PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2011-2012
Prova scritta del 31/01/2012

SOLUZIONI PROPOSTE

Negli esercizi di programmazione in `C` si assuma predefinito il tipo `typedef enum {false, true} boolean`.

ESERCIZIO 1 (punti 6)
Dato il tipo degli alberi binari visto a lezione

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

si definisca in CAML una funzione `check` con tipo

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

in modo che `(check bt x n)` restituisca *true* se l'albero `bt` contiene una foglia etichettata `x` a profondità `n`, restituisca *false* altrimenti.

Soluzione

```
let rec check bt x n = match (bt, n) with

  _,n  when n<=0 -> false |

  Node(y, Void, Void), 1 -> x=y |

  Node(_,lt,rt), n when n>1 -> (check lt x (n-1)) or (check rt x (n-1));;
```

ESERCIZIO 4 (punti 6)
Si completi la seguente definizione

```
 let foo n l = let f x y = ...  in foldr f ... l
```

in modo che `(foo z lis)` restituisca la coppia `(num, sum)`, in cui `num` è il numero di elementi di `lis` maggiori di `z` e `sum` è la somma di tali elementi.

Soluzione

```
let foo n l = let f x (num, sum) =
                   if (x>n) then (num+1, sum+x) else (num, sum)
              in
                 foldr f (0,0) l
```

ESERCIZIO 2 (punti 6)
Si definisca in `C` una funzione

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

che restituisce il valore di verità della seguente formula:

$$\forall i \in [0, dima).\ (\exists j, k \in [0, dimb).\ j \neq k\ \wedge\ a[i] = b[j]\ \wedge\ a[i] = b[k])$$

Soluzione
Forniamo due possibili soluzioni.

Prima soluzione

```
boolean twice (int a [], int dima, int b[], int dimb)
{
   boolean perogni = true;
   int i=0;
   while(i<dima && perogni)
     {
       int j=0;
       boolean trovati=false;
       while (j<dimb && !trovati)
          if (a[i]==b[j]) trovati=true;
              else j=j+1;
       if (trovati)
         { trovati=false;
           j=j+1;
           while (j<dimb && !trovati)
              if (a[i]==b[j]) trovati=true;
                  else j=j+1;
         }
       perogni = trovati;
       i = i+1;
     }
   return perogni;
}
```

Seconda soluzione

```
boolean twice (int a [], int dima, int b[], int dimb)
{
   boolean perogni = true;
   int i=0;
   while(i<dima && perogni)
     {
       int j=0;
       int dueuguali = 0;
       while (j<dimb && (dueuguali != 2))
          { if (a[i]==b[j]) dueuguali = dueuguali+1;
            j=j+1;
          }
       perogni = (dueuguali == 2);
       i = i+1;
     }
   return perogni;
}
```

ESERCIZIO 3 (punti 6)
Si consideri la seguente grammatica $G = \langle \{1,2,3\}, \{\texttt{S}, \texttt{D}, \texttt{T}, \texttt{A}, \texttt{B}\}, \texttt{S}, P \rangle$, dove le produzioni in P sono le seguenti:

```
S ::= 1S | 1D
D ::= 2D | 2T
T ::= 3T | 3A
A ::= 2A | 2B
B ::= 1B | 1
```

Scrivere in `C` una funzione che, dato un array di interi `a` e la sua dimensione `dim` restituisce `true` se la sequenza `a[0]a[1]...a[dim-1]` appartiene al linguaggio generato da G, e restituisce `false` altrimenti.

Soluzione

```
boolean check (int a [], int dim)
{
 boolean app = false;
 if (a[0]==1)
   {
    int i=1;
    boolean go = true;
    while(i<dim && go)
       if (a[i]==a[i-1] || a[i]==a[i-1]+1)
          i = i+1;
       else go = false;
    if (i < dim)
      if (a[i-1]==3)
        {
           go = true;
           while (i<dim && go)
              if (a[i]==a[i-1] || a[i]==a[i-1]-1)
                i = i+1;
              else go = false;
           if (go && a[i-1]==1) app = true;
        }
     }
 return app;
}
```

ESERCIZIO 5 (punti 6)
Date le seguenti definizioni:

```
struct el {int info; struct el *next;};
typedef struct el ElementoDiLista;
typedef ElementoDiLista *ListaDiElementi;
```

scrivere in `C` una procedura che, dati in ingresso attraverso opportuni parametri una lista di interi `lista1`, un intero `x` e una seconda lista di interi `lista2`, modifica `lista1` concatenando ad essa un nuovo elemento contenente il valore `x`, seguito dalla lista `lista2`.

Soluzione

```
void concat (ListaDiElementi *lista1, int x, ListaDiElementi lista2)
{
 ListaDiElementi aux = malloc(sizeof(ElementoDiLista));
 aux -> info = x;
 aux -> next = lista2;
 if (*lista1 == NULL) *lista1=aux;
 else
   {
     ListaDiElementi corr=*lista1;
     while (corr -> next != NULL)
       corr = corr-> next;
     corr -> next = aux;
   }
}
```