RECITATION 4

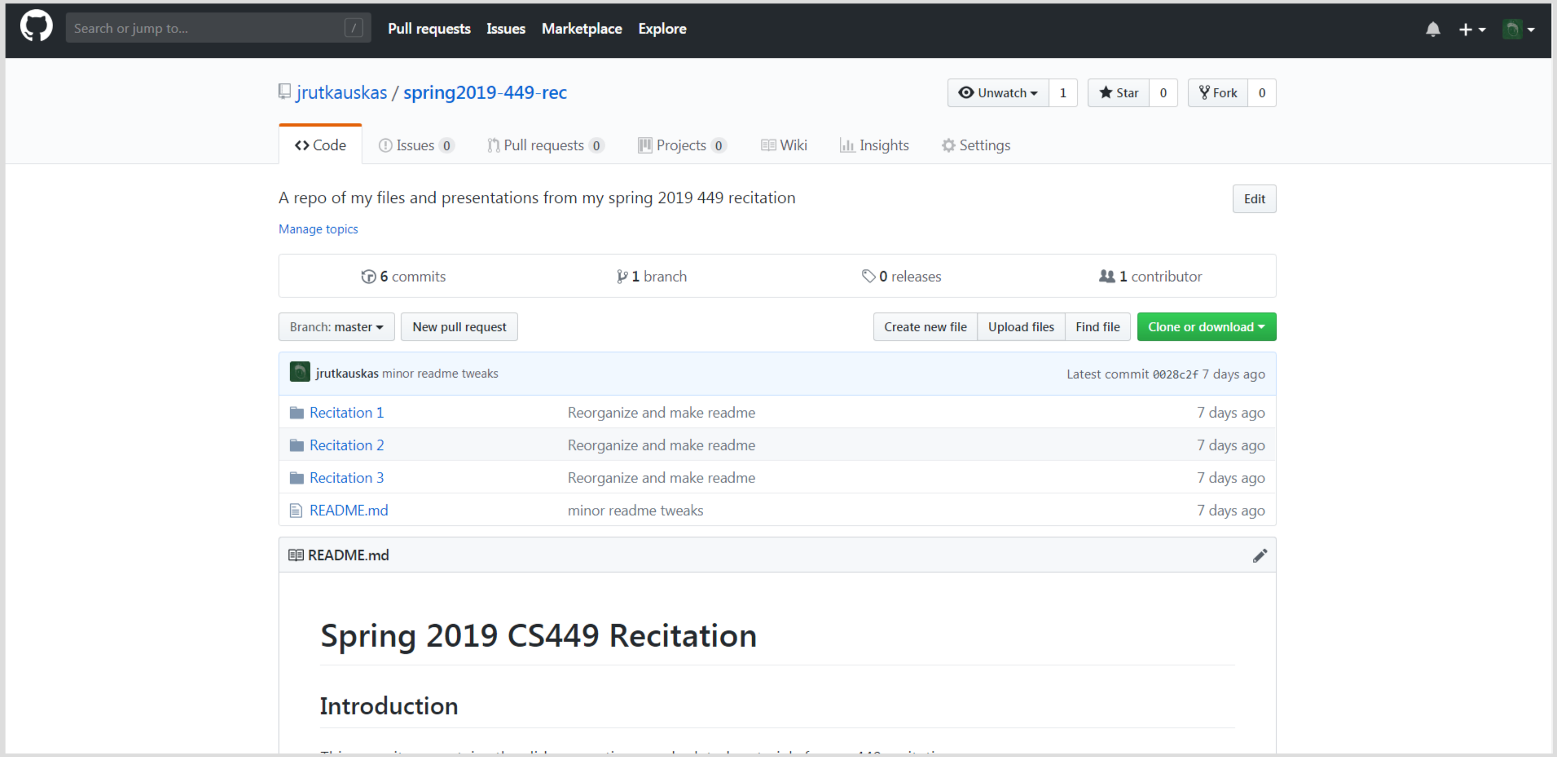
INFO

- 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
- Github:
  https://github.com/jrutkauskas/spring2019-449-rec
- Ask me any questions you have!!!

