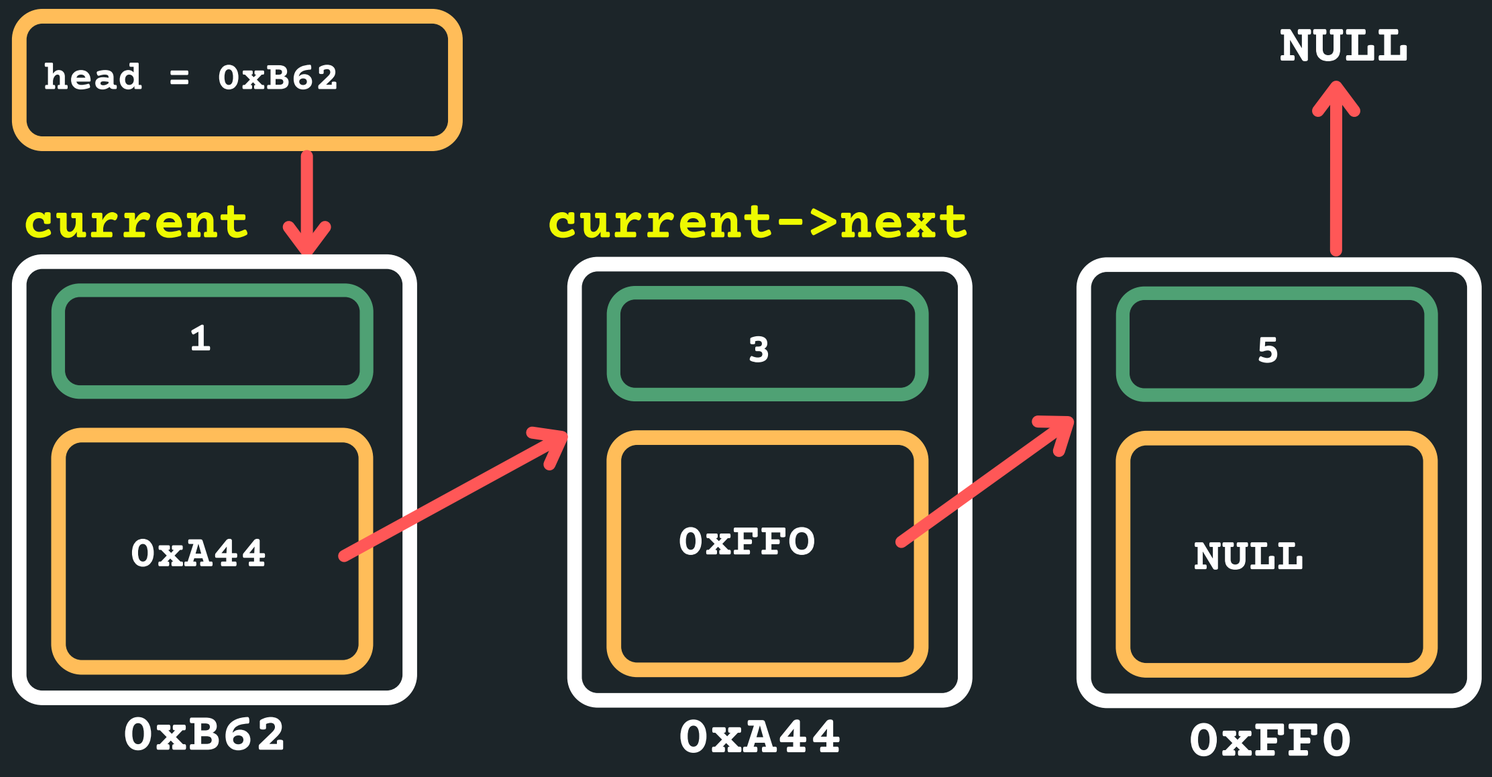


LINKED LISTS

DELETING IN MIDDLE OF TWO NODES

- Deleting when in the middle of two nodes (for example, node with 3)
    - So stop at a previous node (when the next is = 3)

```
while (current->next->data != 3){
    current = current->next;
}
```

