echo.ml

```
module S = Session.Bare

let echo_client ep x =
    let ep = S.send x ep in
    let res, ep = S.receive ep in
    S.close ep;
    res

let echo_service ep =
    let x, ep = S.receive ep in
    let ep = S.send x ep in
    S.close ep

let _ =
    let a, b = S.create () in
    let _ = Thread.create echo_service a in
    print_endline (echo_client b "Hello, world!")
```

ej1.ml

```
(*
val raiz : poly → float option
val linpoly_client : !poly.?float option → poly → unit
val linpoly_server : ?poly.!float option → unit
*)

module S = Session.Bare

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

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

let linpoly_client ep (p: poly) =
    let ep = S.send p ep in
    let r, ep = S.receive ep in

    match r with
    | Some v -> Printf.printf "%f\n" v
    | None -> Printf.printf "no hay raíz\n";

    S.close ep
```

```
let linpoly_server ep =
    let p, ep = S.receive ep in
    let ep = S.send (raiz p) ep in
    S.close ep

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