comp1511 week 09

Catherine Cheng

admin

- **assignment 1 style feedback**
  - not #defining character commands
  - comments – add function comments!! also use comments within functions where appropriate
  - overdeep nesting – use more helper functions and do error checking first
- how are we going with **assignment 2?**
- **myexperience** surveys are out: https://myexperience.unsw.edu.au/

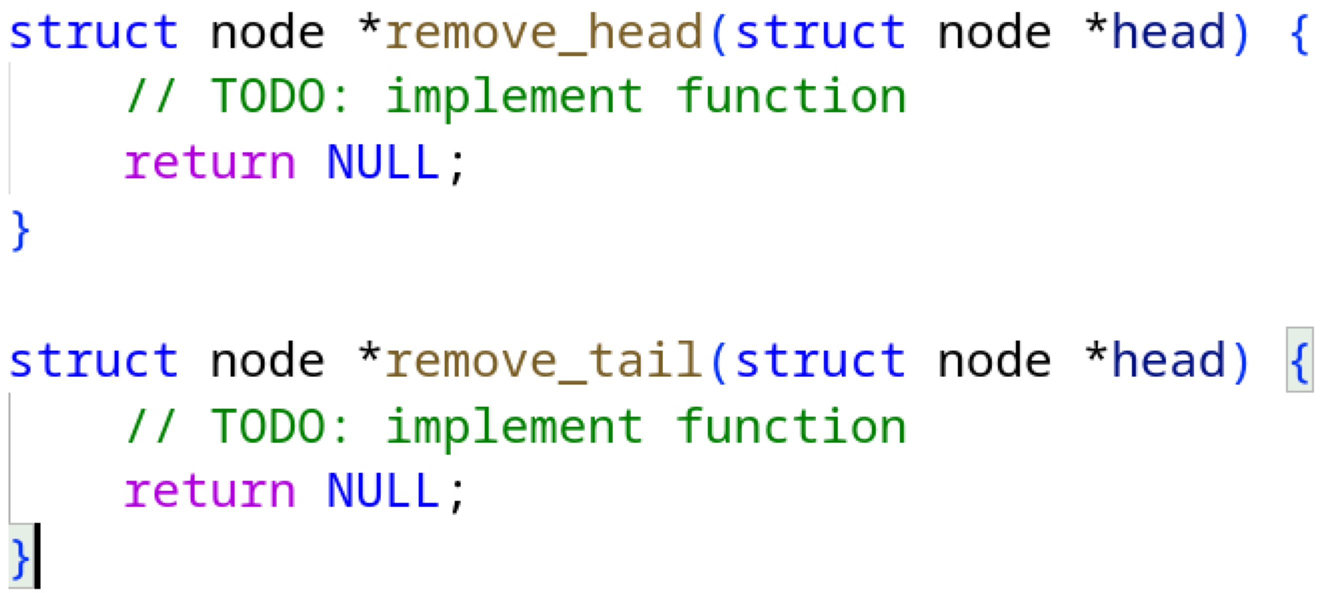
agenda for today

- free
- kahoot!
- linked list exercises

free

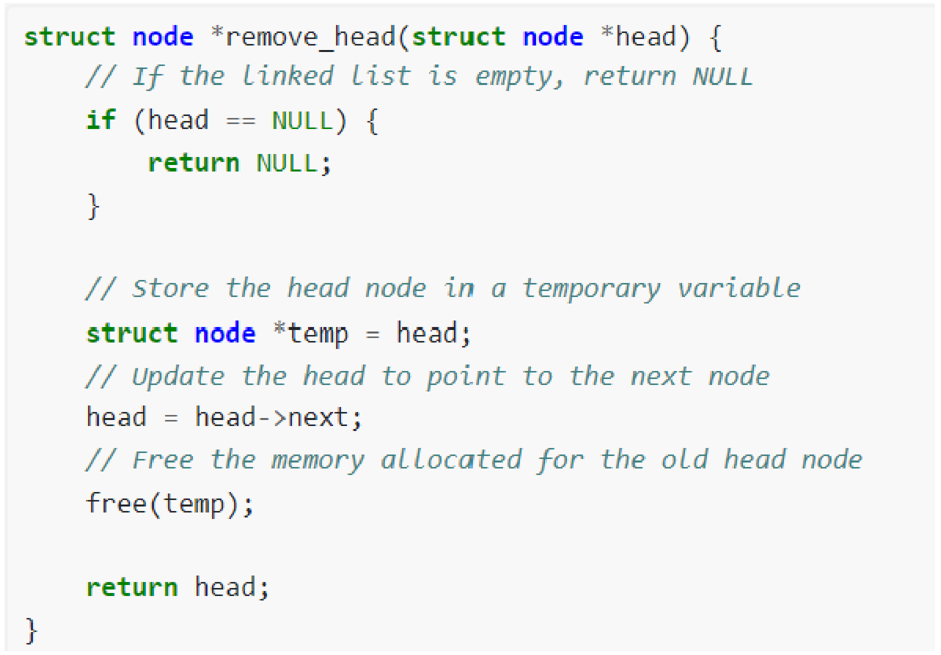
how do we free a list?