NEW: GITHUB!

https://github.com/jrutkauskas/spring2019-449-rec



A repo of my files and presentations from my spring 2019 449 recitation

6 commits | 1 branch | 0 releases | 1 contributor

jrutkauskas minor readme tweaks — Latest commit 0028c2f 7 days ago

Recitation 1	Reorganize and make readme	7 days ago
Recitation 2	Reorganize and make readme	7 days ago
Recitation 3	Reorganize and make readme	7 days ago
README.md	minor readme tweaks	7 days ago

README.md

Spring 2019 CS449 Recitation

Introduction

WARMUP POLLS

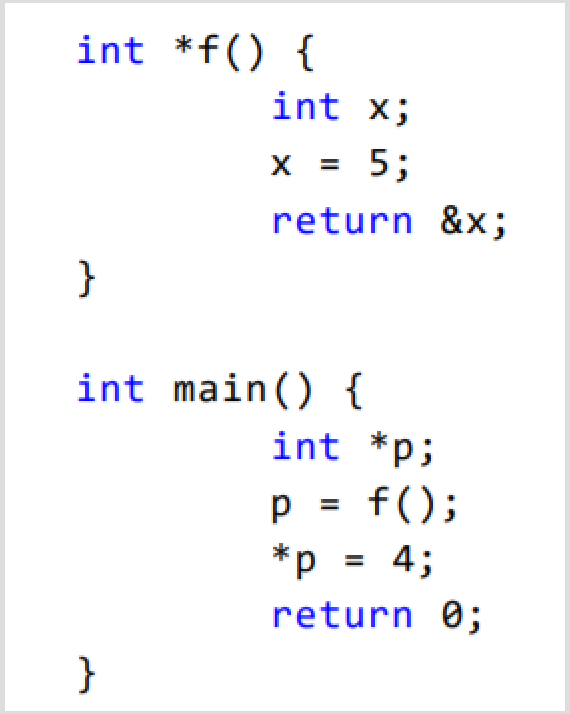
- https://www.polleverywhere.com/my/polls

IF YOU FORGET TO FREE MEMORY THAT WAS MALLOC’D, WHAT HAPPENS?

IF YOU FORGET TO FREE MEMORY THAT WAS MALLOC'D, WHAT HAPPENS?

- The memory is used until the program ends
- It doesn't matter if all pointers to it go out of scope, we don't have a Garbage Collector ☹
- Until we call free on it, it stays reserved, and takes up space on the heap.
- Once the program ends, ALL memory it used is released back to the system
  - Including memory leaks.

WHAT’S WRONG WITH THIS CODE?

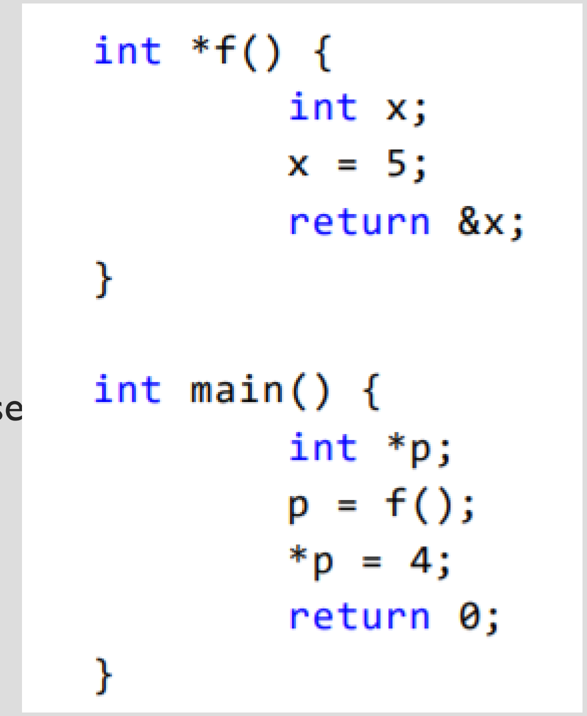


```
int *f() {
        int x;
        x = 5;
        return &x;
}

int main() {
        int *p;
        p = f();
        *p = 4;
        return 0;
}
```

WHAT'S WRONG WITH THIS CODE?

