Soluzioni Primo Compitino di Programmazione - 20 gennaio 2011
Tema B

Esercizio 1. Scrivere il tipo della seguente funzione `g`:

```
let rec g z =
   match z with
   | [] -> 0
   | x::y -> if (x mod 2 <> 0) then (g y) else  x + (g y);;
```

Che cosa restituisce ?

Soluzione. Il tipo della funzione è:

```
g : int list -> int = <fun>
```

Si tratta della funzione ricorsiva che calcola la somma degli elementi pari di una lista di interi.

Esercizio 2. In matematica, dati due numeri interi a e b, un intervallo $[a,b]$ rappresenta l'insieme di tutti i numeri interi x tali che $a \leq x \leq b$. Scrivere una funzione `intervallo` tale che dati due numeri a e b restituisca l'intervallo corrispondente. Per esempio:

```
(intervallo 0 5)= [0; 1; 2; 3; 4; 5]
(intervallo 6 9) = [6; 7; 8; 9].
```

Si dica se la funzione definita è iterativa o ricorsiva. Infine si scriva il tipo della funzione `intervallo`.

Soluzione.

FUNZIONE RICORSIVA

```
# let rec intervallo a b =
        if a > b then []
        else a::intervallo (a+1) b;;
intervallo : int -> int -> int list = <fun>
```

FUNZIONE ITERATIVA

```
# let intervallo (a,b) =
         let rec aux (a,b,acc) =
             if a>b then acc
             else aux (a, b-1, b::acc)
         in aux (a,b,[]);;
intervallo : int * int -> int list = <fun>
```

Esercizio 3. Definire il tipo albero e scrivere la funzione `count` che conta il numero di nodi presenti nell'albero. Inoltre si scriva il tipo della funzione `count`.

Soluzione.

```
type a btree = Empty | Node of a * a btree * a btree;;
Type btree defined.
```

```
# let rec count bt = match bt with
           Empty -> 0
           | Node (x, lt, rt) -> 1 + (count lt) + (count rt);;
count : 'a btree -> int = <fun>
```

Esercizio 4. Scrivere una funzione `maxMin` che, data una lista non vuota di liste non vuote di interi, restituisca il valore massimo tra i minimi di ciascuna lista. Per esempio:

`maxMin([[3;100;1;9]; [2;10;20]; [80;65;4]]) = 4.`

Infine si scriva il tipo della funzione `maxMin`.

Soluzione.

```
# let rec max lst =
        match lst with
        | [x] -> x
        | x::y::ys -> if (x>y) then max (x::ys) else max (y::ys);;
max : 'a list -> 'a = <fun>
# let rec min lst =
        match lst with
        | [x] -> x
        | x::y::ys -> if (x<y) then min (x::ys) else min (y::ys);;
min : 'a list -> 'a = <fun>
#let rec maxMin lst =
        match lst with
        | [x] -> man x
        | x::y::ys -> max [(min x); (maxMin (y::ys))];;
maxMin : 'a list list -> 'a = <fun>
```

Esercizio 5. Si definisca una funzione `scambia` che, data una lista `l` e un indice intero `j` con $j \geq 0$, restituisce la lista che si ottiene da `l` scambiando l'ordine degli elementi di posizione `j` e `j+1` (e solo di questi due elementi). L'indice della prima posizione della lista è `0`; inoltre, se `j` o `j+1` non sono indici validi, allora la funzione `scambia` restituisce la lista `l` immutata. Per esempio:

```
scambia([1;2;3;4;5;6;7;8],0) = [2;1;3;4;5;6;7;8]
scambia([1;2;3;4;5;6;7;8],3) = [1;2;3;5;4;6;7;8]
scambia([1;2;3;4;5;6;7;8],6) = [1;2;3;4;5;6;8;7]
scambia([1;2;3;4;5;6;7;8],7) = [1;2;3;4;5;6;7;8]
```

Inoltre si scriva il tipo della funzione `scambia`.

Soluzione.

```
#let rec scambia (l,j) = match (l,j) with
         ([],j) | ([x],j)  -> l
         | (x::y::xs, 0) -> y::x::xs
         | (x::xs, j) -> x:: scambia(xs,j-1);;
sambia : 'a list * int -> 'a list = <fun>
```