PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2015/16
Prova scritta del 16/12/2015
SOLUZIONI PROPOSTE

ESERCIZIO 1
Dato il tipo degli alberi binari

```
type 'a btree = Void  | Node of 'a * 'a btree * 'a btree
```

si definisca in `CAML` una funzione `subst` con tipo

```
subst : int btree -> int btree -> int btree
```

in modo che `(subst bt tr)` restituisca l'albero ottenuto da `bt` rimpiazzando con `tr` tutte le foglie con valore positivo.

Soluzione

```
let rec subst bt tr = match bt with
  Void  ->  Void
| Node(x, Void, Void) when x>0 -> tr
| Node(x, Void, Void) when x<=0 -> Node(x, Void, Void)
| Node(x, lbt, rbt) when lbt<>Void or rbt<>Void -> Node(x, subst lbt br, subst rbt tr)
```

ESERCIZIO 2
Definire in CAML una funzione

```
sumflat : int list list -> int list
```

in modo che `(sumflat ll)` restituisca la lista di interi in cui al posto di ogni lista `xs` in `ll` compare la somma dei valori in `xs`. Si ricorda che la somma dei valori contenuti nella lista vuota è convenzionalmente 0.

Soluzione
Soluzione 1

```
let rec sumflat ll =
     let rec sum l = match l with
        [] -> 0
      | x::xs -> x + (sum xs)
     in
        match ll with
          [] -> []
        | l::ls -> (sum l) :: (sumflat ls);;
```

Soluzione 2

```
let sumflat ll =
     let rec sum l = match l with
        [] -> 0
      | x::xs -> x + (sum xs)
     in
        map sum ll;;
```

Soluzione 3

```
let sumflat ll =
     let sum l = let f x y = x+y in foldr f 0 l
     in
     let f x y = (sum x)::y
       in foldr f [] ll;;
```

ESERCIZIO 3
Indicare il tipo delle seguenti funzioni

- `let f x y z = (x y) = ((y z) + 1);;`
- `let f x = let g y z = y+z=x in g;;`

Soluzione

- `let f x y z = (x y) = ((y z) + 1);;`

  ```
  f : (('a -> int) -> int) -> ('a -> int) -> 'a -> bool
  ```

- `let f x = let g y z = y+z=x in g;;`

  ```
  f : int -> int -> int -> bool
  ```

ESERCIZIO 4
Definire una funzione ricorsiva f da coppie di naturali in naturali che soddisfi la proprietà

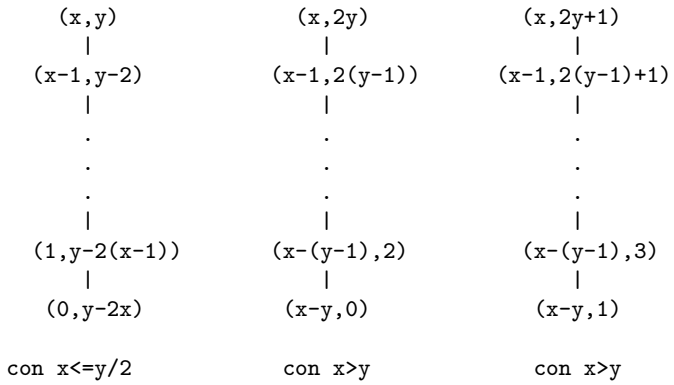
$$\forall n, m \in \mathbb{N}.\ f(n, m) = n + 2m + 3$$

in modo che la relazione di precedenza indotta su $\mathbb{N} \times \mathbb{N}$ sia la seguente

$$\forall n, m, n', m' \in \mathbb{N}. \langle n, m \rangle \prec_f \langle n', m' \rangle \equiv (n = n' - 1 \wedge m' = m + 2)$$

Dimostrare per induzione ben fondata la correttezza della definizione data.

Soluzione
Una possibile rappresentazione grafica della relazione indicata è la seguente



dove `x,y` indicano generici numeri naturali.

Dunque:

$$f(n, m) = \begin{cases} 2m + 3 & \text{se } n = 0 \\ n + 3 & \text{se } m = 0 \\ n + 5 & \text{se } m = 1 \\ f(n - 1, m - 2) + 5 & \text{se } n > 0 \wedge m > 1 \end{cases}$$

Omettiano la dimostrazione dei casi base. Dimostriamo il caso induttivo:

$$\forall n, m \in \mathbb{N}.\ f(n, m) = n + 2m + 3 \Longrightarrow f(n + 1, m + 2) = (n + 1) + 2(m + 2) + 3$$

$f(n + 1, m + 2)$
$=$ {def. di f, quarto caso}
$f(n, m) + 5$
$=$ {Ip. induttiva}
$n + 2m + 3 + 5$
$=$ {calcolo}
$(n + 1) + 2m + 3 + 4$
$=$ {calcolo}
$(n + 1) + 2(m + 2) + 3$

ESERCIZIO 5
Date le seguenti definizioni:

```
struct el {int info; struct el *next;};

typedef struct el ElementoDiLista;
typedef ElementoDiLista *ListaDiInteri;
```

scrivere in C una procedura che, data in ingresso attraverso un opportuno parametro una lista di interi, porta in ultima posizione il primo elemento negativo che compare nella lista. La procedura lascia la lista inalterata se in essa non occorre alcun elemento negativo.

Soluzione

```
void proc (ListaDiInteri *l)
   {
     if (*l != NULL)
     {
        ListaDiInteri precneg, neg, prec, corr;
        int trovato=0;
        prec=NULL; precneg=NULL; corr=*l;
        while (corr != NULL && !trovato)
          {
            if (corr->info<0)
               {
                 neg=corr;
                 precneg = prec;
                 trovato = 1;
               }
            prec=corr;
            corr=corr->next;
          }
        if (trovato)
          while (corr !=NULL)
            {
              prec=corr;
              corr=corr->next;
            }
        if (trovato)
          {
            if (precneg==NULL)
             {
               prec->next=neg;
               *l = *l->next;
             }
            else
             {
               prec->next=neg;
               precneg->next=neg->next;
             }
            neg -> next = NULL;
          }
        }
   }
```