RECITATION 3

INFO

- Jon Rutkauskas
- Recitation: Tue 12-12:50
- Office Hours: Tue 11-11:50
  Thur 11-12:50
  SENSQ 5806
  (additional hours by appointment if needed)

- On discord: @jrutkauskas
- By email: jsr68@pitt.edu
- Ask me any questions you have!!!

POLL

- https://www.polleverywhere.com/my/polls

MALLOC AND FREE

```
#include <stdlib.h>

void* malloc(size_t size);
void free(void* ptr);
```

```
void* malloc(size_t size);
```

- Malloc is a function for **M**emory **Alloc**ation.
- You tell it how many bytes of memory you want, and it gives you a pointer to the start of that space
- All malloc'd data is stored in the heap

```
// one int is sizeof(int).
// multiply that by 10, and you have room for an array of 10 ints.
int* array = malloc(sizeof(int) * 10);

// Then, we can use it as we usually do
array[0] = 25;
array[1] = 75;
// And so on...
```

`void free(void* ptr);`

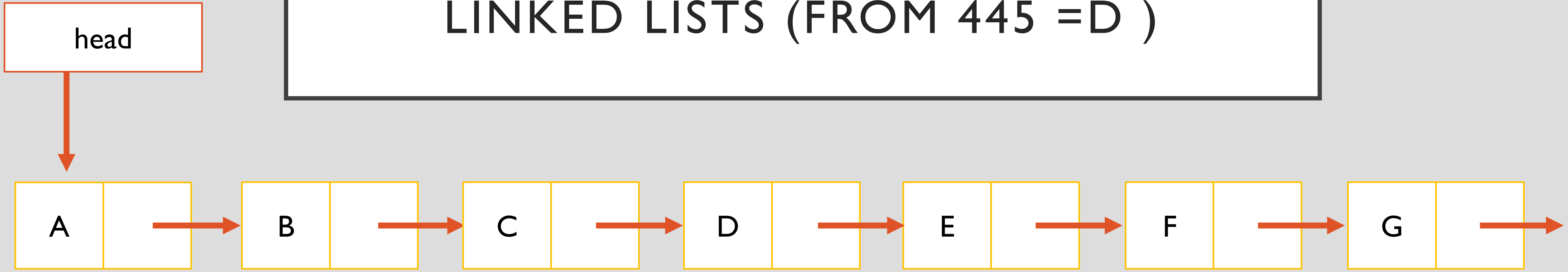
- free is a function to **free** that memory you've previously allocated
- You pass it the pointer that malloc gave you
- Once you free memory, you must never use that pointer again. (Set the pointer to null)
- Be sure you free everything you malloc after you are finished with it.
  - If you don't, there's no garbage collection, so your program will hog that memory *forever* (or until the program closes)

```
void free(void* ptr);
```

```
// Alright! I’ve used my array and now I’m done with it.  Time to free!
free(array); // it’s really that simple

// Then, we should probably make sure we never use that pointer again
array = NULL;
// This isn’t strictly required, but it makes sure that if you try to access
// it again, you’re guaranteed to get an error.  (undefined behavior otherwise)
```

LINKED LISTS (FROM 445 =D)



- Stores data in a linked chain format.
- The lab makes a singly-linked list with only a reference/pointer to the head, and stores ints. (Plan accordingly)
- Must traverse to get to any node
- Last node points to **null**
- Be careful handling nodes in the middle and end, and be sure you update the reference in the prior node
    - Since we'll be using malloc and free and don't have Java's nice garbage collection, you'll also have to be mindful to free any nodes you remove and eliminate all references to them yourself.

QUESTION TIME!

```
typedef struct Node
{
int value;
struct Node* next;
} Node;
```

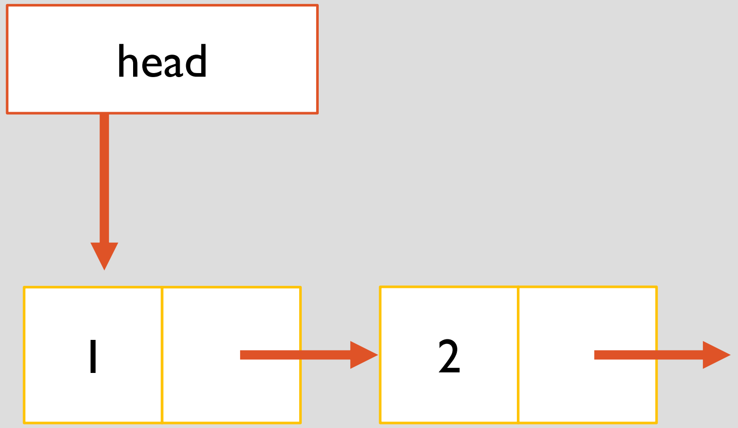
How should we dynamically allocate a new node in the heap?

HOW SHOULD WE DYNAMICALLY ALLOCATE A NEW NODE IN THE HEAP?

```
Node* n = malloc(sizeof(Node));
```

HOW TO MAKE A CHAIN OF NODES?

- How should we make a linked chain that looks like this



```
Node* head = malloc(sizeof(Node));
Node* two = malloc(sizeof(Node));
//What comes next?
```

ANSWER

```
head->value = 1;
head->next = two;

two->value = 2;
two->next = NULL;
```

OTHER QUESTIONS