COMP1511 WEEK 9

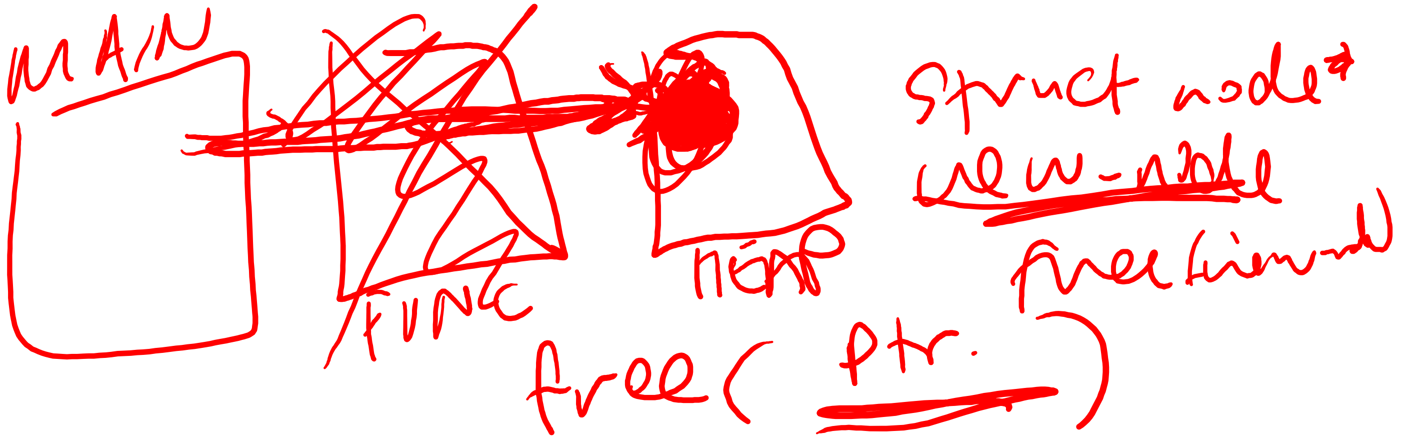
Starting at 9:08am

linked lists

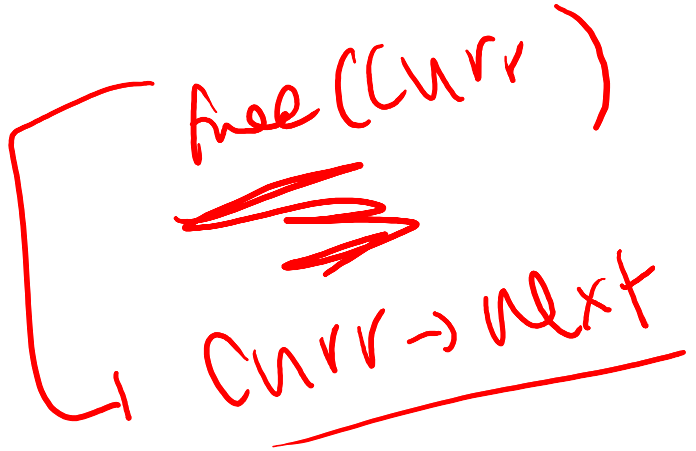
Assignment #2 Check In

- How are you going?
- Do you have any questions?
- Any tips you want to share?

