Tipos Comportamentales y Contratos

Práctica 3: FuSe

1. Considerar la siguiente definición del tipo que permite representar polinomios lineales:

```
type poly = {
  a : float;  (* Coeficiente de x *)
  b : float;  (* Termino constante *)
}
```

y la siguiente función que calcula la raíz de un polinomio lineal $ax + b$

```
let raiz (p : poly) : float option =
  if p.a = 0.0 then
    None
  else
    Some (-. p.b /. p.a)
```

Definir las funciones `val linpoly_client : !poly.?float option→poly→unit`

`val linpoly_server : ?poly.!float option→unit`

donde `linpoly_client` utiliza el primer argumento para calcular la raiz del polinomio recibido como primer parámetro, y muestra la raiz, si existe. `linpoly_server` es una función que recibe un polinomio en el endpoint recibido como parámetro y responde con su raiz, si existe.

El siguiente código debería mostrar que la raiz del polinomio es -2

```
let _ =
  let b,a = create () in
  let _ = Thread.create linpoly_server a in
  linpoly_client b {a = 1.0; b = 2.0}
```

2. Modificar la solución anterior, de manera tal que sobre la sesión no se comunican valores de tipo `float option`. Por ejemplo, la función `linpoly_server` tiene el siguiente tipo:

```
linpoly_server : ?poly.⊕[ Raiz: !float | SinRaiz: end ] → unit
```

3. Modifique la solución anterior de manera que el cliente envíe los coeficientes del polinomio en lugar del polinomio en sí.

4. Modifique la definición del protocolo anterior para el caso en el que el cliente envía un polinomio cuadrático. Tenga en cuenta que, en este caso, el servidor puede responder con 0, 1 o 2 raíces.

5. Redefina el protocolo de modo que el cliente no envíe un polinomio directamente, sino que inicie una sesión a través de la cual el servidor recibirá los coeficientes del polinomio cuadrático.

6. Se desea implementar un servicio que calcula el mayor elemento de una secuencia de valores recibidos sobre una sesión. Por ejemplo, se espera poder escribir una función

   `max_client : S→int list→unit`

   donde `S` es un tipo sesión que permite enviar sucesivamente de una cantidad arbitraria de enteros (posiblemente vacía), y que recibe luego el máximo elemento de la secuencia desde el servidor. Notar que el servidor podría no enviar un máximo si la secuencia es vacía.

7. Implementar un sistema utilizando dos sesiones que permite a un cliente enviar mensajes codificados a un servidor, utilizando un proceso codificador. Por simplicidad, considere que el cliente utiliza una sesión para enviar números enteros a un proceso codificador. El codificador envía al servidor cada número enviado en su codificación binaria. El cliente puede enviar una secuencia de longitud arbitraria de números enteros al servidor.

8. Se desea implementar un carrito de compras online utilizando tipos sesión. El carrito de compras permite al cliente ordenar productos del cátalogo del comercio. El cliente ordena productos enviando un código de producto y la cantidad deseada de unidades. El servicio le confirma si el producto y las cantidades solicitadas están disponibles. Además, si hay unidades suficientes, el pedido se agrega al carrito. En caso contrario, el servicio le informa al cliente la cantidad de unidades disponibles. El cliente podrá solicitar en todo momento el contenido del carrito y el total. Además podrá quitar unidades y o productos del carrito. Cuando el cliente lo desee podrá finalizar la compra o abandonarla (vaciando el carrito). Al finalizar la compra, el cliente realizará el pago de la misma. Por simplicidad puede asumir que el servicio internamente verificará los datos de pago del cliente y autorizará o no la transacción (puede modelar esto generando una respuesta aleatoria). En caso de no autorizar el pago, se le permitirá al cliente reintentar todas las veces necesarias o abandonar la compra. El servidor cierra la sesión cuando el cliente realiza el pago exitosamente o cuando abandona la compra.

9. Extender la solución del ejercicio anterior suponiendo que los pagos no los procesa directamente el servicio, sino que llegado el momento de pago, el cliente interactúa directamente con un servicio de pagos (por ejemplo, la tarjeta de crédito), a quien el carrito proporciona el monto a pagar y el cliente brinda los datos del pago. El carrito debe confirmar la compra al cliente sólo cuando el servicio de pago indica que el mismo fue exitoso.