# PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2012-2013
# II Verifica scritta del 19/12/2012

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 `check` con tipo

```
check : 'a btree -> int -> bool
```

in modo che `(check bt n)` restituisca `true` se sono al più `n` i nodi di `bt` con due figli, `false` altrimenti.

**Soluzione**

```
let check bt n =
  let rec conta_nodi_2_figli bt = match bt with
     Void -> 0 |
     Node(_, lt, rt) -> if (lt<>Void & rt<>Void)
                           then 1 + (conta_nodi_2_figli lt) + (conta_nodi_2_figli rt)
                           else     (conta_nodi_2_figli lt) + (conta_nodi_2_figli rt)
  in (conta_nodi_2_figli bt) <= n ;;
```

**ESERCIZIO 2**
Scrivere in `C` una funzione

```
boolean check (int a [], int b [], int dima, int dimb)
```

che, dati due array `a` e `b` di dimensione `dima` e `dimb` rispettivamente, verifica che ogni elemento di `a` occorra in `b` un numero dispari di volte.

**Soluzione**

```
boolean check (int a [], int b [], int dima, int dimb)
  {
    int ia, ib, conta;
    boolean trovato = false;
    ia=0;
    while (ia < dima && !trovato)
       {
         conta = 0;
         for (ib=0; ib < dimb; ib++)
            if (a[ia]==b[ib])
              conta = conta + 1;
         trovato = (conta % 2 == 0);
         ia = ia + 1;
       }
    return (!trovato);
  }
```

**ESERCIZIO 3**
Scrivere in `C` una procedura

```
void azzera (int a[], int dim)
```

che azzera ogni elemento dell'array minore della somma degli elementi che lo precedono, fino al primo elemento uguale a 0.
Dato ad esempio il seguente array `vet1`

| -1 | 6 | 8 | 7 | 3 | 0 | 5 | 3 |
|---|---|---|---|---|---|---|---|

la chiamata `azzera(vet1,8)` deve modificare `vet1` come segue

| 0 | 6 | 8 | 0 | 0 | 0 | 5 | 3 |
|---|---|---|---|---|---|---|---|

Dato invece il seguente array `vet2`

| 2 | 1 | 8 | 20 | 3 |
|---|---|---|---|---|

la chiamata `azzera(vet2,5)` deve modificare `vet2` come segue

| 2 | 0 | 8 | 20 | 0 |
|---|---|---|---|---|

**Soluzione**

```
void azzera (int a[], int dim)
  {
    int i, somma_prec;
    boolean trovato_zero = false;
    i=0;
    somma_prec=0;
    while (i < dim && !trovato_zero)
       {
         if (a[i]==0)
          trovato_zero = true;
         else
            if (a[i]<somma_prec)
               {
                 somma_prec = somma_prec + a[i];
                 a[i]=0;
               }
            else
                 somma_prec = somma_prec + a[i];
         i = i + 1;
       }
  }
```

**ESERCIZIO 4**
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, antepone ad ogni elemento negativo un elemento nullo.

**Soluzione**

```
void p (ListaDiElementi *lista)
   {
      if (*lista != NULL)
        {
          ListaDiElementi prec, corr, aux;

          if (*lista -> info < 0)
            {
              aux = malloc(sizeof(ElementoDiLista));
              aux -> info = 0;
              aux -> next = *lista;
              *lista = aux;
              prec = *lista -> next;
              corr = *lista -> next -> next;
            }
          else
            {
              prec = *lista;
              corr = *lista -> next;
            }

          while (corr != NULL)
            {
               if (corr -> info <0)
                {
                  aux = malloc(sizeof(ElementoDiLista));
                  aux -> info = 0;
                  aux -> next = corr;
                  prec->next = aux;
                }
                prec = corr;
                corr = prec->next;
            }
        }
   }
```

**ESERCIZIO 5**
Senza utilizzare ricorsione esplicita, definire in CAML una funzione

```
foo : 'a list -> ('a -> bool) -> 'a -> ('a * int)
```

in modo che `(foo lis p x)` restituisca la coppia `(w,n)` dove `n` è il numero di elementi di `lis` che soddisfa `p` e `w` è il primo elemento di `lis` che soddisfa `p`, oppure la coppia `(x,0)` se nessun elemento di `lis` soddisfa `p`.

**Soluzione**

```
let foo lis p x =
  let f z (y,n) = if (p z) then (z, n+1) else (y,n)
in foldr f (x,0) lis;;
```