CS 551 Spring 2025 Quiz 11 KEY

April 24, 2025

Name:

Email:

Instructions: Put your name and email in the appropriate places. Answer, to the best of your ability the ***FIVE QUESTIONS*** below. Additionally, sign your name below the academic honesty statement. If you deviate from these instructions in any way, you will receive a zero on the quiz.

Unless otherwise specified, you can assume that all necessary imports have been made and there are no deliberate typos in function or type names.

Consider the following stucts, traits, enums, and functions:

```rust
#[derive(Debug, Clone)]
struct MyStruct {
    inner: Arc<Mutex<Inner>>,
}

#[derive(Debug, Clone)]
struct Inner {
    string: String,
}

impl MyStruct {
    fn new(string: String) -> Self {
        let inner = Arc::new(Mutex::new(Inner { string }));
        Self { inner }
    }
}

fn function1(my_string: &str) {}

fn function2(my_struct: &MyStruct) {
    let mut inner = my_struct.inner.lock().unwrap();
    inner.string.clear()
}

fn main() {}
```

Question 1: Will the following program compile:

```
fn question1() {
    let my_struct = MyStruct::new("question1".to_string());
    println!("{my_struct:?}");

    function2(&my_struct);

    println!("{my_struct:?}");
}
```

(circle your answer)

Yes

Reason: Borrows all around.

Question 2: Will the following program compile:

```
fn question2() {
    let my_struct = MyStruct::new("question2".to_string());

    function1(&my_struct.inner.lock().unwrap().string);

    println!("{my_struct:?}");
}
```

(circle your answer)

Yes

Reason: Still borrows, we are just having to lock the mutex ourself instead of within another function.

Question 3: Will the following program compile:

```
fn question3() {
    let my_struct = MyStruct::new("question3".to_string());

    for i in 0..3 {

        let my_struct = my_struct.clone();

        thread::spawn(move || {
            function2(&my_struct);
        });
    }
}
```

(circle your answer)

Yes

Reason: We are `clone()`ing `my_struct` in each iteration of the loop so no worries. **NB:** the loop is using variable *shadowing*. The original `my_struct` still exists.

Question 4: Will the following program compile:

```
fn question4() {
    let my_struct = MyStruct::new("question4".to_string());

    let my_inner = my_struct.inner;

    let my_other_struct = MyStruct { inner: my_inner };

    println!("{my_struct:?}");
}
```

(circle your answer)

No

Reason: When we are creating a new `MyStruct`, we are *moving* the orignal's `inner`, and thus we have a borrow of a partially moved value.

Question 5: Will the following program compile:

```
fn question5() {
    let my_struct = MyStruct::new("question5".to_string());

    let handles = (0..5)
        .map(|_| {
            let inner = my_struct.inner.clone();

            let handle = thread::spawn(move || {
                let my_struct = MyStruct { inner };

                function2(&my_struct);

                println!("{my_struct:?}")
            });

            my_struct.inner.lock().unwrap().string = "question5".to_string();

            handle
        })
        .collect::<Vec<_>>();

    for handle in handles {
        handle.join().unwrap();
    }
}
```

(circle your answer)

Yes

Reason: We are cloning the original `my_struct`'s `inner`, which is an `Arc`. While there is a cost associated here, it's relatively cheap all things considered.

Academic honesty statement: I have done this quiz completely on my own. I have not copied it from, nor have I given answers to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the quiz involved, my grade in the class will be reduced by at least one level (e.g., from A to B) for my offense, and that I will receive a grade of "F" for the course for any additional offense of any kind.