ECE 421
Assignment 8

Arun Woosaree

March 30, 2020

1

a)

$$\frac{}{C + t.const\ \ c : [] -> [t]}$$

b)

$$\frac{}{C + t.add\ \ c : [tt] -> [t]}$$

c)

$$\frac{}{C + t.eq\ \ c : [tt] -> [i32]}$$

2

a)

On line 9, a local address and port is being declared. 127.0.0.1 refers to the same computer that is running the program, and 3000 is the port that the application is requesting to use. The application will listen for connections to this port.

b)

A future is a trait for types which act as a placeholder for a value that may become available in the future. It can be compared to the Promise type one might encounter in JavaScript, but it works a bit differently.

A future is a value that represents the completion of an asynchronous computation. Usually, the future completes due to an event that happens elsewhere in the system (I/O event from the operating system, a timer elapsing, receiving a message on a channel, ...).

In a language like JavaScript, you might run a callback function when the promise is resolved (this is a push model). With rust futures, it is a pull model where it relies on something else asking if it is complete or not, instead of the Future being responsible for pushing data into a callback.

c)

httpbin is a simple http request and response service. In the application example, it makes a request to the /get endpoint, which should respond with a HTTP 200 response code.

d)

On line 45, the body is a slice of bytes which httpbin returned. The serde library is used to deserialize the data to convert it to a form which rust can unserstand and interact with.

e)

The BoxFut type takes the Future type, and puts it on the heap, using a Box type.

f)

A type is Send if it is safe to send it to another thread. A type is Sync if it is safe to share between threads (&T is Send).

BoxFut should implement send because box implements send $\iff$ what it is boxing also implements send. BoxFut is wrapping something that implements both send and future, so it should implement send. It will not implement sync.

g)

Trait objects need lifetimes in a box. Since Future is a trait, BoxFut needs a lifetime.

h)

cURL is a command-line tool which can transfer data to or from a server using many protocols. (see man curl (1)) In the example, cURL makes a GET request to localhost:3000/wait, which is the server program in the code example.

hyper 0.11 or higher should be used for async IO. `https://seanmonstar.com/post/161786147642/hyper-v011`

To use async I/O, the async keyword can be used in function defnitions, and the guide below can be followed: `https://hyper.rs/guides/server/hello-world/`

3

a)

Libra is written in Rust

b)

- rust is a fast and modern language
- memory safety (as opposed to C which is prone to leaks)
- type safety

Here is a quote from the tech lead:

> Ben here, I'm the tech lead on the Calibra team. Libra started with a blank slate – all options were open. Once we made the decision to build our own blockchain infrastructure, the next question was what language to use. The team spent time considering different options, many of which OP listed. As a project where security is a primary focus, the type-safety and memory-safety of Rust were extremely appealing. Over the past year, we've found that even though Rust has a high learning curve, it's an investment that has paid off. Rust has helped us build a clean, principled blockchain implementation.
>
> Part of our decision to chose Rust was based on the incredible momentum this community has achieved. We'll need to work together on challenges like tooling, build times, and strengthening the ecosystem of 3rd-party crates needed by security-sensitive projects like ours. We're looking forward to working with the Rust community for years to come. I'd love to hear your thoughts on ways that we can work together to continue building the momentum behind Rust.

`https://www.reddit.com/r/rust/comments/c20aed/facebook_just_picked_rust_to_implement_their_new/erhsz9q/`

c)

1. lazy_static is A macro for declaring lazily evaluated statics. Using this macro, it is possible to have statics that require code to be executed at runtime in order to be initialized. This includes anything requiring heap allocations, like vectors or hash maps, as well as anything that requires function calls to be computed.
2. tokio is an event-driven, non-blocking I/O platform for writing asynchronous applications with the Rust programming language.
3. failure is re-imagined way of doing error management. It provides 2 components: Fail which is a trait for custom error types, and Error which is a struct for any type that implements Fail.

4

a)

The rust nightly channel is a release where unstable and in-development features are available. `https://doc.rust-lang.org/book/appendix-07-nightly-rust.html`

b)

Unstable features are available in the beta and nightly channels of rust. Some are only available in the nightly channel. Some may require more testing, or they're tied to unstable features in the rust compiler. `https://doc.rust-lang.org/rustdoc/unstable-features.html`

c)

The rust playground can run this code because the playground supports rust nightly. It has a react frontend, and an Iron backend. The compiler is run in docker containers, and of course, WebAssembly is involved. The code is open-source and can be viewed here: `https://github.com/integer32llc/rust-playground`.

d)

The output is `James, you are completely mad`

e)

```
mov $$1, %rax # use the write syscall
mov $$1, %rdi # write to stdout
mov $0, %rsi  # use string pointed to by msg_ptr
mov $1, %rdx  # write len characters
syscall       # make syscall
```