

Question 1. True or False

Circle **T** if the statement is true, otherwise circle **F** if the statement is false.

- | | | |
|---|----------|----------|
| 1. One major disadvantage of Rust is that it automatically adds runtime type safety checks, which negatively affects its runtime performance. | T | F |
| 2. In Rust, ownership is a mechanism to check for memory leaks at runtime. | T | F |
| 3. Rust literals are first class citizens. | T | F |
| 4. The <code>as</code> operator in Python and Rust does the same thing. | T | F |
| 5. The Rust compiler will attempt to copy an object <i>only after</i> it decides that moving the object is not permissible. | T | F |

Question 2. Multiple Choices

Pick all answer(s) that are correct.

a) Which of the following keyword can be used as part of an expression in Rust?

- i. `if`
- ii. `let`
- iii. `loop`
- iv. `match`
- v. `use`

b) Which of the following types of bugs are absent from Rust programs?

- i. Data race
- ii. Null pointer exception
- iii. Divide by zero error
- iv. Dangling pointers
- v. Buffer overflow

Question 3. Short Answer

a) What is the difference between the following two sets of statements?

```
// set 1
let x = 5;
let x = x + 2;
let x = x * 3;
```

```
// set 2
let mut x = 5;
x = x + 2;
x = x * 3;
```

b) Given the following program:

```
struct Foo { name: String }
impl Foo { fn new(v: &str) -> Foo { Foo { name: v.to_string() } } }

fn foobar(n: i32, list: &Vec<Foo>) {
    for x in {0..n} {
        for (y, f) in list.iter().enumerate() {
            println!("{}", x, y, f.name); /* here */
        }
    }
}

fn main() {
    let i = 5;
    let list = vec![ Foo::new("hello"), Foo::new("new"),
                    Foo::new("world") ];
    foobar(i, &list);
}
```

Draw an ownership diagram in the form of a tree at the point when “0.2: world” is printed (the line with the comment “here”). Do **not** show borrowed objects.

Question 4. Programming Questions

a) Given the following definition for Shape:

```
enum Shape {  
    Rectangle(f64, f64), /* width, height */  
    Triangle(f64, f64),  /* height, base */  
    Circle(f64),         /* radius */  
}
```

Complete the following function, which sums the area of a list of Shapes. Assume constant π is defined.

```
fn total_area(list: &Vec<Shape>) -> f64 {
```

```
}
```

- b) Create the following function that will parse a csv file filled with integers and return a 2-dimensional vector of integers. Hint: you can parse a string to an integer using the parse method. Return an `io::Error` with `ErrorKind::Other` if a value cannot be parsed to an integer.

```
fn parse_csv(filename: &str) -> Result<Vec<Vec<isize>>, io::Error> {
```

}