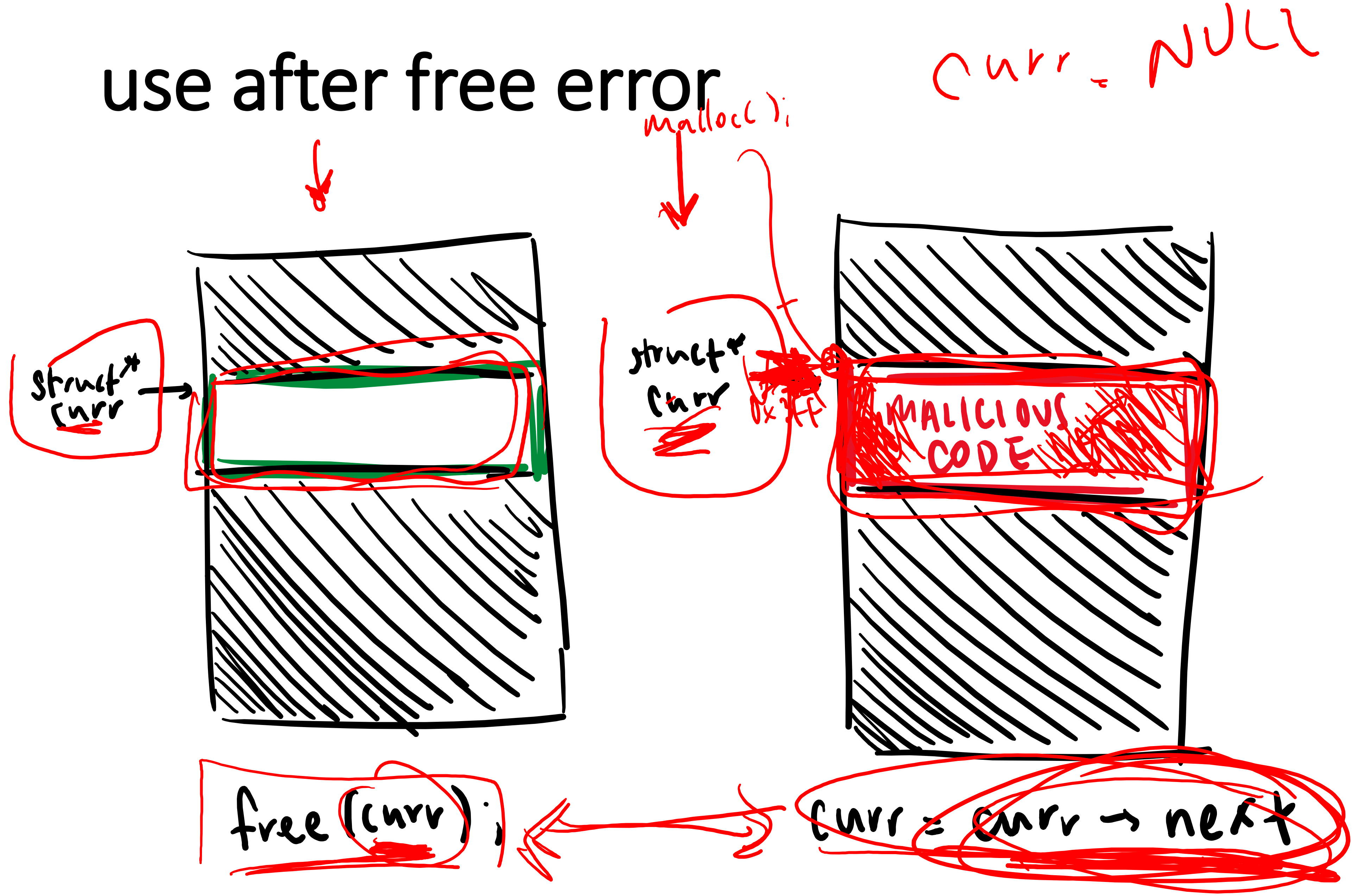
Freeing Memory



- What does free do?
  - What is the input to free and how does it help it do what it needs to do?
- What is a *use after free* error?
  - Give an example.
  - Discuss why these are extremely dangerous, one of the worst causes of bugs in C programs and a major source of security vulnerabilities.
- What is a memory leak?
  - What does **dcc--leak-check** do?

