PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2015-2016
Prova scritta del 6/09/2016

SOLUZIONI PROPOSTE

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

ESERCIZIO 1 (6 punti)
Si definisca una grammatica **regolare** che genera il linguaggio Exp di espressioni sull'alfabeto $\Lambda = \{a, b, +, *, (,)\}$ in cui le parentesi non possono essere annidate. Ad esempio:

$$a + b + (a * b) \in Exp \qquad a + (b + (a)) \notin Exp \qquad a + *b \notin Exp$$

Soluzione

```
S ::= a | b | aA | bA | (B
A ::= +S | *S
B ::= aC | bC
C ::= ) | )A | +B | *B
```

ESERCIZIO 2 (6 punti)
Dato il tipo degli alberi binari

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

si scriva in CAML una funzione
`depth : 'a btree -> 'a -> int list`
tale che `(depth bt el)` restituisca la lista delle profondità di tutte le occorrenze di `el` in `bt` (la lista vuota se `el` non occorre in `bt`). Si ricorda che la radice di un albero ha profondità 1.

Soluzione

```
let depth bt el =
   let rec foo bt n = match bt with
     Void -> [] |
     Node(x, lt, rt)  ->
            let l1,l2 = foo lt (n+1), foo rt (n+1)
            in
              if x=el then n::(l1@l2) else l1@l2
   in
     foo bt 1;;
```

ESERCIZIO 3 (6 punti)
Si supponga data la seguente funzione CAML

```
let ins el lis =
   let rec insa el l1 l2 =
      match l1 with
        [] -> l2@[el]
      | x::xs when x>=el -> l2@[el]@l1
      | x::xs when x<el  -> insa el xs (l2@[x])
in
   insa el lis [];;
```

Si definisca, senza utilizzare ricorsione esplicita ma utilizzando `ins`, la funzione
`sort : 'a list -> 'a list`
tale che `(sort lis)` ordini `lis` in modo non decrescente.

Soluzione

```
let sort lis = foldr ins [] lis;;
```

ESERCIZIO 4 (6 punti)
Si suppongano predefiniti i tipi

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

Si scriva in `C` una procedura che, presa attraverso un opportuno parametro una lista di almeno tre elementi, scambia le posizioni del primo e del penultimo elemento, **senza utilizzare assegnamenti sui campi** `info` delle strutture che compongono la lista.

Soluzione

```
void scambia (ListaDiInteri *l)
{
ListaDiInteri terzultimo = *l, penultimo=(*l)->next, ultimo=(*l)->next->next;

while (ultimo->next != NULL)
{
  terzultimo=penultimo;
  penultimo=ultimo;
  ultimo=ultimo->next;
}

terzultimo->next = *l;
penultimo->next=(*l)->next;
(*l)->next = ultimo;
*l = penultimo;
}
```

ESERCIZIO 5 (6 punti)
Si scriva in `C` una funzione che, dato un array di interi a di dimensione dim e un intero $m \in [2, dim]$, restituisce il valore di verità della seguente formula

$$\exists i, j \in [0, dim).\ i < j\ \wedge\ (\forall k \in [i, j].\ a[k] = a[i])\ \wedge\ (j - i + 1 \geq m)$$

Soluzione

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

   return trovato;
}
```