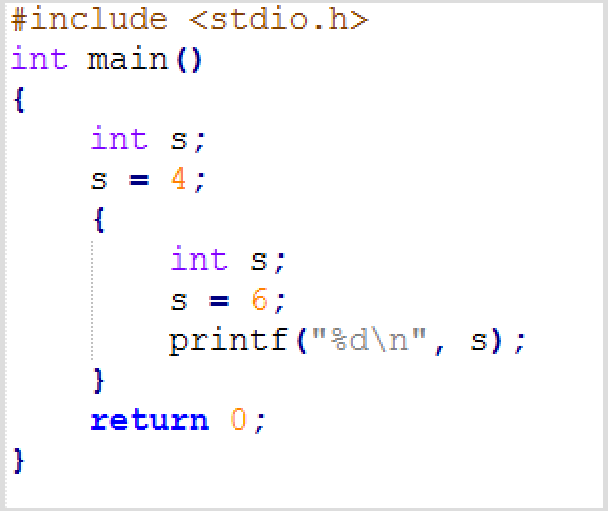
- X is created in the function f
- It exists in f's Activation Record
- When f returns, that activation record is popped off
- All of its memory is now 'inaccessible'
  - Not really, you shouldn't access it, the system could give it to someone else
  - It could be garbage, it could segfault, it could do anything… don't do it!



```
int *f() {
        int x;
        x = 5;
        return &x;
}

int main() {
        int *p;
        p = f();
        *p = 4;
        return 0;
}
```

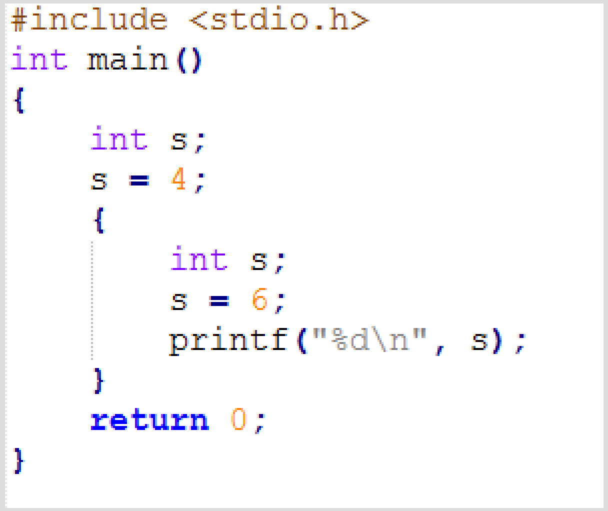
WHAT HAPPENS WHEN THIS CODE IS RUN?



```
#include <stdio.h>
int main()
{
    int s;
    s = 4;
    {
        int s;
        s = 6;
        printf("%d\n", s);
    }
    return 0;
}
```