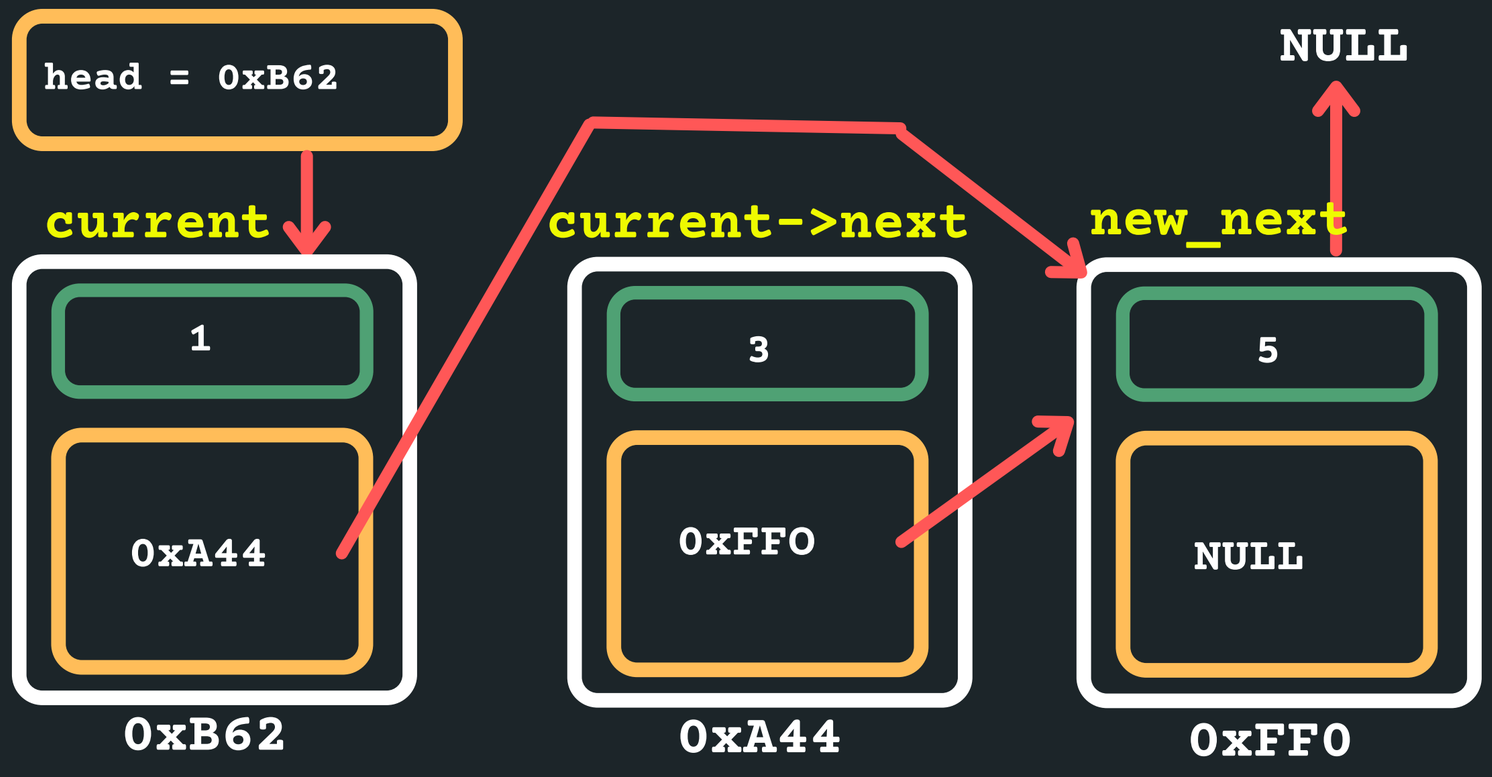


LINKED LISTS

DELETING IN MIDDLE OF TWO NODES

- Deleting when in the middle of two nodes (for example, node with 3)
    - Create new next node to store address

```
struct node *new_next = current->next->next;
```

