Oggi lavorate voi

Haskell

Esercizio 1

Scrivere una funzione che calcola i il prodotto scalare di due liste di numeri.

Ad esempio scalare [2,3,4] [1,2,0] si riduce a 8

Esercizio 2

Scrivere una funzione coppie che costruisce, a partire da una lista di numeri, una lista di coppie in cui:

1. il primo elemento di ogni coppia è uguale alla somma di tutti gli elementi che seguono nella lista originale e
2. il secondo elemento di ogni coppia è uguale alla somma di tutti gli elementi antecedenti della lista originale.

Esercizio 3

Scrivere il tipo dell'espressione Haskell

take 3

- **Esercizio 4**

  Si assumano I seguenti fatti su persone e età delle persone.

  ```
  person(fred).
  person(peter).
  person(ann).
  person(beth).
  person(tom).
  person(talullah).
  age(peter,10).
  age(ann,5).
  age(beth,10).
  age(tom,8)
  ```

  scrivere un predicato noage(L) che istanzi L con la lista dei nomi delle persone la cui età non è nota.

- **Esercizio 5**
- Definire il predicato sottolisten(N, ListaIn, ListaOut) che, presa una lista di elementi ground, ListaIn, restituisce la lista composta dalle sottoliste di elementi contigui di ListaIn lunghe N.
- **Esercizio 6**
- Cosa è una sostituzione? Come si applica una sostituzione ad una espressione E? Come si compongono due sostituzioni? Illustrare mediante un esempio che la composizione di sostituzioni non è commutativa.

Soluzione Esercizio 1

```
scalare :: [Int] -> [Int] -> Int
scalare [] [] = 0
scalare (x:xs) (y:ys) = x*y+ scalare xs ys
scalare xs ys = error "liste di lunghezza diversa: non deve succedere"
```

Soluzione Esercizio 2

```haskell
coppie :: [Int] -> [(Int, Int)]
coppie lista = coppieAux 0 (sum lista) lista
  where
    coppieAux _ _ [] = []
    coppieAux precSum succSum (x:xs) =
      let nuovoPrecSum = precSum + x
          nuovoSuccSum = succSum - x
      in (precSum, nuovoSuccSum) : coppieAux nuovoPrecSum nuovoSuccSum xs
```

Soluzione Esercizio 3

- take 3 :: [a] -> [a]

Soluzione Esercizio 4

```
noage(L) :- findall(X, noageapp(X), L).
noageapp(X):- person(X), \+age(X,_).
```

Soluzione Esercizio 5

```
sottoL(0, _, []).
sottoL(N, [X|Resto], [X|L1Resto]) :- N > 0, NuovoN is N - 1, sottoL1(NuovoN, Resto, L1Resto).
sottoL(N, [_|Resto], L1) :- N > 0, sottoL(N, Resto, L1).

sottoL1(0, _, []).
sottoL1(N, [X|Resto], [X|L1Resto]) :- N > 0, NuovoN is N - 1, sottoL1(NuovoN, Resto, L1Resto).


bagof(X, sottoL(3,[1,2,3,4,5],X), L).
```