CS 551 Spring 2025 Quiz 9 KEY

April 15, 2025

Name:

Email:

Instructions: Put your name and email in the appropriate places. Answer, to the best of your ability the ***THREE 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 {
    string: String,
}

impl MyStruct {
    fn new(string: String) -> Self {
        Self { string }
    }
}

impl IntoIterator for MyStruct {
    type Item = char;
    type IntoIter = std::vec::IntoIter<Self::Item>;
    fn into_iter(self) -> Self::IntoIter {
        self.string.chars().collect::<Vec<_>>().into_iter()
    }
}

fn function1(my_string: &str) {}

fn function2(my_string: String) {}

fn main() {}
```

Question 1: Will the following program compile:

```
1  fn question1() {
2      let my_string = String::from("question1");
3      let my_struct = MyStruct::new(my_string);
4
5      function1(&my_struct.string);
6
7      println!("{my_struct:?}");
8  }
```

(circle your answer)

Yes

Reason: `function1()` only *borrows* `my_struct.string`.

Question 2: Will the following program compile:

```
1  fn question2() {
2      let my_string = String::from("question2");
3      let my_struct = MyStruct::new(my_string);
4
5      while let Some(c) = my_struct.into_iter().next() {
6          function1(&c.to_string());
7      }
8
9      function1(&my_struct.string);
10     println!("{my_struct:?}");
11 }
```

(circle your answer)

No

Reason: `into_iter()` takes *ownership* of the struct it is called on. In rust `into_*` functions take ownership of the struct they are called on by convention.

Question 3: Will the following program compile:

```
1  fn question3() {
2      let my_string = String::from("question3");
3      let my_struct = MyStruct::new(my_string);
4
5      while let Some(c) = my_struct.into_iter().clone().next() {
6          function2(c.to_string());
7      }
8  }
```

(circle your answer)

No

Reason: We are cloning the *iterator* which is returned by `into_iter()`, but, the first time through the loop, we are consuming `my_struct`, and so the second iteration of the loop will be a use after move.

Question 4: Will the following program compile:

```
1  fn question4() {
2      let my_string = String::from("question4");
3      let my_struct = MyStruct::new(my_string);
4
5      let my_string = my_struct.into_iter().collect::<String>();
6
7      let my_struct = MyStruct::new(my_string);
8
9      println!("{my_string}");
10 }
```

(circle your answer)

No

Reason: Everything is fine until line 7, where `MyStruct::new()` takes ownership of `my_string`.

Question 5: Will the following program compile:

```
1  fn question5() {
2      let my_string = String::from("question5");
3      let my_struct = MyStruct::new(my_string);
4
5      let my_string = my_struct.into_iter().collect::<String>();
6
7      function1(&my_string);
8
9      println!("{my_string}");
10 }
```

(circle your answer)

Yes

Reason: `function1()` only *borrows* `my_string`.

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.