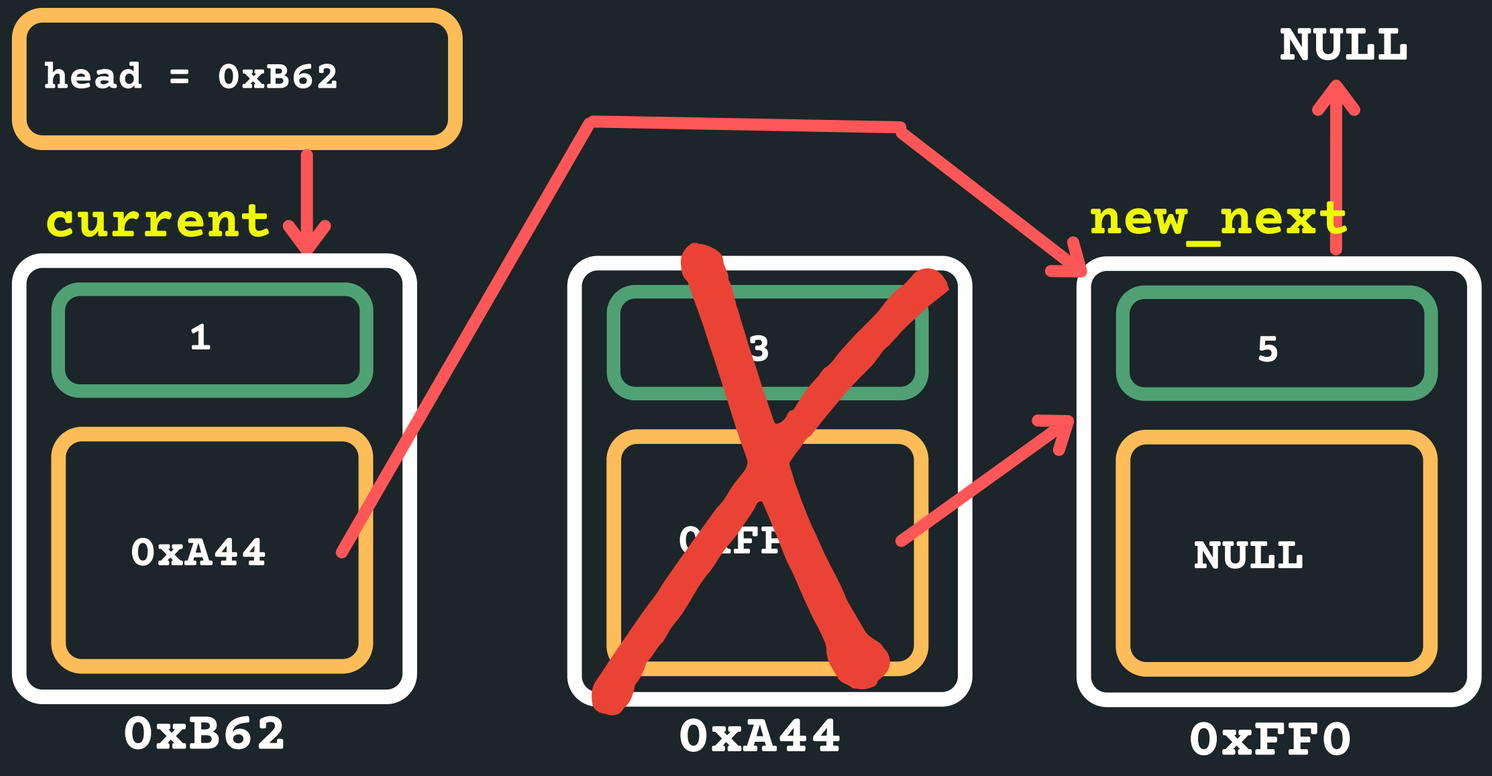


LINKED LISTS

DELETING IN MIDDLE OF TWO NODES

- Deleting when in the middle of two nodes (for example, node with 3)
    - Delete current->next

```
free(current->next);
```

