```
(programmare con tipi astratti: Higher Order)

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 "^(floatToStrin
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;;
ax
```