Primo Compitino di Programmazione - 15 novembre 2013

Cognome ..... **Nome** ..... **Matricola** .....

Esercizio 1. Definire una funzione ricorsiva `compress: 'a list -> 'a list` che data una lista di elementi `l` restituisca la lista ottenuta da `l` sostituendo ogni sequenza di elementi uguali e contigui in `l` con una sola occorrenza di tali elementi.
Per esempio:

```
# compress [1;1;2;2;2;3;3;4;3;3;3])
- : int list = [1;2;3;4;3].
```

Soluzione Esercizio 1.

```
# let rec compress l = match l with
                 [] -> []
                 | [x] -> [x]
                 | x::y::xs when x<>y -> x::compress(y::xs)
                 | x::y::xs when x=y -> compress(x::xs);;
```

Esercizio 2. Scrivere una funzione `factors` che dato un numero intero `n` restituisca la lista dei fattori primi di `n` con le opportune ripetizioni.
Per esempio:

```
# factors 315;;
- : int list = [3; 3; 5; 7].
```

Infine si scriva il tipo della funzione `factors`.

Soluzione Esercizio 2.

```
 # let factors n =
     let rec aux d n =
       if n = 1 then [] else
         if n mod d = 0 then d :: aux d (n / d) else aux (d+1) n
     in
     aux 2 n;;
 factors : int -> int list = <fun>
```

Esercizio 3. La cogettura di Goldbach dice che ogni numero positivo pari maggiore di 2 è la somma di due numeri primi. Scrivere una funzione `goldbach` che dato un numero intero `n` restituisca una coppia di numeri primi che sommati danno `n`.
Per esempio:

```
# goldbach 28;;
- : int * int = (5, 23)
```

Infine si scriva il tipo della funzione `goldbach`.

Soluzione Esercizio 3.

```
# let prime n =
          let rec trynext i =
              (n = i) || ((n mod i !=0) & trynext (i+1))
          in trynext 2;;
prime : int -> bool = <fun>

 # let goldbach n =
     let rec aux d =
       if prime d && prime (n - d) then (d, n-d)
       else aux (d+1) in
     aux 2;;
 val goldbach : int -> int * int = <fun>
```

Esercizio 4. Scrivere una funzione `remove` che dati un numero intero `k` e una lista `l` restituisca la lista che si ottiene da `l` rimuovendo il `k`-eseimo elemento, assumendo che il primo elemento della lista si trovi in posizione 0, il secondo in posizione 1, e così via.
Per esempio:

```
# remove 1 [1.0; 5.3; 7.9; 6.8];;
- : float list = [1.0; 7.9; 6.8]
```

Infine si scriva il tipo della funzione `remove`.

Soluzione Esercizio 4.

```
# let rec remove k l = match l with
     | [] -> []
     | h :: t -> if k = 0 then t else h :: remove (k-1) t;;
 val remove : int -> 'a list -> 'a list = <fun>
```