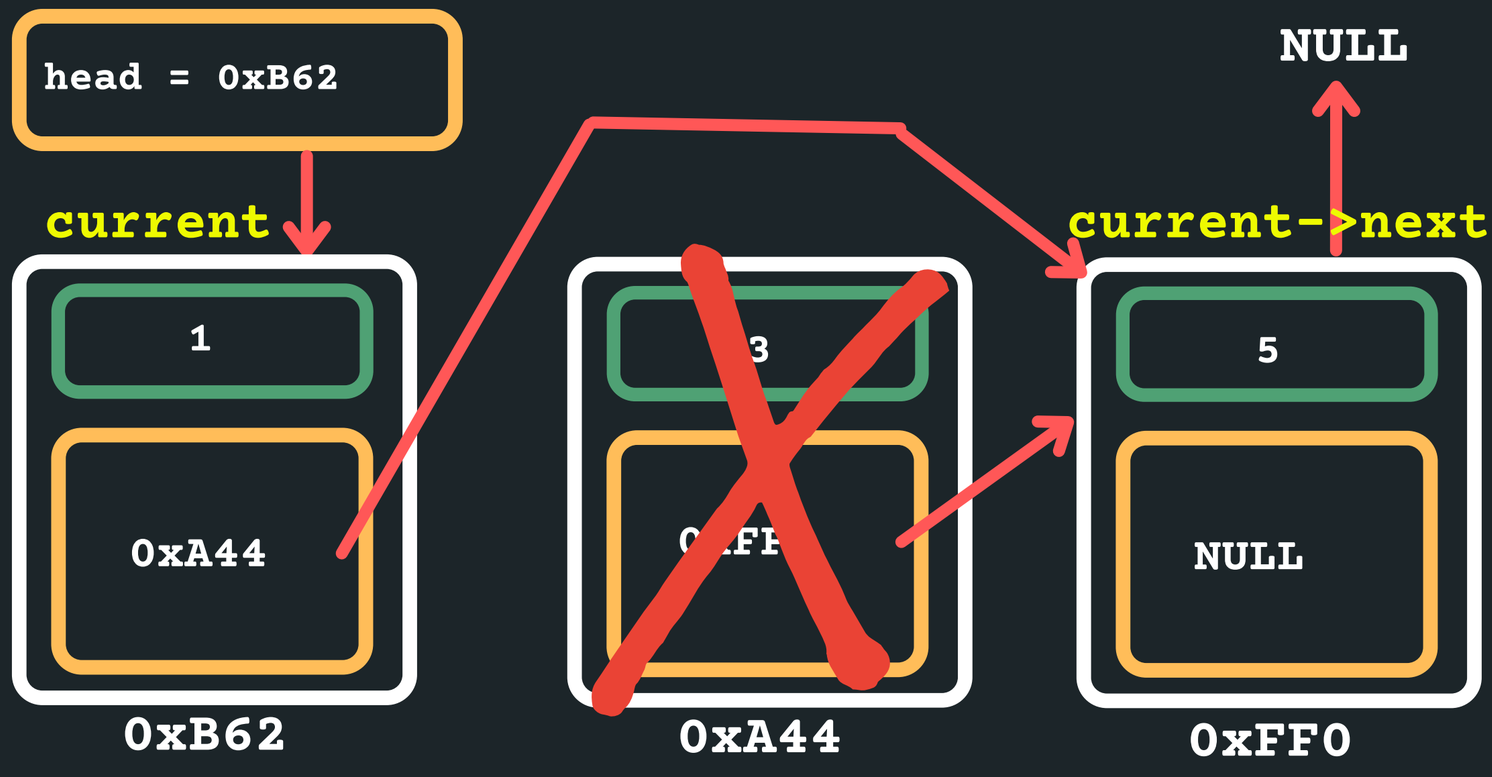


LINKED LISTS

DELETING IN MIDDLE OF TWO NODES

- Deleting when in the middle of two nodes (for example, node with 3)
    - Set the new current->next to the new_next node

```
current->next = new_next;
```

