

## Laboratorio 2: Le liste di naturali (prima parte)

Fausto Spoto

28 gennaio 2005

## Ocaml ha le liste...

- Sono costruite a partire dalla lista vuota `[]`
- E dal *cons* `::`
- Esiste anche una sintassi enumerativa: `[e1; e2; ...; en]`

Le liste di naturali hanno elementi del tipo `natural` definito nel primo laboratorio

## Esempi:

`[Z]`, `[SZ; Z; SZ]`, `[]`, `[S(S(SZ)); S(SZ)]`

# element x lst se e solo se x appartiene a lst

equal deve essere definito

```
let rec element x lst = match lst with
  [] -> false
  | y :: tail -> if (equal y x) then true
                  else (element x tail);;
```

```
val element : natural -> natural list -> bool
          = <fun>
# element (S Z) [Z;Z;S (S Z);S Z];;
- : bool = true
# element (S Z) [Z;Z;S (S Z)];;
- : bool = false
#
```

noelement x lst se e solo se x non appart. a lst

diff deve essere definito

```
let rec noelement x lst = match lst with
  [] -> true
  | y :: tail -> if (diff y x) then
                    (noelement x tail)
                  else false;;
```

```
val noelement : natural -> natural list -> bool
      = <fun>
# noelement (S Z) [Z;Z;S (S Z);S Z];;
- : bool = false
# noelement (S Z) [Z;Z;S (S Z)];;
- : bool = true
#
```

Definite le seguenti funzioni sui nostri naturali (senza usare =)

<code>length lst</code>	=	la lunghezza di <code>lst</code>
<code>last lst</code>	=	l'ultimo elemento di <code>lst</code>
<code>equal lst1 lst2</code>	=	<code>lst1=lst2</code>
<code>prefix lst1 lst2</code>	=	" <code>lst1</code> è un prefisso di <code>lst2</code> "
<code>suffix lst1 lst2</code>	=	" <code>lst1</code> è un suffisso di <code>lst2</code> "
<code>sublist lst1 lst2</code>	=	" <code>lst1</code> è una sottolista di <code>lst2</code> "
<code>append lst1 lst2</code>	=	la concatenaz. di <code>lst1</code> e <code>lst2</code>