PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2015-2016
Prova scritta del 8/06/2016

SOLUZIONI PROPOSTE

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

ESERCIZIO 1 (6 punti)
Data una stringa α ed un simbolo x denotiamo con $|\alpha|_x$ il numero di occorrenze di x in α. Si definisca una grammatica libera che genera il seguente linguaggio sull'alfabeto $\Lambda = \{a, b, c\}$:

$$\mathcal{L} = \{\alpha a \beta \mid \alpha, \beta \in \Lambda^+ \ \wedge \ |\alpha|_a = 0 \ \wedge \ |\beta|_c = 1\}$$

Soluzione

```
S ::= AaB
A ::= b | c | bA | cA
B ::= aB | bB | c | cC
C ::= a | b | aC | bC
```

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

```
leaves : 'a btree -> 'a*int list
```

che, dato un albero binario, restituisce una lista di coppie in cui ciascuna coppia contiene il valore di una foglia ed il suo livello. La lista deve contenere una coppia per ciascuna foglia presente nell'albero. L'ordine degli elementi nella lista risultato è irrilevante.

Soluzione

```
let leaves bt =
   let rec foo bt n = match bt with
     Void -> [] |
     Node(x, Void, Void) -> [(x,n)]  |
     Node(_, lt, rt) when lt<>Void or rt<> Void ->
               (foo lt (n+1)) @ (foo rt (n+1))
   in
     foo bt 1;;
```

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

```
split : 'a list -> 'a list * 'a list
```

in modo che `(split xs)` restituisca la coppia $(l1, l2)$, in cui $l1$ contiene tutti gli elementi di xs maggiori del primo elemento di xs e $l2$ contiene tutti gli elementi di xs minori o uguali al primo elemento di xs. Se xs è vuota la funzione restituisce la coppia [ ], [ ].

Soluzione

```
let split xs = match xs with
  [] -> [],[]   |
  z::_ -> let f x (l1,l2) = if x>z then z::l1, l2 else l1, x::l2 in foldr f ([],[]) xs;;
```

Una seconda soluzione che utilizza la funzione di ordine superiore `filter`

```
let split xs = match xs with
  [] -> [],[]   |
  z::_ -> let f x = x>z in let g x = x<=z  in filter f xs, filter g xs;;
```

ESERCIZIO 4 (6 punti)
Si scriva in C una procedura che, prese attraverso opportuni parametri due liste di interi, cancella dalla prima tutti gli elementi che non compaiono nella seconda. Si suppongano predefiniti i tipi

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

Soluzione

```
int member (ListaDiInteri l, int x)
{
   int trovato = 0;
   while (l != NULL && !trovato)
     if (l->info == x)
       trovato = 1;
     else
       l = l->next;
   return trovato;
}

void del (ListaDiInteri *prima, seconda)
{
   ListaDiInteri prec=NULL, corr=*prima;
   while (corr != NULL)
     {
       if (!member(seconda, corr->info))
          {
            if (prec==NULL)
               { *prima = *prima -> next;
                 free(corr);
                 corr = *prima;
               }
            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

$$\exists i.\ i \in [0, dim) \ \wedge \ (\forall j \in [0, i).a[j] > a[i]) \ \wedge \ \#\{k \mid k \in [i+1, dim) \wedge a[k] > a[i]\} = 1$$

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

Soluzione

```
int check (int a[], int dima)
{
   int i=0, trovato = 0;
   while (i < dima & !trovato)
     {
       int j = 0;
       int tuttimaggiori = true;
       while (j<i && tuttimaggiori)
          if (a[j] > a[i]) j = j+1;
             else tuttimaggiori = false;
       if (tuttimaggiori)
          {
            int k = i+1;
            int conta = 0;
            while (k<dim && conta <2)
              {
                 if (a[k]>a[i]) conta = conta+1;
                 k = k+1;
              }
            if (conta == 1) trovato = 1;
          }
       i = i+1;
       }
   return trovato;
}
```