

Laboratorio 3: Le liste di naturali (seconda parte)

Fausto Spoto

31 gennaio 2005

minmax lst la coppia (minimo,massimo) di lst

```
let rec minmax_aux min max lst = match lst with
  [] -> (min,max) | head :: tail ->
  if (sup head max)
  then minmax_aux min head tail
  else if (sup min head)
  then minmax_aux head max tail
  else minmax_aux min max tail;;
```

```
exception Empty_list;;
```

```
let minmax lst = match lst with
  [] -> raise Empty_list
  | head :: tail -> minmax_aux head head tail;;
```

```
val minmax : natural list->natural*natural = <fun>
# minmax [S Z; S (S (Z)); S Z; Z; S (S (S Z))];;
- : natural * natural = (Z, S (S (S Z)))
```

add_follows lst aggiunge a ogni elemento la somma di quelli che lo seguono

```
let rec add_follows_aux lst = match lst with
  [] -> ([],Z)
  | head :: tail -> match add_follows_aux tail
    with (l,s) -> (add head s :: l,add head s);;

let add_follows lst = match add_follows_aux lst
  with (l,_) -> l;;
```

```
val add_follows : natural list->natural list=<fun>
# add_follows [Z;Z;S (S Z);S Z];;
- : natural list = [S (S (S Z)); S (S (S Z));
                   S (S (S Z)); S Z]
#
```

la testa del risultato è la somma di ciò che segue!

```
let rec add_follows lst = match lst with
  [] -> []
| [head] -> [head]
| head :: tail -> match add_follows tail with
  [] -> []
  | hd :: tl -> (add head hd) :: hd :: tl;;
```

Il primo caso del secondo pattern matching è ridondante ma il compilatore lo richiede...

Definite le seguenti funzioni su liste

<code>ordered lst</code>	=	<code>lst</code> è in ordine crescente
<code>sum lst</code>	=	la somma degli elementi di <code>lst</code>
<code>suppress x lst</code>	=	<code>lst</code> senza le occorrenze di <code>x</code>
<code>select x lst</code>	=	le liste ottenute levando da <code>lst</code> una occorrenza di <code>x</code>
<code>perm lst</code>	=	tutte le permutazioni di <code>lst</code>