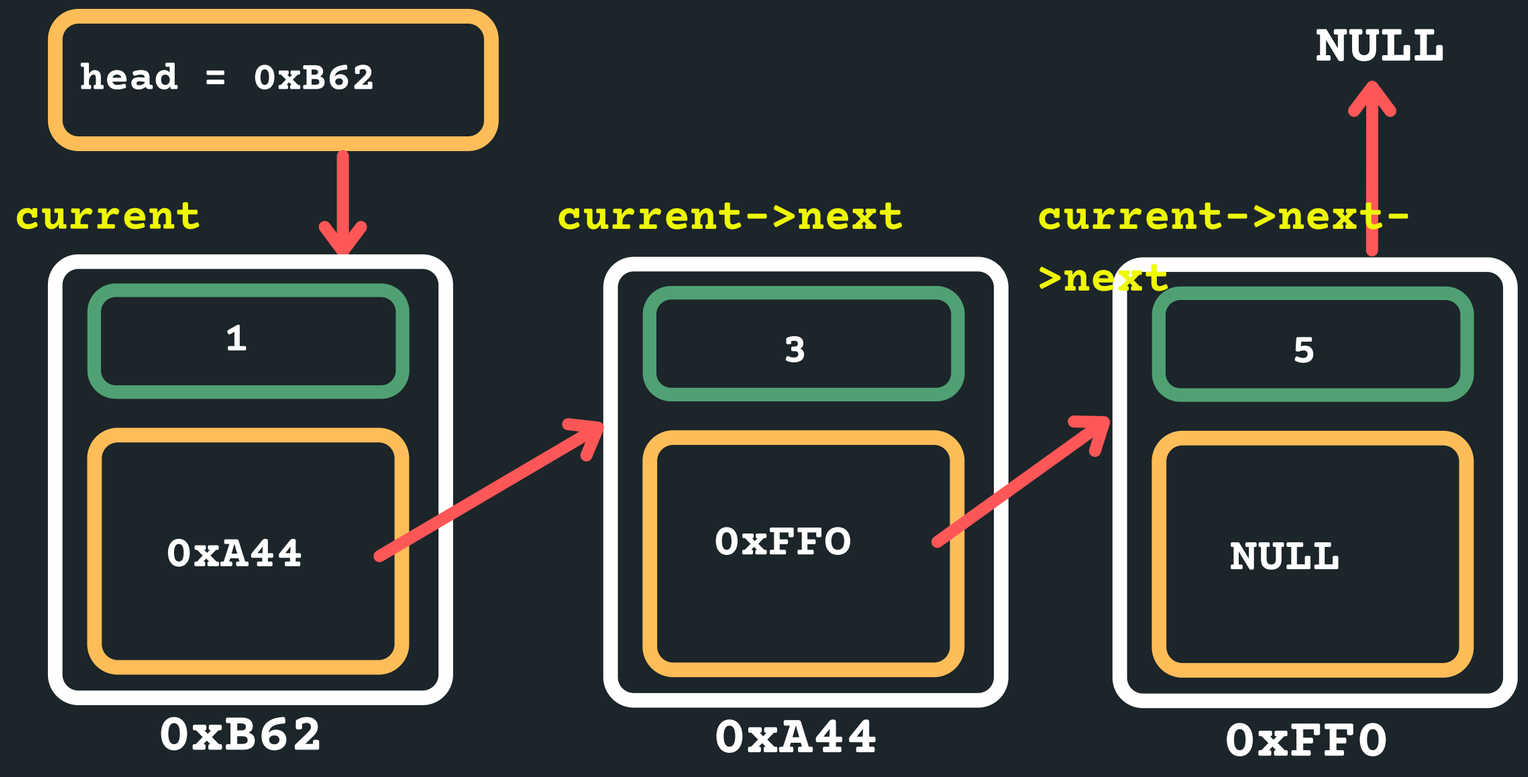


LINKED LISTS

DELETING THE TAIL

- Deleting when in the tail
  - Set the current pointer to the head of the list

```
struct node *current = head
```