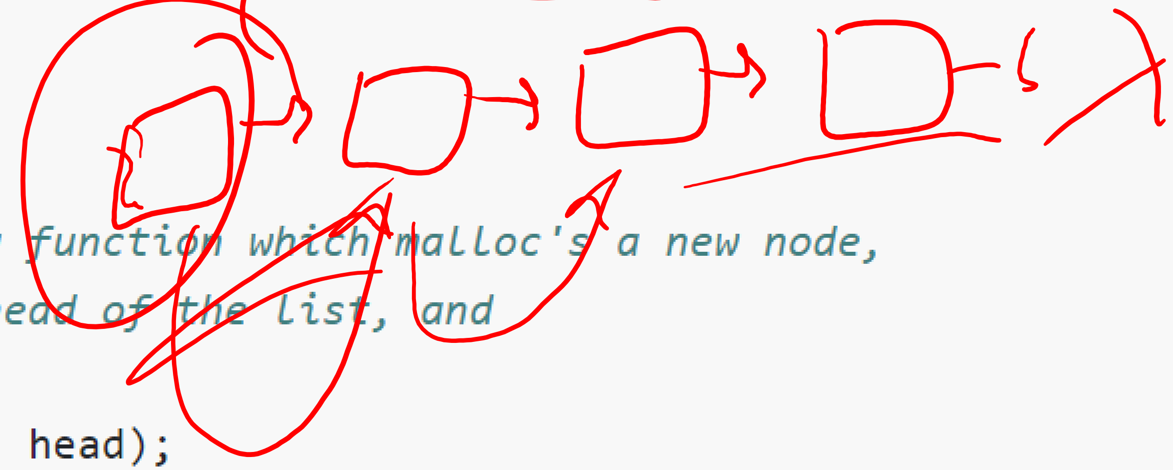


use after free error



help! I have a memory leak

8. In the following example, how much memory are we leaking? How do we stop it from leaking?



```c
int i = 0;
struct node *head = NULL;
while (i < 10) {
    // `insert_first` is a function which malloc's a new node,
    // inserts it at the head of the list, and
    // returns it.
    head = insert_first(i, head);
    i++;
}
free(head);
```

Help me debug?????

```
struct node *new_node(int data) {
    struct node *new = malloc(sizeof(struct node ~~*~~));
    new->data = data;
    new->next = NULL;
    return new;
}
```

More Linked Lists

When tackling a linked list exercise, it's a good idea to consider the following questions:

- What cases do I need to consider? Some of the common cases to consider are:
  - Number of nodes (ie empty list, list with one node, list with many nodes)
  - Location in the list (ie, at the start/middle/end of the list)
- Do I need to iterate through a linked list?
  - What loop condition(s) should I use?
  - How many iterators do I need?
- Do I need to malloc/free memory?

10. Implement a function `copy` which returns a copy of a linked list.

    `copy` should have this prototype.

    ```
    struct node *copy(struct node *old_head);
    ```

    `copy` should call `malloc` to create a new linked list of the same length and which contains the same data.

11. Implement a function `list_append` which creates a new list by appending the second list to the first.

    `list_append` should have this prototype:

    ```c
    struct node *list_append(struct node *first_list, struct node *second_list);
    ```

    Why do we need to make sure it is a new list? Why can't we just change the first list's final node's next pointer to the second list's head?

12. Implement a function `identical` that returns 1 if the contents of the two linked lists are identical (same length, same values in data fields) and otherwise returns 0.

    `identical` should have this prototype:

    ```
    int identical(struct node *first_list, struct node *second_list);
    ```

    `identical` should not create (`malloc`) any new list elements.

13. Implement a function `set_intersection` which given two linked lists in strictly increasing order returns a new linked list containing a copy of the elements found in both lists.

