PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2012/13
Prova scritta del 10 gennaio 2013 - SOLUZIONI PROPOSTE

ESERCIZIO 1
Si definisca una grammatica libera che genera il linguaggio

$$\mathcal{L} = \Lambda^+ \setminus \{a\}$$

sull'alfabeto $\Lambda = \{a, b\}$.

Soluzione

```
S ::= aA | bA | b
A ::= aA | bA | a | b
```

ESERCIZIO 2
Si consideri il seguente tipo di alberi *ternari* in cui ogni nodo ha al più tre figli.

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

Si definisca una funzione `check` con tipo

```
check : 'a ttree -> boolean
```

in modo che `(check tt)` restituisca `true` se ogni nodo non foglia ha al più un figlio non vuoto, e restituisca `false` altrimenti.

```
let rec check tt = match tt with
      Void -> true  |
      Node(_, Void, Void, Void) -> true  |
      Node(_, Void, Void, t) when t<> Void -> check t |
      Node(_, t, Void, Void) when t<> Void -> check t |
      Node(_, Void, t, Void) when t<> Void -> check t |
      _ -> false;;
```

ESERCIZIO 3
Dato un array a di dimensione dim ed un indice $i \in [1, dim-1)$, siano:

- $dprec(i) = \texttt{max}\{|(a[i] - a[j]| \mid j \in [0, i)\}$
- $dsucc(i) = \texttt{max}\{|(a[i] - a[j]| \mid j \in [i+1, dim)\}$

Si definisca in `C` una funzione
`boolean check (int a[], int dim)`
che restituisce il valore di verità della seguente formula:

$$\exists i.\ i \in [1, dim-1) \ \wedge \ dprec(i) = dsucc(i)$$

(**N.B.** $|x - y|$ indica la differenza in valore assoluto tra x e y.
Si assuma predefinito il tipo `typedef enum {false, true} boolean`).

Soluzione
Definiamo per comodità la funzione di supporto che calcola la differenza in valore assoluto di due interi;

```
int abs (int x, int y)
 {
    if (x-y > 0) return (x-y) else return (y-x);
 }

boolean check (int a[], int dim)
{
  int i, j, d1, d2, temp;
  boolean trovato = false;

  i=1;
  while (i < dim-1 && !trovato)
    {
      j=0;
      d1 = abs(a[i], a[j]);
      for (j=1; j<i; j++)
        {
          temp = abs(a[i],a[j]);
          if (temp>d1) d1=temp;
        }
      j=i+1;
      d2 = abs(a[i], a[j]);
      for (j=i+2; j<dim; j++)
        {
          temp = abs(a[i],a[j]);
          if (temp>d2) d2=temp;
        }
      if (d1==d2) trovato = true; else i=i+1;
    }
  return trovato;
}
```

ESERCIZIO 4
Si completi la seguente definizione

```
let foo lis = let f x y = ....  in foldr f ... lis
```

della funzione

```
foo: int list -> int list
```

in modo che (`foo lis`) sia la lista ottenuta da `lis` rimpiazzando ogni elemento con il numero di elementi uguali a `0` nella sottolista di `lis` che inizia dall'elemento stesso.

Soluzione

```
let foo lis = let f x y = match y with
                            [] -> if (x=0) then [1] else [0] |
                            z::zs -> if (x=0) then (z+1)::y else z::y
              in foldr f [] lis
```

ESERCIZIO 5
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 ed un intero x, elimina dalla lista tutti gli elementi maggiori di x ed inserisce in testa alla lista il numero di elementi eliminati.

Soluzione

```
void p (ListaDiElementi *lis, int x)
  {
     boolean trovato=false;
     int eliminati = 0;
     ListaDiElementi temp = malloc(sizeof(ElementoDiLista));
     ListaDiElementi prec, corr;

     while (*l != null && !trovato)
       {  if (*l -> info > x)
              { corr = *l; *l = *l->next; free(corr); eliminati=eliminati+1; }
          else trovato = true; }
     if (trovato)
       { prec = *l;
         corr = *l -> next;
         while (corr != NULL)
           {
             if (corr->info > x)
               { prec->next = corr->next;
                 eliminati = eliminati + 1;
                 free(corr);
                 corr = prec->next; }
             else
               { prec=prec->next; corr=corr->next; }
           }
         }

       temp->info = eliminati;
       temp->next = *l;
       *l = temp;
  }
```

ESERCIZIO 6
Si definisca una procedura C che, dato un array di interi `a` di dimensione `dim` ordinato in senso non decrescente, ed un intero `x`, inserisce `x` nell'array mantenendo l'ordinamento ed eliminando da `a` l'ultimo elemento. Non si possono utilizzare array di supporto.

Soluzione

```
void p (int a[], int dim, int x)
   {
     int i=0;
     boolean trovato = false;
     while (i<dim && !trovato)
       {
         if (a[i]>=x)
            trovato=true;
         else
            i = i+1;
       }
     if (!trovato)
        a[dim-1] = x;
     else
        {
          int j;
          for (j=dim-1; j>i; j--)
            a[j]=a[j-1];
          a[i]=x;
        }
   }
```