FACULTY OF ENGINEERING
UNIVERSITY OF ALBERTA

Assignment 8: Rust and WebAssembly

Question 1: WebAssembly is a small, well-defined language. It is specified using mainly sequent calculus. You can find the specification here:

https://webassembly.github.io/spec/core/valid/instructions.html

Given this specification and notational conventions, we define several lower-level functions in WebAssembly. Write sequent calculus definitions for the following functions;

a- "The instruction t.const results in the same generic type."

b- "The instruction t.add takes two generic values and returns the same generic type"

c- "The instruction t.eq takes two generic values and returns in an i32 value." Yes, WebAssembly implements Boolean as an Integer.

Question 2: Consider the following piece of code:

```
1    use hyper::rt::Future;
2    use hyper::service::service_fn_ok;
3    use hyper::{Body, Request, Response, Server};
7
8    fn main() {
9     let addr = ([127, 0, 0, 1], 3000).into();
10    let server = Server::bind(&addr)
11        .serve(|| {
12            service_fn(service_router)
13        })
14        .map_err(|e| eprintln!("server error: {}", e));
15
16    println!("Listening on http://{}", addr);
17    hyper::rt::run(server);
18   }
19
20   fn svc_wait(t: u64) -> impl Future<Item = (), Error = ()> {
21     println!("[start] waiting...");
22     let when = Instant::now() + Duration::from_millis(t);
23     Delay::new(when)
24        .map_err(|e| panic!("timer failed; err={:?}", e))
25        .and_then(|_| {
26            println!("[end] waiting");
27            Ok(())
28        })
29   }
30
31
32   fn fetch_data() -> impl Future<Item = future::FutureResult<RespStruct,
33        String>, Error = ()> {
34      let uri: Uri = "http://httpbin.org/get".parse().expect("Cannot parse
35      URL");
36      Client::new()
37          .get(uri)
38          // Future is polled here
39          .and_then(|res| {
40              res.into_body().concat2()
41          })
42          .map_err(|err| println!("error: {}", err))
```

```
43              .map(|body| {
44                  let decoded: RespStruct =
45                  serde_json::from_slice(&body).expect("Couldn't deserialize");
46                  future::ok(decoded)
47              })
48  }
49
50
51  type BoxFut = Box<dyn Future<Item = Response<Body>, Error = hyper::Error>
52                                                      + Send>;
53
54
55  fn service_router(req: Request<Body>) -> BoxFut {
56      let mut response = Response::new(Body::empty());
57
58      match (req.method(), req.uri().path()) {
60
61
62        (&Method::GET, "/wait") => {
63              let r = svc_wait(1500);
64              hyper::rt::spawn(r);
65              *response.body_mut() = Body::from(format!("Triggered waiting
66                  {}ms", 1500));
67          }
68
69
70          (&Method::GET, "/fetch") => {
71              let r = fetch_data().map(|x| {
72                  println!("got data: {:?}", x);
73              });
74              hyper::rt::spawn(r);
75              *response.body_mut() = Body::from("Sent request to external
76  webservice");
77          }
78
79          // ... more routers
80      }
81      eprintln!("Returning a response");
82      Box::new(future::ok(response))
83  }
```

a- Explain what do the numbers mean in line 9.

b- The function in line 20 uses **Future**; what is Future?

c- What does http://httpbin.org do (line 34)?

d- Give a definition for the **body** variable in line 45.

e- Explain the **BoxFut** type in line 51

f- Should **BoxFut** (Line 51) implement the Sync trait?

g- Should **BoxFut** (Line 51) use a lifetime?

h- At some points, you will be using the following instruction:

> *$ curl localhost:3000/wait*

   What does **curl** do?

Does this code use Async/IO, if not, how would you change the program to use it? **Question 3:**

Question 3: Libra (libra.org) is a major new product from Facebook. Libra is a cryptocurrency platform. Facebook expect to make billions from Libra and revolutionize the financial industry.

a- What language is Libra written in?
b- Discuss the technical reasons why this choice of language suits the application and its objectives.
c- Libra uses many standard packages, including lazy_static, tokio, failure, etc. Briefly, describe each of these packages.

Question 4: Consider the following program:

a- What is nighty channel in Rust (check Playground)
b- What are unstable features?
c- Why can playground run this code (think O.S.)
d- What is the output from this code?
e- Provide comments for the lines ending in #

```
#![feature(asm)]
fn main() {
    let message = String::from("James, you are completely mad\n");
    syscall(message);
}

#[cfg(target_os = "linux")]
fn syscall(message: String) {
    let msg_ptr = message.as_ptr();
    let len = message.len();
    unsafe {
        asm!("
        mov     $$1, %rax    #
        mov     $$1, %rdi    #
        mov     $0, %rsi     #
        mov     $1, %rdx     #
        syscall              #
    "
        :
        : "r"(msg_ptr), "r"(len)

        )
    }
}
```