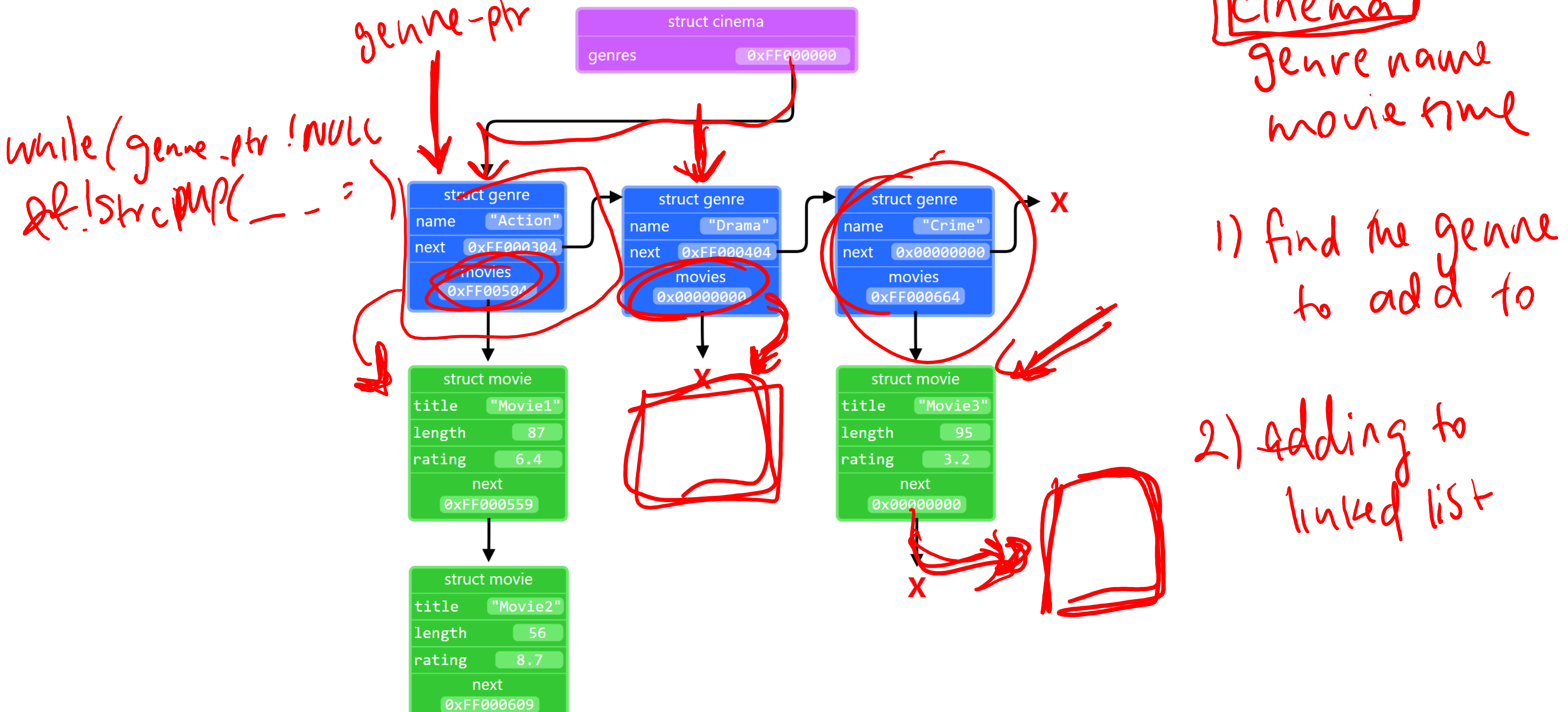
    `set_intersection` should have this prototype:

    ```c
    struct node *set_intersection(struct node *set1, struct node *set2);
    ```

    The new linked list should also be in strictly increasing order. It should include only elements found in both lists.

    `set_intersection` should call `malloc` to create the nodes of the new linked list.

1. `add_movie()` : This function should add a new `struct movie` node to the `*movies` linked list inside its corresponding `struct genre` node.
2. `print_genre()` : This function shoud print each movie associated with a given genre name. Each movie should be printed in the format: "*TITLE, RATING/100 (LENGTH minutes)*"



genre-ptr
while(genre_ptr != NULL
&& !strcmp(_ _ =
struct cinema
genres 0xFF000000
struct genre
name "Action"
next 0xFF000304
movies 0xFF00504
struct genre
name "Drama"
next 0xFF000404
movies 0x00000000
struct genre
name "Crime"
next 0x00000000
movies 0xFF000664
X
struct movie
title "Movie1"
length 87
rating 6.4
next 0xFF000559
struct movie
title "Movie2"
length 56
rating 8.7
next 0xFF000609
struct movie
title "Movie3"
length 95
rating 3.2
next 0x00000000
Cinema
genre name
movie time
1) find the genre to add to
2) adding to linked list