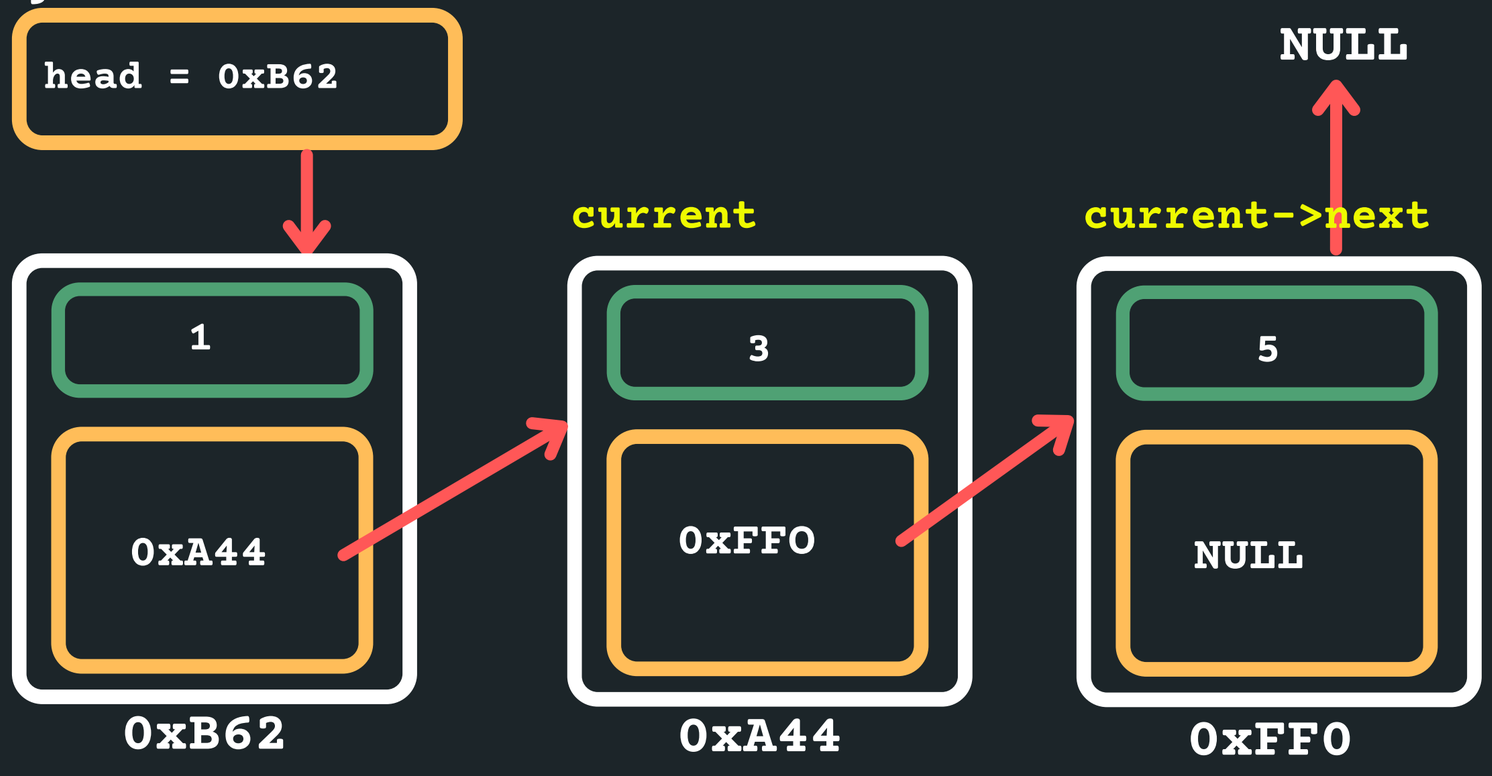


LINKED LISTS

DELETING THE TAIL

- Deleting when in the tail
  - Find the tail of the list (should I stop on the tail or before the tail?)
  - If the next is NULL than I am at the tail...

```
while (current->next->next != NULL){
    current = current->next;
}
```

