Considerazioni nell' Higher Order nei ling. Funzionali e Applicazioni a strutture di controllo di tipo iterazione

Nei linguaggi funzionali non si può ignorare H.O.

I operatori HO importanti per strutturare MEGLIO il controllo e rendere più semplice la prog. Funzionale

Foldr (in OCAML: List.fold_right)

Foldr : ('a -> 'b -> b) -> 'a list -> 'b -> 'b

map : ('a -> 'b) -> 'a list -> 'b list

foldr può essere realizzata come operatore primitivo del linguaggio o no, in ogni caso si comporta come se fosse definita così:

```
foldr g es b = match es with
    [] -> b
  | a::as -> g a (foldr g as b)
```

map può avere parametri, o meno nel linguaggio funzionale, in più così ha il seguente comportamento:

```
map g ls = match ls with
    | [] -> []
    | a :: as -> (g a) :: (map g as)
```

le due funzioni sono molto utilizzate in combinazione per realizzare una struttura di controllo molto simile ad una iterazione

Esempio

(1) Somma di una lista di interi

```
sum : int list -> int
sum ls = fold (+) ls 0
```

(2) somma degli interi maggiori di K di una lista di interi

```
sumK : int list -> int -> int
sumK ls k = fold (+) (map (fun x -> if x > k then x else 0) ls) 0
```

③ isIn 'a 'a list → bool

isIn a ls = foldr (||) (map (fun x → x = a) ls) false