```
(* ESERCIZIO 1: SOLUZIONE POLIMORFISMO MULTIPLO*)


(*PUNTO A-B-C: IDEM COME NELLA VERSIONE A POLIMORFISMO SINGOLO*)


(*PUNTO D: DUE VARIABILI DI TIPO IN COPPIA E TRE VARIABILI DI TIPO UN TERNA*)
module type cOPPIAOfT =
 sig type ('a, 'b) coppia
  val coppiaC: 'a  -> 'b -> (('a,'b) coppia)
  val getUno: (('a,'b) coppia) -> 'a
  val getDue: (('a,'b) coppia) -> 'b
end;;

module CoppiaOfT =
 (struct
  type ('a,'b) coppia = {uno:'a;due:'b}
  let coppiaC x y = {uno=x;due=y}
  let getUno x = x.uno
  let getDue x = x.due
end:cOPPIAOfT);;

module type TERNAOfT =
 sig
  type ('a,'b,'c) terna
  val ternaC:  'a -> 'b -> 'c ->  ('a,'b,'c) terna
  val getC: ('a,'b,'c) terna  -> ('a,'b) CoppiaOfT.coppia
  val getTre:  ('a,'b,'c) terna -> 'c
end;;

module TernaOfT =
 (struct
  type ('a,'b,'c) terna = {c: (('a,'b) CoppiaOfT.coppia); tre:'c}
  let ternaC x y z = {c= (CoppiaOfT.coppiaC x y);tre=z}
  let getC t = t.c
  let getTre t = t.tre
end: TERNAOfT);;

module type TUPLEOfT =
 sig
  type ('a,'b,'c) tuple
  val tupleCC:  'a -> 'b -> 'c ->  ('a,'b,'c) tuple
  val tupleCT:  'a -> 'b -> 'c ->  ('a,'b,'c) tuple
  val isC: ('a,'b,'c) tuple  -> bool
  val isT:  ('a,'b,'c) tuple -> bool
  val getC:  ('a,'b,'c) tuple ->  (('a,'b) CoppiaOfT.coppia)
  val getT:  ('a,'b,'c) tuple ->  (('a,'b,'c) TernaOfT.terna)
end;;

module TupleOfT =
 (struct
 type ('a,'b,'c) tuple =C of (('a,'b) CoppiaOfT.coppia)| T of (('a,'b,'c) TernaOfT.terna)
 let tupleCC x y z = C(CoppiaOfT.coppiaC x y)
 let tupleCT x y z = T(TernaOfT.ternaC x y z)
 let isC x = (match x with C (y) -> true | _-> false)
 let isT x = (match x with T (y) -> true | _-> false)
 let getC x = (match x with C (y) -> y | _-> failwith ("Errore"))
 let getT x = (match x with T (y) -> y | _-> failwith ("Errore"))
end: TUPLEOfT);;
```

```
(*PUNTO E*)
let rec fun_CA f (x:('a 'b 'c) TupleOfT.tuple) =
  if (TupleOfT.isC x) then let u=TupleOfT.getC x in f (CoppiaOfT.getUno u) (CoppiaOfT.getDue
  else let d=(TupleOfT.getT x)
        in let c= TernaOfT.getC d
            in f (fun_CA f (TupleOfT.tupleCC (CoppiaOfT.getUno c)(CoppiaOfT.getDue c)))
                 (TernaOfT.getTre d);;


(*PUNTO F-G: IDEM COME NELLA VERSIONE A POLIMORFISMO SINGOLO*)
```