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

(*PUNTO A*)
type coppia = {uno:int;due:int};;
type terna = {c:coppia;tre:int};;
type tuple = C of coppia | T of terna;;
let rec fun_C f (x:tuple) =
    match x with
      C y -> f (y.uno) (y.due) | T y -> f(fun_C f (C y.c)) (y.tre)
    ;;


(*PUNTO B*)
let k1 = 3; (*un valore per k1*)
let k2 = 5; (*un valore per k2*)
let k3 = 5; (*un valore per k3*)
let c = C({uno=k1;due=k2});;
let t = T({c={uno=k1;due=k2};tre=k3});;


(*PUNTO C*)
let r2 = fun_C (fun x y -> if x<y then x else y) t;;
let r1 = fun_C (+) c;;


(*PUNTO D*)
module type cOPPIAOfT =
 sig type 'a coppia
  val coppiaC: 'a  -> 'a -> ('a coppia)
  val getUno: ('a coppia) -> 'a
  val getDue: ('a coppia) -> 'a
end;;

module CoppiaOfT =
 (struct
  type 'a coppia = {uno:'a;due:'a}
  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 terna
  val ternaC:  'a -> 'a -> 'a ->  'a terna
  val getC: 'a terna  -> 'a CoppiaOfT.coppia
  val getTre:  'a terna -> 'a
end;;

module TernaOfT =
 (struct
  type 'a terna = {c: ('a CoppiaOfT.coppia); tre:'a}
  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 tuple
  val tupleCC:  'a -> 'a ->  'a tuple
```

```
  val tupleCT:  'a -> 'a -> 'a ->  'a tuple
  val isC: 'a tuple  -> bool
  val isT:  'a tuple -> bool
  val getC:  'a tuple ->  ('a CoppiaOfT.coppia)
  val getT:  'a tuple ->  ('a TernaOfT.terna)
end;;

module TupleOfT =
 (struct
 type 'a tuple = C of ('a CoppiaOfT.coppia)| T of ('a TernaOfT.terna)
 let tupleCC x y  = 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 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*)
let cc = TupleOfT.tupleCC 3 4;;
fun_CA (+) cc;;


(*PUNTO G*)
let tt = TupleOfT.tupleCT 3 4 7;;
fun_CA (fun x y -> x * y) tt;;
```