```
(* Programmare con tipi astratti: Supertipi e overriding *)

let floatToString f = "1.1";;
let pigreco=3.14;;

module type RECTANGLE =
   sig type rectangle
   val rectC: float -> float -> rectangle
   val area : rectangle -> float
   val perimeter : rectangle -> float
   val toString : rectangle -> string
end;;

module Rectangle =
  (struct
   type rectangle = {base:float; height:float}
   let rectC x y = {base=x; height= y}
   let area r = r.base*.r.height
   let perimeter r = r.base*.2. +.r.height*.2.
   let toString r = "rettangolo di base "^ (floatToString r.base) ^" e altezza "^(floatToStri
end: RECTANGLE);;

module type CIRCLE=
   sig type circle
   val circlec: float -> circle
   val area: circle -> float
   val perimeter: circle -> float
   val toString: circle -> string
end;;

module Circle =
  (struct
   type circle = {radius:float}
   let circlec x ={radius=x}
   let area r = r.radius*.r.radius *.pigreco
   let perimeter r = r.radius*.2.*.pigreco
   let toString r = "cerchio di raggio "^ (floatToString r.radius)
end: CIRCLE);;

let l=[Circle.circlec 3.4; Circle.circlec 1.2];;

let rec map f xs= match xs with
 [] -> []
|x::ys -> (f x)::(map f ys);;

let ax =map Circle.area l;;
let px =map Circle.perimeter l;;


(* Introduciamo un tipo shape ad emulate un supertipo di circle e rectangle. Ma Circle.area e
(* hanno tipi diversi e succede che non possaimo definire una funzione area che si specializz
(* o Rectangle.area a seconda del valore applicato. Confronta l'analogo in Java con i superti

type shape = Cerchio of Circle.circle | Rettangolo of Rectangle.rectangle;;
(*let area s = match s with
  Cerchio x -> Circle.area
| Rettangolo x -> Rectangle.area;;*)

(* In particolare non possiamo disporre di una funzione area da applicare alla map come sotto

let l=[Cerchio(Circle.circlec 3.4); Rettangolo(Rectangle.rectC 5.6 7.8); Cerchio(Circle.circl
```

```
(*map area l;;*)

(* Dobbiamo rinunciare ad usare l'higher order e procedere come sotto: *)

let rec aree xs= match xs with
 [] -> []
|(Cerchio x)::ys -> (Circle.area x)::(aree  ys)
|(Rettangolo x)::ys -> (Rectangle.area x)::(aree  ys);;
aree l;;
```