Ejercicio 1

Definir usando la notación `μ`, el tipo sesión `File` dado con las siguientes ecuaciones recursivas:

```
File = ?mode.Opened
Opened = &[read:⊕[eof:Opened, val:!string.Opened], close:end]
```

```
File = ?mode.μX.&[read:⊕[eof:X, val:!string.X], close:end]
```

Ejercicio 2

Para cada uno de los siguientes protocolos, defina el tipo sesión correspondiente.

a)

`Suma`: El cliente envía una secuencia de números enteros, que finaliza cuando envía el mensaje `fin`. Cuando el servidor recibe `fin`, responde con la suma de los elementos recibidos.

```
Suma = μX.⊕[numero:!int.X, fin:?int.end]
```

b)

`Files`: El cliente desea utilizar repetidamente un archivo. Es decir, el cliente puede abrir un archivo, leerlo hasta que decide cerrarlo. A continuación puede volver a abrir el archivo, o finalizar su utilización.

```
Files = μX.⊕[open:μY.⊕[read:?string.Y, close:X], stop:end]
```

c)

Un cliente puede enviar los coeficientes de un polinomio de grado arbitrario, y luego el servidor responde con todas las raices reales calculadas. Cada coeficiente y cada raíz se transmite en un mensaje.

```
Poly = μX.⊕[coef:!float.X, roots:μY.&[root:?float.Y, stop:end]]
```

Ejercicio 3

Para cada uno de los siguientes procesos, indicar si son bien tipados. En caso afirmativo, dar una derivación del juicio de tipado.

a)

```
Tester = ?int.!bool.end
Pserver = !(server?(x:Tester).x?(y:int).x!true.0)
```

```
                        server:[Tester], x:end, y:int completed
------------- [T-Aux]   ---------------------------------------- [T-Nil]
∅ ⊢ true:bool           server:[Tester], x:end, y:int ⊢ 0
------------------------------------------------------------------ [T-Out]
server:[Tester], x:!bool.end, y:int ⊢ x!true.0
------------------------------------------------------- [T-In]
server:[Tester], x:?int.!bool.end ⊢ x?(y:int).x!true.0
------------------------------------------------------- [T-In-Un]
server:[Tester] ⊢ server?(x:Tester).x?(y:int).x!true.0    server:[Tester] unlimited
```

```
------------------------------------------------------------------------------------ [T-Rep]
server:[Tester] ⊢ !(server?(x:Tester).x?(y:int).x!true.0)
```

b)

```
Tester = ?int.!bool.end
Pserver = !(server?(x:Tester).x?(y:int).x!true.0)
Pclient = x-!1.x-?(z:bool).0
Qclient = (νx:Tester)(server!x+.Pclient)
```

```
Pserver | Qclient | Qclient
```

```
server:[Tester], x-:end, z:bool completed

---------------------------------------- [T-Nil]

server:[Tester], x-:end, z:bool ⊢ 0
                                                                                    --------- [T-Aux]
------------------------------------------------ [T-In]                             ø ⊢ 1:int
server:[Tester], x-:?bool.end ⊢ x-?(z:bool).0
                                                        ---------------------- [T-Aux]    ----------------------
------------------------------------------------ [T-Out]
                                                        x+:Tester ⊢ x+:Tester            server:[Tester],
x-:!int.?bool.end ⊢ x-!1.x-?(z:bool).0
                                                        ------------------------------------------------------
---------------------------------- [T-Out-Un]
                           igual a la otra rama --->    server:[Tester], x+:Tester, x-:!int.?bool.end ⊢
server!x+.Pclient
                           -------------------------    ------------------------------------------------------
-------------------- [T-Res]
inciso a)                  server:[Tester] ⊢ Qclient    server:[Tester] ⊢ (νx:Tester)(server!x+.Pclient)
-------------------------    ------------------------------------------------------------------------------- [T-
Par]
server:[Tester] ⊢ Pserver    server:[Tester] ⊢ Qclient | Qclient
------------------------------------------------------------- [T-Par]
server:[Tester] ⊢ Pserver | Qclient | Qclient
```

Ejercicio 4

Dar el LTS para el proceso del ejercicio 3b).

```
Tester = ?int.!bool.end
Pserver = !(server?(x:Tester).x?(y:int).x!true.0)
Pclient = x-!1.x-?(z:bool).0
Qclient = (νx:Tester)(server!x+.Pclient)
```

Por la equivalencia estructural, podemos replicar infinitas veces a `Pserver`. En particular lo vamos a replicar 2 veces para poder luego reducir cada `Qclient`.

```
Pserver | Qclient | Qclient
≡ Pserver | (server?(x:Tester).x?(y:int).x!true.0) | (server?(x:Tester).x?(y:int).x!true.0) | Qclient | Qclient
```

Las reducciones de cada `Qclient` son iguales, así que miramos una sola.

```
(server?(x:Tester).x?(y:int).x!true.0) | (νx:Tester)(server!x+.Pclient)
≡ (server?(z:Tester).z?(y:int).z!true.0) | (νx:Tester)(server!x+.Pclient)
≡ (νx:Tester)(server?(z:Tester).z?(y:int).z!true.0 | server!x+.Pclient)
{server,−} → (νx:Tester)((z?(y:int).z!true.0){x+/z} | Pclient)
≡ (νx:Tester)(x+?(y:int).x+!true.0 | Pclient)
≡ (νx:Tester)(x+?(y:int).x+!true.0 | x−!1.x−?(z:bool).0)
≡ (νx:?int.!bool.end)(x−!1.x−?(z:bool).0 | x+?(y:int).x+!true.0)
{τ,−} → (νx:!bool.end)(x−?(z:bool).0 | (x+!true.0){1/y})
≡ (νx:!bool.end)(x−?(z:bool).0 | x+!true.0)
≡ (νx:!bool.end)(x+!true.0 | x−?(z:bool).0)
{τ,−} → (νx:end)(0 | 0{true/z})
≡ (νx:end)(0 | 0)
≡ (νx:end)(0)
≡ 0
```

Ejercicio 5

Dar una definición de `P` tal que el siguiente proceso implementa a un servidor del protocolo `Suma` definido en el ejercico 2a).

```
!(suma?(x:Suma).P)
```

Nota: En el ejercicio 2a) definimos `Suma` desde la perspectiva del cliente, así que tomamos el dual para este ejercicio.

```
Suma = μX.&[numero:?int.X, fin:!int.end]
```

```
!(suma?(x:Suma).x▷[numero:x?(n:int).suma!x.0, fin:x!42.0])
```