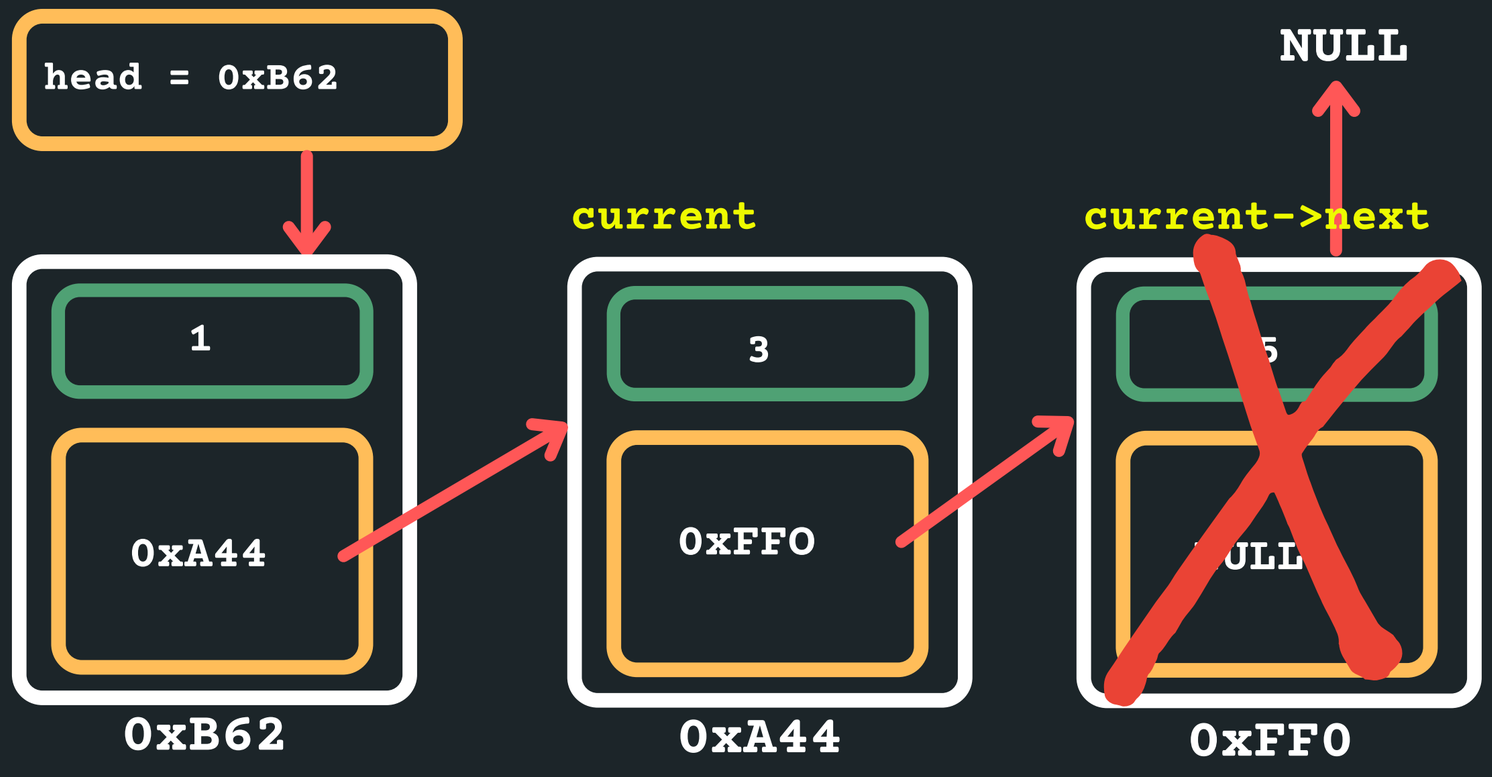


LINKED LISTS

DELETING THE TAIL

- Deleting when in the tail
  - Delete the current->next node

```
free(current->next);
```

