PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2008-2009
Esercitazione del 14/12/2009

ESERCIZIO 1
Dato il tipo degli alberi binari

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

si definisca in CAML una funzione `minmax` con tipo

```
minmax : 'a btree -> 'a * 'a
```

che dato un albero binario restituisce la coppia `(minimo, massimo)` dove `minimo` è l'elemento minimo nell'albero e `massimo` è l'elemento massimo nell'albero.

SOLUZIONE PROPOSTA

```
let rec minmax bt = match bt with
    Node(x, Empty, Empty) -> (x,x) |


    Node(x, lt, Empty) when lt <> Empty -> let (mi, ma) = minmax lt in
                                          if x>ma
                                             then (mi, x)
                                             else if x < mi then (x, ma)
                                                            else (mi, ma) |
    Node(x, Empty, rt) when rt <> Empty -> let (mi, ma) = minmax rt in
                                           if x>ma
                                             then (mi, x)
                                             else if x < mi then (x, ma)
                                                            else (mi, ma) |
    Node(x, lt, rt) when lt <> Empty and rt <> Empty -> let (mi1, ma1) = minmax lt
                                                        and (mi2, ma2) = minmax rt
                                                        in
                                                          let (mi,ma) = (min(mi1, mi2), max(ma1, ma2))
                                                          in
                                                             if x>ma
                                                             then (mi, x)
                                                             else if x < mi then (x, ma)
                                                                            else (mi, ma);
```

ESERCIZIO 2
Scrivere in C una funzione

```
int somma (int a[], int dim)
```

che restituisce la somma di tutti gli elementi dell'array `a` di dimensione `dim` diversi dall'elemento massimo.

SOLUZIONE PROPOSTA

```
int somma (int a[], int dim)
{
int max, occ, i, sum;
max = a[0]; occ=1; sum = 0;
for (i=1; i<dim; i=i+1)
   if (a[i]==max)
       occ = occ+1;
   else
      if (a[i]>max)
         {
           sum = sum + max*occ;
           max = a[i];
           occ = 1;
         }
       else
       sum = sum + a[i];
return sum;
}
```

ESERCIZIO 3
Scrivere in C una procedura

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

che azzera gli ultimi due elementi negativi dell'array `a` di dimensione `dim`.

SOLUZIONE PROPOSTA

```
void azzera (int a[], int dim)
{
   int i = dim -1;
   int cancellati = 0;

      while (cancellati < 2 && i>=0)
         {
            if (a[i] < 0)
                {  a[i] = 0; cancellati = cancellati + 1;}
            i = i-1;
         }
}
```

ESERCIZIO 4
Date le seguenti definizioni:

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

typedef struct el ElementoLista;
typedef ElementoLista *ListaDiElementi;
```

scrivere in C una funzione

```
int conta (ListaDiElementi lista)
```

che restituisce il numero degli elementi nulli di `lista`.

SOLUZIONE PROPOSTA

```
int conta (ListaDiElementi lista)
 {
    int c = 0;
    while (lista != NULL)
       {
         if (lista -> info == 0)
             c = c + 1;
         lista = lista -> next;
       }
    return c;
 }
```

ESERCIZIO 5
Date le definizioni dell'esercizio 4, scrivere in `C` una procedura

```
void ins (ListaDiElementi *lista, int el)
```

che inserisce, nella lista passata come argomento, l'elemento `el` prima del primo elemento pari.

SOLUZIONE PROPOSTA

```
void ins (ListaDiElementi *lista, int el)
 {
     ListaDiElementi aux, corr, prec;
     int trovato;

     if (*lista != NULL)
      {
         if (*lista->info % 2 == 0)
            {
              aux = malloc(sizeof(ElementoLista));
              aux -> info = el;
              aux -> next = *lista;
              *lista = aux;
            }
          else
            {
               corr = *lista -> next;
               prec = *lista;
               trovato = 0;
               while (!trovato && corr != NULL)
                   if (corr->info % 2 == 0)
                       trovato = 1;
                   else
                      {
                        prec = corr;
                        corr = corr -> next;
                      }
               if (trovato)
                 {
                   aux = malloc(sizeof(ElementoLista));
                   aux -> info = el;
                   prec -> next = aux;
                   aux -> next = corr;
                 }
            }
       }
 }
```

ESERCIZIO 5bis
Date le definizioni dell'esercizio 4, scrivere in `C` una procedura

```
void ins (ListaDiElementi *lista, int el)
```

che inserisce, nella lista passata come argomento, l'elemento `el` prima del primo elemento pari. Se la lista non contiene elementi pari, `el` viene inserito in coda.

SOLUZIONE PROPOSTA

```
void ins (ListaDiElementi *lista, int el)
 {
     ListaDiElementi aux, corr, prec;
     int trovato;

     aux = malloc(sizeof(ElementoLista));
     aux -> info = el;
     if (*lista == NULL)
         {
           aux -> next = NULL;
           *lista = aux;
         }
     else
         if (*lista -> info % 2 == 0)
            {
              aux -> next = *lista;
              *lista = aux;
            }
          else
            {
               corr = *lista -> next;
               prec = *lista;
               trovato = 0;

               while (!trovato && corr != NULL)
                   if (corr -> info % 2 == 0)
                       trovato = 1;
                   else
                      {
                        prec = corr;
                        corr = corr -> next;
                      }
               prec -> next = aux;
               aux -> next = corr;
            }
 }
```