0/6 Questions Answered
6 questions with unsaved changes

Quiz 5 - Rust

STUDENT NAME

Search students by name or email...

Q1 Scoping & Borrowing
4 Points

Q1.1 Ownership
2 Points

```
pub fn add_bye(a: &mut String) {
        a.push_str("bye");
}

pub fn add_period(mut a: String)->String {
        a.push_str(".");
        return a;
}

fn main() {
        let mut s = String::from("hi ");
        let b = add_bye(&mut s);
        let a = add_period(s);
        // HERE
}
```

Which variable owns the data originally assigned to s at HERE?

- s
- b

- [x] a
- [ ] Data is out of scope

Save Answer ***Unsaved Changes**

Q1.2 Lifetimes

2 Points

The following code takes two strings and returns the shorter of the two strings.

```
fn shortest (x:&str, y:&str) -> &str {
    if x.len() < y.len() { x } else { y }
}
```

It works fine until we have code like this:

```
fn main () {
    let x = String::from("there");
    let z;
    {
        let y = String::from("hi");
        z = shortest(&x,&y); //will be &y
    } //drop y, and thereby z

    println!("z = {}",z); //yikes!
}
```

We can help mitigate this by having the function definition to be:

```
fn shortest<'a>(x:&'a str, y:&'a str) -> &'a str {...}
```

Select the statements that are true about the updated function

definition:

- [ ] This is an example of implicit lifetimes
- [x] This is an example of explicit lifetimes
- [ ] `x` and `y` must have the same lifetime
- [x] The returned reference must have the same lifetime as the shortest living parameter
- [ ] The main code now runs with this change

Save Answer ***Unsaved Changes**

Q2 Struct, Traits, Enums

8 Points

Refer to the following enum and struct definition for the next questions.

```
#[derive(Debug)]
enum Languages {
    OCaml,
    Ruby,
    Rust
}

struct Project { name: &'static str, language: Languages, grades:

fn main () {
    const project1: &Project = &Project {
        name: _____BLANK 1_____,
        language: Languages::Rust,
        grades: &[48, 52, 0]
    };

    _____BLANK 2_____
```

```
}
```

Q2.1

1 Point

What would go in place for `BLANK 1` so that the project name is `Stark Suit Repair`?

- [ ] `String::from("Stark Suit Repair")`
- [x] `"Stark Suit Repair"`
- [ ] `&String::from("Stark Suit Repair")`
- [x] `&"Stark Suit Repair"`

Save Answer ***Unsaved Changes**

Q2.2

1 Point

Which of the following code options would print `Rust` at `BLANK 2`? Select all that apply.

- [x] `println!("{:?}", project1.language);`
- [ ] `println!("{}", project1.language);`
- [ ] `println!("{:#}", project1.language);`
- [x] `println!("{:#?}", project1.language);`
- [ ] `println!("{:?}", language);`

println!("{:?}", language);

Save Answer *Unsaved Changes

Q2.3 Traits

6 Points

Let's say we wanted to implement the `Assignment` trait to the `struct Project`. It is defined as the following:

```
trait Assignment {
    fn maxPoints(&self) -> i32;
    fn quiz() -> Project { // returns a project
      Project {
        name:"Quiz",
        language: Languages::Rust,
        grades: &[4,8,8]
      }
   }
}
```

Complete the following code so that Project is an Assignment by filling in the blanks,

```
___Blank 3 ____{
    fn maxPoints(&self) -> i32 { // sums up points
        let mut total = 0;
        for &i in self.grades {
            total += i;
        }
        return total;
    }
}

fn main () {
  let q = ___Blank 4____;
  println!("The total possible points for {} is {}.", q.name, ___
  //Should print out the sum of q's grades
}
```

Blank 3 (extending project to implement assignment)

impl Assignment for Project

Blank 4 (getting the quiz assignment)

Project::quiz()

Blank 5 (calling the method, you will not receive points for hardcoding this)

q.maxPoints()

Save Answer *Unsaved Changes

Q3

8 Points

```
fn shortest(x: &str, y: &str) -> &str {
    if x.bytes().len() < y.bytes().len() {
        x
    } else {
        y.bytes()
    }
}

fn main() {
    let hi = String::from("HelloCMSC330");
    let hello = "WinterBreakLoading";

    let l = shortest(hi, hello);

    println!("{}", hi);
}
```

There exists three errors/bugs to the code above. State 2 of the bugs and provide a fix to them.

Bug 1

shortest expects two borrowed strs, but we send them str which if done is a type error but also means the data "HelloCMSC330" would be dropped. Fix can be changing `let l = shortest(&hi, &hello);` or changing variable hi to "HelloCMSC330"

Bug 2

shortest expects explicit lifetime annotations, can be fixed with `fn shortest<'a>(x: &'a str, y:'a str) -> &'a str`

Save Answer ***Unsaved Changes**

Save All Answers Submit & View Submission >