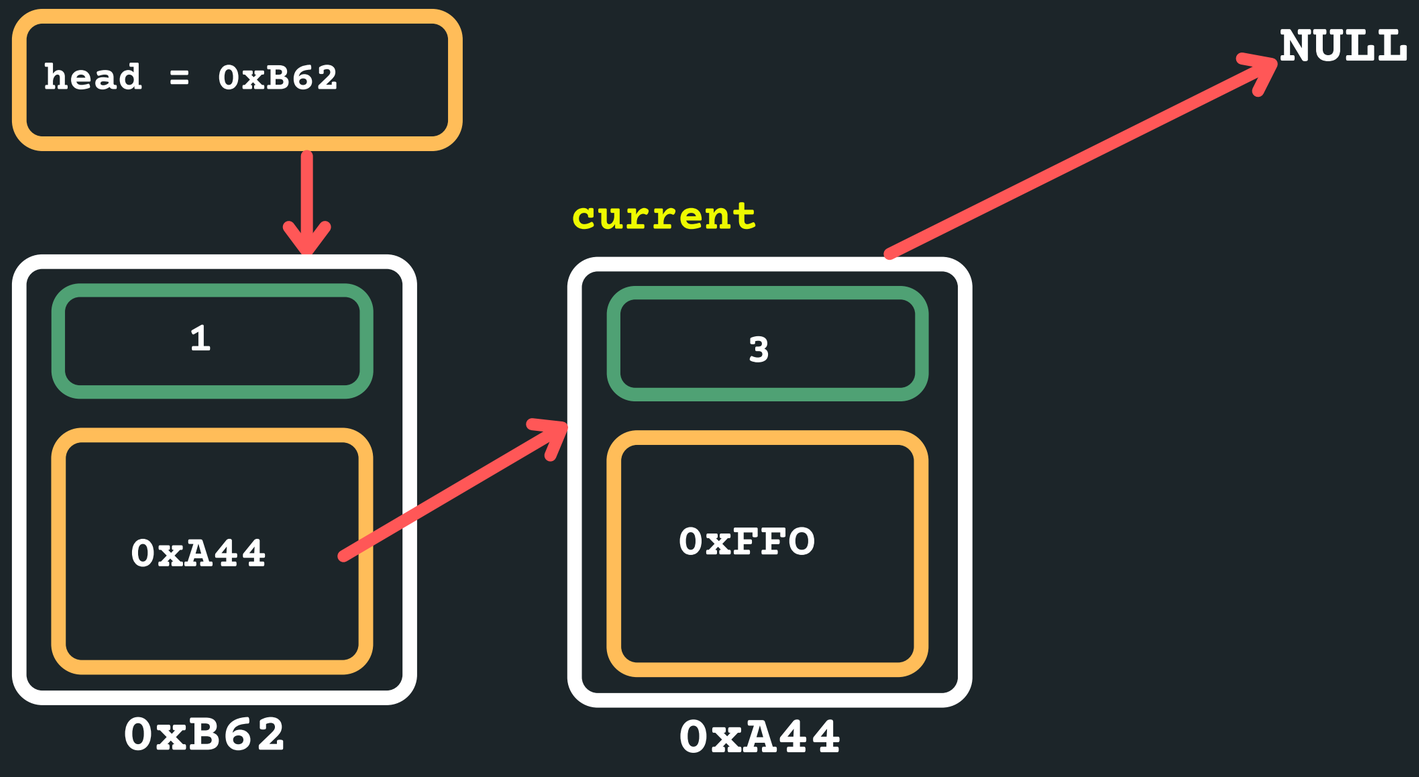


LINKED LISTS

DELETING THE TAIL

- Deleting when in the tail
  - Point my current->next node to a NULL

```
current->next = NULL;
```



LINKED LISTS

DELETING A NODE

- In all instances, we follow a similar structure of what to do when deleting a node. Please draw a diagram for yourself to really understand what you are deleting and the logic of deleting in a particular way.
- To delete a node in a linked list:
  - Find the previous node to the one that is being deleted
  - Change the next of the previous node
  - Free the node that is to be deleted

LINKED LISTS

DELETING A NODE

```c
struct node *delete_node (struct node *head, int data) {
    //create a current pointer that is set to the head of the list
    struct node *current = head;
    // if there is nothing in the list
    if (current == NULL) {
        return NULL;
    // deleting at the head of the list
    } else if (current->data == data) {
        struct node *new_head = current->next;
        free(current);
        return new_head; //will return whatever was after current as the new head
    // if there is only one node in the list and it is the one to be deleted
    // above will capture it.
    }
    //otherwise start looping through the list to find the data
    //1. find the previous node to the one you want to delete
    while (current->next->data != data && current->next->next != NULL) {
        current = current->next;
    }
    //2. if the next node is the one to be deleted
    if (current->next->data == data) {
        // create a pointer to the new next
        struct node *new_next = current->next->next;
        // 3. free the node to be deleted
        free(current->next);
        //point the next node to the new pointer
        current->next = new_next;
    }
    return head;
}
```

BREAK TIME...

Can you determine how many times do the minute and hour hands of a clock overlap in a day?

HARDER EXAMPLE

A PROBLEM

I've changed my mind on the problem, and now we are going to run an ice-cream shop...

I have decided to run an ice-cream shop (no surprises here)

I want to create a program that can:

1) Take in all the flavours of ice cream that I have to offer by adding them to the end of the list
2) I want to be able to print out the flavours
3) I would then like to add the flavours in alphabetical order
4) Delete flavours as we finish them.

Starter code is provided

Feedback please!

I value your feedback and use to pace the lectures and improve your overall learning experience. If you have any feedback from today's lecture, please follow the link below. Please remember to keep your feedback constructive, so I can action it and improve the learning experience.

https://www.menti.com/na5yrb8evn

WHAT DID WE LEARN TODAY?

LINKED LISTS - DELETING

linked_list.c

HARDER EXAMPLES

fifa.c

REACH OUT

CONTENT RELATED QUESTIONS

Check out the forum

ADMIN QUESTIONS

cs1511@cse.unsw.edu.au