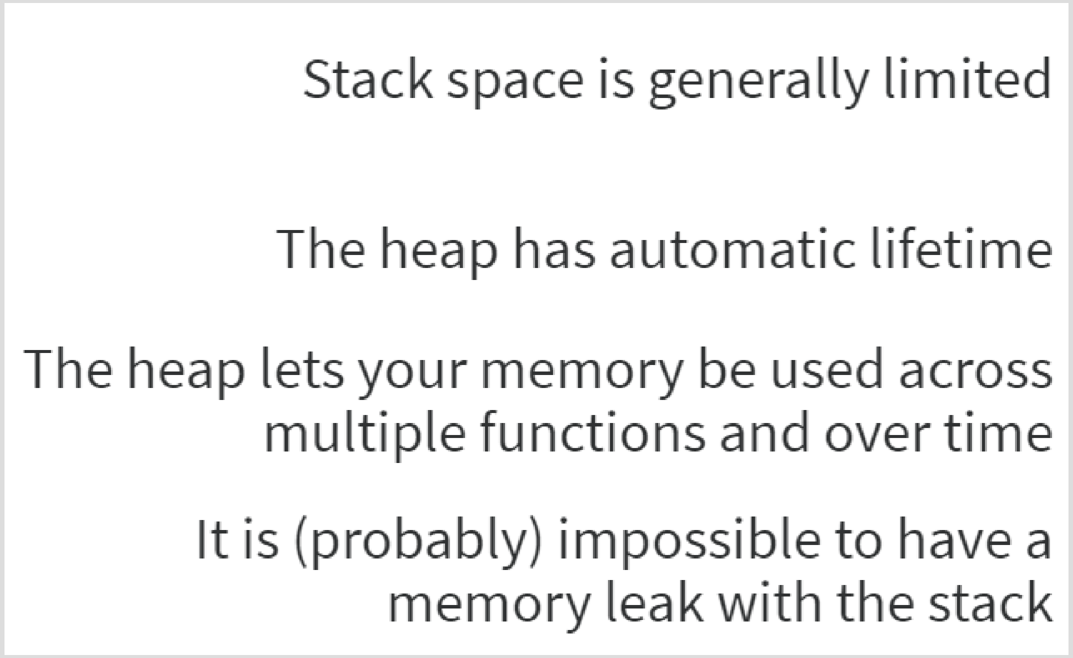
WHAT HAPPENS WHEN THIS CODE IS RUN?

- 6 is printed!
- The scope of that second ‘s’ variable is limited to the inside of those curly braces
- You set it to 6 in there
- Then it’s printed.
- This wacky nonsense is valid C… but ugly and hard to read and you shouldn’t do it



```
#include <stdio.h>
int main()
{
    int s;
    s = 4;
    {
        int s;
        s = 6;
        printf("%d\n", s);
    }
    return 0;
}
```

WHAT IS NOT TRUE OF THE HEAP?



Stack space is generally limited

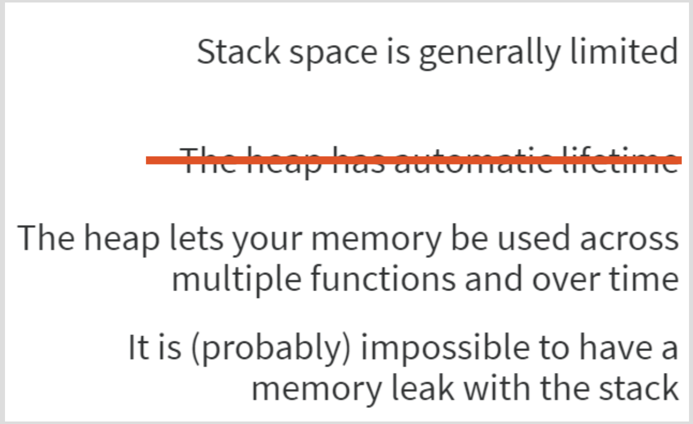
The heap has automatic lifetime

The heap lets your memory be used across multiple functions and over time

It is (probably) impossible to have a memory leak with the stack

ANSWER

- Stack space is quite limited – 10MB on thoth (FOR ALL FUNCTIONS TOGETHER)
- The heap does not have automatic lifetime. Memory allocated here must be manually free'd when you're done
- Which is useful because this memory can be used across multiple functions and all over the place until it is manually free'd. So no need to worry about if it goes out of scope as long as you don't lose your last pointer to the memory.
- Stack memory cannot leak. It DOES have automatic lifetime, and will be released at the end of the function.



Stack space is generally limited

~~The heap has automatic lifetime~~

The heap lets your memory be used across multiple functions and over time

It is (probably) impossible to have a memory leak with the stack

OTHER QUESTIONS