PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2014-2015
Prova scritta del 3/06/2015

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

$$\mathcal{L} = \{a^n b^m c^k \mid n > m + k \geq 1\}$$

sull'alfabeto $\Lambda = \{a, b, c\}$ non è regolare.

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

$$w = a^{n+2} b^{n+1}$$

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 $z = a^{(n+2-h-t)} b^{n+1}$.
La stringa $xy^0 z = a^h a^{(n+2-h-t)} b^{n+1} = a^{n+2-t} b^{n+1} \notin \mathcal{L}$ poiché $t > 0 \Rightarrow n + 2 - t \not> n + 1$.

ESERCIZIO 2 (6 punti)
Si supponga di disporre di una griglia 2×2 e di una pedina sulla griglia. La pedina si può muovere in verticale verso l'alto ($\uparrow$) o verso il basso ($\downarrow$), oppure in orizzontale verso sinistra ($\leftarrow$) o verso destra, ($\rightarrow$). La pedina non può mai uscire dai limiti della griglia.
Si dia una grammatica sull'alfabeto $\Lambda = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ in modo che le stringhe generate siano tutte e sole quelle che rappresentano sequenze di movimenti che portano la pedina dalla prima casella in basso a sinistra alla stessa casella (al termine della sequenza).
La grammatica deve generare, ad esempio, le sequenze $\uparrow\rightarrow\leftarrow\rightarrow\downarrow\leftarrow$ e $\rightarrow\uparrow\downarrow\leftarrow\uparrow\downarrow$, ma non deve generare, ad esempio, le sequenze $\uparrow\rightarrow\leftarrow\rightarrow$ e $\rightarrow\uparrow\downarrow$.

Soluzione
$S ::= \epsilon \mid \uparrow A \mid \rightarrow C$
$A ::= \rightarrow B \mid \downarrow S$
$B ::= \downarrow C \mid \leftarrow A$
$C ::= \leftarrow S \mid \uparrow B$

Una semplice variante che non genera la sequenza vuota:

$S ::= \uparrow A \mid \rightarrow C$
$A ::= \downarrow \mid \rightarrow B \mid \downarrow S$
$B ::= \downarrow C \mid \leftarrow A$
$C ::= \leftarrow \mid \leftarrow S \mid \uparrow B$

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

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

in modo che `(split xxs)` restituisca la coppia `(ys, zs)` in cui

- se `xxs` contiene almeno una occorrenza della lista vuota, `ys` è la concatenzione delle liste in `xxs` che precedono l'ultima occorrenza della lista vuota e `zs` è la concatenzione delle liste in `xxs` che seguono l'ultima occorrenza della lista vuota;
- se `xxs` non contiene la lista vuota, `ys=[]` e `zs=xxs`.

Soluzione

```
let split xxs =
      let f xs (ys,zs,b) = if b then (xs@ys, zs, b)
                                else if xs=[]
                                     then (ys,zs,true)
                                     else (ys, xs@zs,b)
      in
          let (ys, zs, _) = foldr f ([],[],false) xxs
          in (ys, zs);;
```

ESERCIZIO 4 (6 punti)
Si scriva in C una procedura che, dati attraverso opportuni parametri una lista di interi e un intero $n \geq 0$, elimina dalla lista i primi n elementi positivi. Si suppongano predefiniti i tipi

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

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

Soluzione

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

ESERCIZIO 5 (6 punti)
Dato un intero x e un array di interi v di dimensione dim, denotiamo con $x \circledcirc v$ la seguente formula

$$x \circledcirc v \equiv (\exists i.\ 0 \leq i < dim \wedge x = v[i])$$

Si scriva una funzione `C`

```
int check (int a[], int b[], int dima, int dimb)
```

che, dati due array `a` e `b` e le loro dimensioni `dima` e `dimb`, restituisce il valore di verità della seguente formula

$$\exists i \in [0, \texttt{dima}).\ \big((\forall j \in [0, i).\ \texttt{a}[j] \circledcirc \texttt{b}) \wedge \neg(\exists j \in [i, \texttt{dima}).\ \texttt{a}[j] \circledcirc \texttt{b})\big)$$

Soluzione
Definiamo dapprima una funzione `member` che realizza la proprietà $x \circledcirc v$, dati un intero x, un array v e la sua dimensione.

```
int member (int x, int v[], int dim)
{
   int i=0, trovato=0;
   while (i<dim && !trovato)
      if (x==v[i]) trovato=1; else i=i+1;
   return trovato;
}

int check (int a[], int b[], int dima, int dimb)
{
   int i=0, trovato=0;
   while (i<dima && !trovato)
      if (!member(a[i],b,dimb))
         trovato=1;
      else i=i+1;
   if (trovato)
      {
         i=i+1;
         trovato=0;
         while (i<dima && !trovato)
            if (member(a[i],b,dimb))
               trovato=1;
            else i=i+1;
      }
   return (!trovato);
}
```