PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2015-2016
Prova scritta del 6/07/2016

SOLUZIONI PROPOSTE

Per ogni esercizio vengono proposte una o più soluzioni tra le tante possibili.

ESERCIZIO 1 (6 punti)
Si definisca una grammatica libera sull'alfabeto $\Lambda = \{(,),[,],\{,\}\}$ che genera le stringhe di parentesi bilanciate in modo che le parentesi quadrate e graffe non stiano mai all'interno di parentesi tonde e le parentesi graffe non stiano mai all'interno di parentesi quadrate. Ad esempio, detto $\mathcal{L}$ il linguaggio generato dalla grammatica:

$$([\,]()) \notin \mathcal{L} \qquad\qquad \{()[\,](())\} \in \mathcal{L}$$

Soluzione

```
S ::= {} | {S} | S S | A
A ::= [] | [A] | A A | B
B ::= () | (B) | B B
```

ESERCIZIO 2 (6 punti)
Dato un albero binario, si definisce *livello* di un nodo nell'albero il numero di nodi che si incontrano nel cammino dalla radice al nodo medesimo. Quindi, ad esempio, la radice ha livello 1, i figli della radice hanno livello 2, e così via. Dato il tipo degli alberi binari visti a lezione

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

si scriva in CAML una funzione

```
nonleaves : 'a btree -> 'a list * 'a list
```

che, dato un albero binario, restituisce una coppia di liste $(l1, l2)$ in cui $l1$ contiene i valori dei nodi non foglia di livello pari e $l2$ contiene i valori dei nodi non foglia di livello dispari.

Soluzione

```
let nonleaves bt =
   let rec foo bt b = match bt with
     Void -> [], [] |
     Node(_, Void, Void) -> [], []  |
     Node(x, lt, rt) when lt<>Void or rt<> Void ->
            let l1,l2= foo lt (not b) in
               let ll1,ll2 = foo rt (not b) in
               if b then l1@ll1, x::l2@ll2 else x::l1@ll1, l2@ll2
   in
     foo bt true;;
```

ESERCIZIO 3 (6 punti)
Si definisca in CAML, senza utilizzare ricorsione esplicita, una funzione

```
minfirst : 'a list -> 'a list
```

in modo che `(minfirst xs)` sia una lista che contiene tutti gli elementi di xs, in cui le occorrenze dell'elemento minimo sono tutte in testa (l'ordine in cui gli elementi non minimi di xs occorrono nella lista risultato è irrilevante). Ad esempio

```
minfirst [2;3;4;1;3;1;6] = [1;1] @ L
```

dove `L` è una qualunque permutazione della lista `[2;3;4;3;6]`

Soluzione

```
let minfirst xs = let f x y = match y with
                    [] -> [x] |
                    z::zs when x>z -> y@[x] |
                    z::zs when x<=z -> x::y
                  in foldr f [] xs;;
```

ESERCIZIO 4 (6 punti)
Si scriva in C una procedura che, presi attraverso opportuni parametri una lista di interi ed un intero n, cancella dalla lista i primi n elementi maggiori di 0. Si suppongano predefiniti i tipi:

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

Soluzione

```
void del (ListaDiInteri *lis, int n)
{
   ListaDiInteri prec=NULL, corr=*lis;
   while (corr != NULL && n>0)
     {
       if (corr->info > 0)
          {
            n=n-1;
            if (prec==NULL)
               { *lis = *lis -> next;
                 free(corr);
                 corr = *lis;
               }
            else
               {
                 prec->next = corr->next;
                 free(corr);
                 corr = prec->next;
               }
          }
       else
          {
            prec = corr;
            corr = corr->next;
          }
     }
}
```

ESERCIZIO 5 (6 punti)
Si scriva in C una funzione che, dato un array di interi a e la sua dimensione dim, restituisce il valore di verità della seguente formula

$$\exists\, i, j \in [0, dim).\ a[i] \neq a[j]\ \ \wedge\ \ \#\{k \mid k \in [0, dim) \wedge a[k] = a[i]\} = \#\{h \mid h \in [0, dim) \wedge a[h] = a[j]\}$$

Si ricorda che dato un insieme finito A, $\#A$ indica il numero di elementi di A.

Soluzione

```
int check (int a[], int dim)
{
   int i=0, trovato = 0;
   while (i < dim && !trovato)
     {
       int j = i+1;
       while (j<dim && !trovato)
         {
          if (a[i] == a[j]) j = j+1;
             else trovato=1;
          if (trovato)
            {
              int conta=0; int k;
              for (k=0; k<dim; k++)
               if (a[k]==a[i]) conta = conta + 1;
                  else if (a[k]==a[j]) conta = conta-1;
              if (conta != 0)
                {trovato=0; j=j+1;}
            }
         }
         i=i+1;
     }
   return trovato;
}
```