free



```c
struct node *remove_head(struct node *head) {
    // TODO: implement function
    return NULL;
}

struct node *remove_tail(struct node *head) {
    // TODO: implement function
    return NULL;
}
```

free



```c
struct node *remove_head(struct node *head) {
    // If the linked list is empty, return NULL
    if (head == NULL) {
        return NULL;
    }

    // Store the head node in a temporary variable
    struct node *temp = head;
    // Update the head to point to the next node
    head = head->next;
    // Free the memory allocated for the old head node
    free(temp);

    return head;
}
```

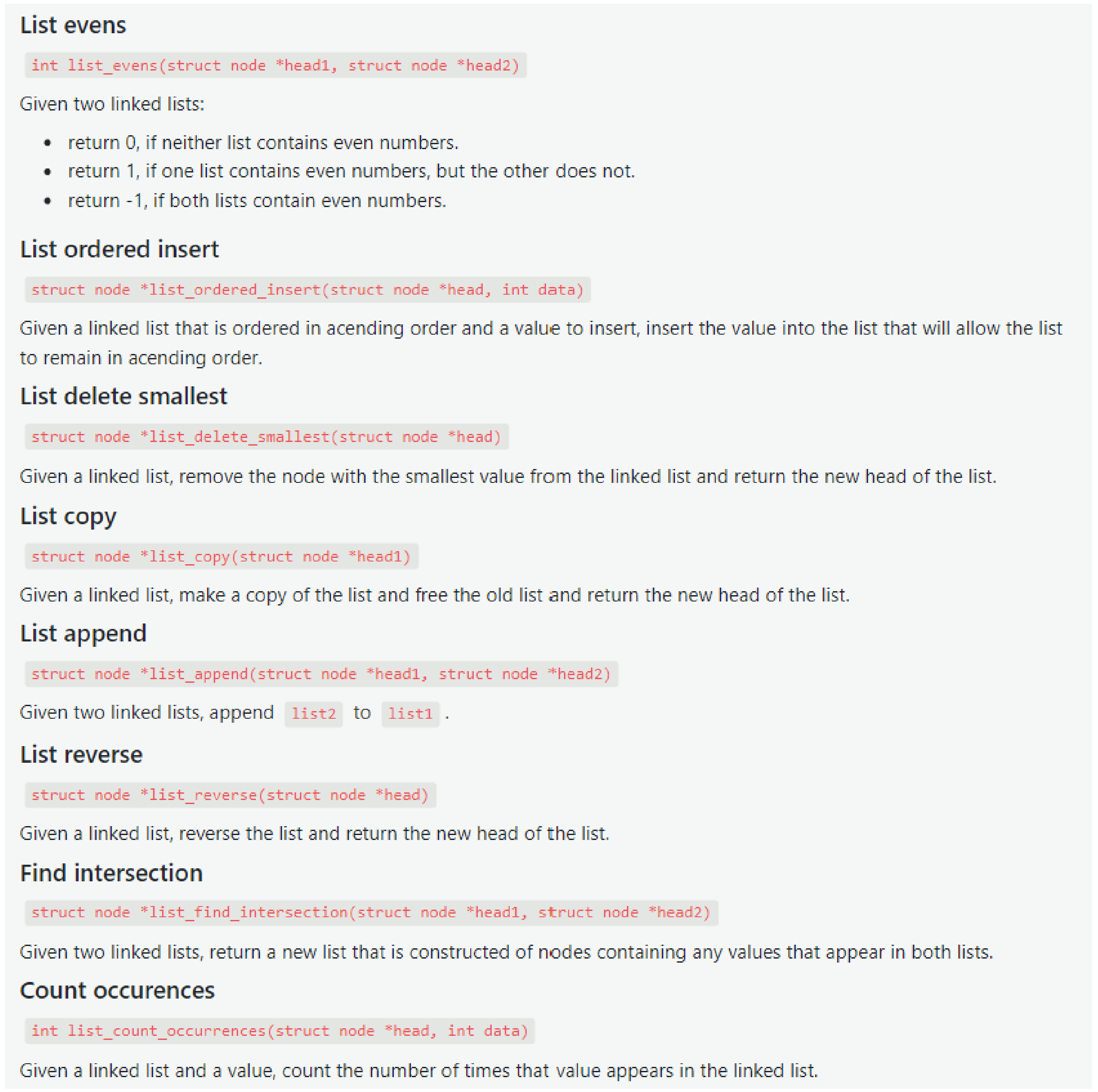
free

```c
struct node *remove_tail(struct node *head) {
    // If the linked list is empty, return NULL
    if (head == NULL) {
        return NULL;
    }

    // If the linked list has only one node, free the memory
    // allocated for the node and return NULL
    if (head->next == NULL) {
        free(head);
        return NULL;
    }

    // Traverse the linked list to find the second last node
    struct node *current = head;
    while (current->next->next != NULL) {
        current = current->next;
    }

    // Store the last node in a temporary variable
    struct node *temp = current->next;

    // Update the second last node to point to NULL
    current->next = NULL;

    // Free the memory allocated for the last node
    free(temp);

    return head;
}
```

linked list kahoot

https://create.kahoot.it/share/comp1511-tut09/0844f207-33c0-4d6d-9e3c-fdb3c5b0996b

linked list exercises



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 acending order and a value to insert, insert the value into the list that will allow the list to remain in acending 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 occurences

`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.