Little Languages

Lecture 14:

Review Game Extravaganza

Form teams of 2, 3, or 4 of you and come up with a good punny, nerdy trivia team name. Register: http://bit.ly/590-game

Form teams of 2-4 (4 max!)

- Come up with a punny, nerdy team name
  - Or something lame is fine, too
- Register your team here: http://bit.ly/590-game
- Designate a scribe with good handwriting and go ahead and fill in the answer sheets with your team name
- You have 4 minutes on this! GO!

1. Is there an error? If so, what? If not, what is the output?

```
let mut x = 0;
{
    x = 1;
    {
        let mut x = 2;
        x = 3;
        println!("{}", &x);
    }
    println!("{}", &x);
}
println!("{}", &x);
```

2. There are two errors in this code. They share the same root cause. Find them and explain them to your neighbor.

```
1  fn main() {
2      let word = String::from("hi");
3      let word3 = repeat(word, 3);
4      println!("word: {}, word3: {}", &word, &word3);
5  }
6
7  fn repeat(word: String, i: usize) -> String {
8      if i <= 0 {
9          String::from(word)
10     } else {
11         let mut repeated = repeat(word, i - 1);
12         repeated.push_str(&word);
13         repeated
14     }
15 }
```

3. Is there an error? If so, what? If not, what is the output?

```
let a = 0;
let a_mut_ptr = &mut a;
*a_mut_ptr = 1;

let mut y = 10;
let mut z = 20;
let yz_mut_ptr = &mut y;
*yz_mut_ptr = 99;
yz_mut_ptr = &mut z;
*yz_mut_ptr = 99;

println!("{} {} {}", &a, &y, &z);
```

```
1  struct Counter {
2      x: u64,
3  }
4
5  impl Counter {
6      fn from(x: u64) -> Counter {
7          Counter { x }
8      }
9
10     fn incr(&mut self) {
11         self.x += 1;
12     }
13
14     fn get(&self) -> u64 {
15         self.x
16     }
17 }
```

4. Write a few lines of code to establish, increment, and print a Counter's x value. Its final value should be 590.

```
1  struct Counter {
2      x: u64,
3  }
4
5  impl Counter {
6      fn from(x: u64) -> Counter {
7          Counter { x }
8      }
9
10     fn incr(&mut self) {
11         self.x += 1;
12         // Pause here!
13     }
14
15     fn get(&self) -> u64 {
16         self.x
17     }
18 }
19
20 fn main() {
21     let mut c = Counter::from(0);
22     c.incr();
23     println!("{}", c.get());
24 }
```

5. Draw a stack/heap diagram of the program left paused at the moment line #12 is reached.

6. Draw a stack/heap diagram of the following:

```
1 fn main() {
2     let mut a: u32 = 0;
3     let b: Box<&mut u32> = Box::new(&mut a);
4     **b += 1;
5     println!("{}", a);
6 }
```

7. How could you rewrite the declaration of res (lines 6-8) to be more concise while still calling the *bar* function?

```
1  fn main() {
2      println!("{:?}", plans(21));
3  }
4
5  fn plans(x: u32) -> Result<String, &'static str> {
6      let res = match bar(x) {
7          Ok(b) => b,
8          Err(s) => return Err(s)
9      };
10     Ok(format!("Can get in bar: {}", res))
11 }
12
13 fn bar(age: u32) -> Result<bool, &'static str> {
14     if age >= 21 {
15         Ok(true)
16     } else {
17         Err("Yikes")
18     }
19 }
```

8. There are two Strings in this example, "a" and "b". In what function frame does each's lifetime end/drop?

```
1  fn main() {
2      let s = foo(String::from("a"));
3      println!("{}", s);
4  }
5
6  fn foo(s: String) -> String {
7      bar(&s)
8  }
9
10 fn bar(s: &str) -> String {
11     println!("{}", s);
12     return String::from("b");
13 }
```