PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2014-2015
Prova scritta del 23/06/2015

SOLUZIONI PROPOSTE

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

ESERCIZIO 1 (6 punti)
Si dimostri, utilizzando il pumping lemma, che il seguente linguaggio sull'alfabeto $\{a, b, c\}$ non è regolare

$$\mathcal{L} = \{a^n b^k c^m \mid n, k, m > 0 \wedge (n = k \vee n = m)\}$$

Soluzione
Per dimostrare che il linguaggio non è regolare utilizzando il pumping lemma mostriamo quanto segue:

$$\forall n \in \mathbb{N}.\ \exists w \in \mathcal{L}.\ |\ w\ | \geq n \wedge \forall x, y, z. \neg\big(w = xyz\ \wedge\ y \neq \epsilon \wedge\ |\ xy\ | \leq n\ \wedge\ (\forall i \in \mathbb{N}. xy^i z \in \mathcal{L})\big)$$

ovvero

$$\forall n \in \mathbb{N}.\ \exists w \in \mathcal{L}.\ |\ w\ | \geq n \wedge \forall x, y, z. \big(w = xyz\ \wedge\ y \neq \epsilon \wedge\ |\ xy\ | \leq n) \Rightarrow\ \neg(\forall i \in \mathbb{N}. xy^i z \in \mathcal{L})\big)$$

Sia allora $n \in \mathbb{N}$ e consideriamo, tra le tante possibili, la stringa

$$w = a^n b^n c^n$$

Chiaramente $w \in \mathcal{L}$ e $|\ w\ | \geq n$. Siano dunque x, y, z tali che:

(a) $w = xyz$

(b) $y \neq \epsilon$

(c) $|\ xy\ | \leq n$

Dunque:
$x = a^h\ \wedge\ y = a^t$, con $t > 0$ e $h + t \leq n$, e $z = a^{(n-h-t)} b^n c^n$.
La stringa $xy^0 z = a^h a^{(n-h-t)} b^n c^n = a^{(n-t)} b^n c^n \notin \mathcal{L}$ poiché $t > 0 \Rightarrow n - t \neq n$.

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

```
even : 'a btree -> 'a list
```

che, dato un albero binario, restituisce la lista con i valori di tutti i nodi che occorrono a livelli pari nell'albero.

Soluzione

```
let even bt =
   let rec aux bt b = match bt with
       Void -> [] |
       Node(x,lt,rt) -> if b then x::(aux lt false)@(aux rt false) else (aux lt true)@(aux rt true)
   in
      aux bt false;;
```

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

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

in modo che `(split xs)` restituisca la lista delle più lunghe sottoliste non decrescenti in `xs`.
Ad esempio

```
split [1;2;2;1;5;4;6;3;2;3] = [[1;2;2]; [1;5]; [4;6]; [3]; [2;3]]
```

Soluzione

```
let split xs =
      let f x y = match y with
         [] -> [[x]] |
         (z::zs)::ws -> if x<=z then (x::z::zs)::ws else [x] :: y
      in
          foldr f [] xs;;
```

ESERCIZIO 4 (6 punti)
Si scriva in C una procedura che, prese attraverso opportuni parametri due liste di interi, aggiunge alla seconda lista tutti gli elementi della prima che non appartengono anche alla seconda.
Si suppongano predefiniti i tipi

```
struct el {int info; struct el *next;};

typedef struct el ElementoDiLista;
typedef ElementoDiLista *ListaDiInteri;
```

Soluzione
Scegliamo di aggiungere gli elementi in testa alla seconda lista.

```
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 insTesta (ListaDiInteri *l, int x)
{
  ListaDiInteri new=malloc(sizeof(ElementoDiLista));
  new->info = x;
  new->next = *l;
  *l=*l->next;
}

void add (ListaDiInteri prima, *seconda)
{
   while (prima != NULL)
      { if (!member(*seconda, prima->info))
          InsTesta(seconda, prima -> info);
        prima = prima -> next;
      }
}
```

ESERCIZIO 5 (6 punti)
Si scriva in `C` una funzione che, dati due array di interi `a` e `b` e le loro dimensioni `dima` e `dimb`, restituisce il valore

$$min\ I$$

dove

$$I = \{i \mid i \in [0, dima) \wedge \#\{j \mid j \in [0, dimb) \ \wedge \ a[i] = b[j]\}\ mod\ 2 = 0\}$$

se I non è vuoto; restituitsce $dima$ se I è vuoto.
Si ricorda che:

- dato un insieme finito A, $\#A$ indica il numero di elementi di A
- dato un insieme finito non vuoto di numeri S, $min\ S$ indica il valore minimo dell'insieme.

Soluzione

```
int val (int a[], int b[], int dima, int dimb)
{
   int i=0, trovato_pari = 0;
   while (i < dima & !trovato_pari)
     {
       int conta = 0;
       int j;
       for (j=0, j<dimb, j++)
         if (a[i]==b[j]) conta=conta+1;
       if (conta mod 2 == 0) trovato_pari = 1;
     }
   return i;
}
```