comp1511 week 08

Catherine Cheng

admin

- **assignment 1** marking is underway!
- **assignment 2** has been released 🥲

agenda for today

- malloc and free
- diagramming linked lists
- inserting into linked lists

malloc

what does malloc do?

what are the inputs?

what are the outputs?

malloc

Using your knowledge of `malloc()` and `sizeof()`, write up code to malloc the following:

- an integer
- a double
- a character
- an array of 10 characters
- the following struct

```
struct my_struct {
    int number;
    char letter;
    double another_number;
}
```

malloc

```c
// malloc an int
int *ptr = malloc(sizeof(int));

// malloc a double
double *ptr = malloc(sizeof(double));

// malloc a char
char *ptr = malloc(sizeof(char));

// malloc an array of 10 characters
int *ptr = malloc(10*sizeof(int));

// malloc a struct
struct my_struct *ptr = malloc(sizeof(struct my_struct));
```

free

what does free do?

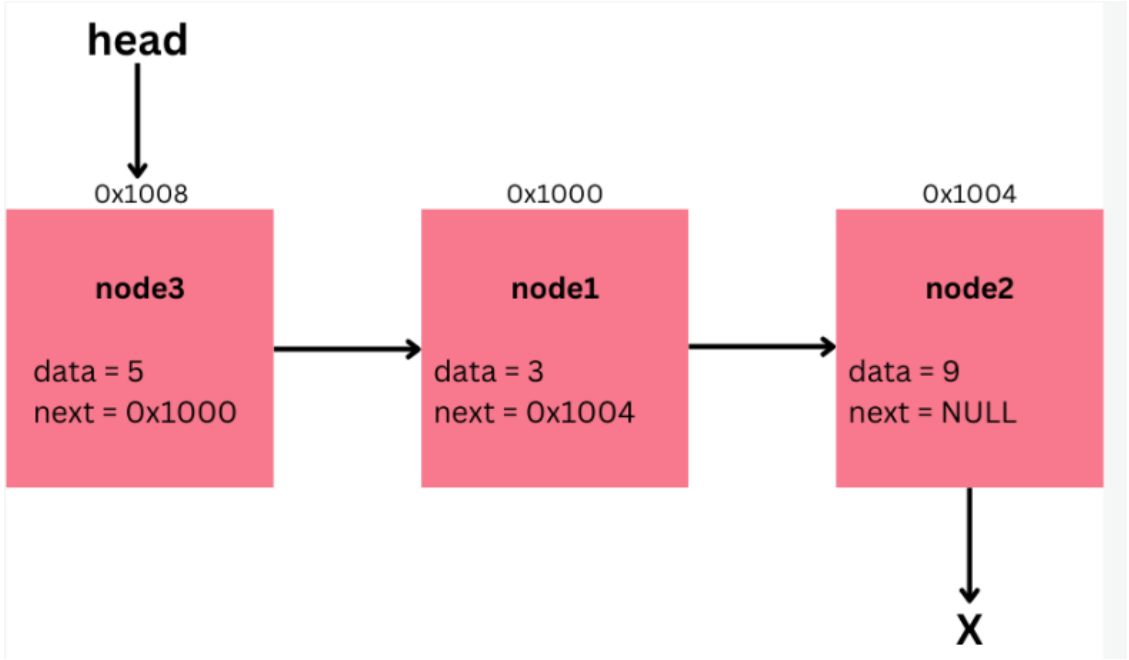
why do we need to free?

diagramming linked lists

In this activity we will run through the following instructions and build a diagram.

```
malloc memory for a new node called node1

node1 data = 3
node1 next = NULL

make the head pointer points to node1

malloc memory for a new node called node2

node2 data = 9
node2 next = NULL

add node2 to the tail of the list, making node1 next point to node2

malloc memory for a new node called node3

node3 data = 5
node3 next = NULL

add node3 to the head of the list, making node3 next point to the current head of the list
make the head point to node3
```

diagramming linked lists



linked list exercises

```c
// Creates a new node
//
// Parameters:
//      data: The data to be stored in the new node
//
// Returns:
//      A pointer to the new node
struct node *create_node(int data) {
    // TODO
    exit(1);
}

// Inserts a new node at the head of a linked list
//
// Parameters:
//      head: A pointer to the head of the linked list
//      data: The data to be stored in the new node
//
// Returns:
//      A pointer to the new head of the linked list
struct node *insert_head(struct node *head, int data) {
    // TODO
    exit(1);
}

// Inserts a new node at the tail of a linked list
//
// Parameters:
//      head: A pointer to the head of the linked list
//      data: The data to be stored in the new node
//
// Returns:
//      A pointer to the head of the linked list
struct node *insert_tail(struct node *head, int data) {
    // TODO
    exit(1);
}
```

creating node

```c
struct node *create_node(int data) {
    struct node *new_node = malloc(sizeof(struct node));
    new_node->data = data;
    new_node->next = NULL;

    return new_node;
}
```

inserting at head

```c
// Inserts a new node at the head of a linked list
//
// Parameters:
//      head: A pointer to the head of the linked list
//      data: The data to be stored in the new node
//
// Returns:
//      A pointer to the new head of the linked list
struct node *insert_head(struct node *head, int data) {
    // TODO
    exit(1);
}
```

inserting at head

```c
struct node *insert_head(struct node *head, int data) {
    // Create a new node
    struct node *new_node = create_node(data);
    new_node->next = head;

    return new_node;
}
```

inserting at tail

```c
// Inserts a new node at the tail of a linked list
//
// Parameters:
//      head: A pointer to the head of the linked list
//      data: The data to be stored in the new node
//
// Returns:
//      A pointer to the head of the linked list
struct node *insert_tail(struct node *head, int data) {
    // TODO
    exit(1);
}
```

inserting at tail

```c
struct node *insert_tail(struct node *head, int data) {
    // Create a new node
    struct node *new_node = create_node(data);

    // If the linked list is empty, return the new node
    if (head == NULL) {
        return new_node;
    }

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

    // Insert the new node at the end of the linked list
    current->next = new_node;

    return head;
}
```

inserting in between two nodes

any questions?