PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2015/2016
Verifica scritta del 2/11/2015

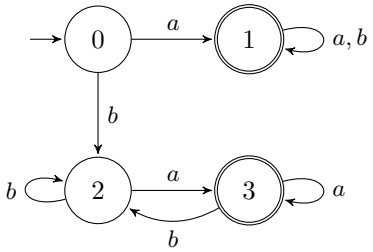
SOLUZIONI PROPOSTE

ESERCIZIO 1 (6 punti)
Definire un automa a stati finiti **deterministico** che riconosce il seguente linguaggio sull'alfabeto $\Lambda = \{\mathtt{a}, \mathtt{b}\}$.

$$\mathcal{L} = \left\{\alpha\mathtt{a} \mid \alpha \in \Lambda^*\right\} \cup \left\{\mathtt{a}\alpha \mid \alpha \in \Lambda^*\right\}$$

Definire quindi una grammatica **regolare** che genera il medesimo linguaggio.

Soluzione
Un semplice automa deterministico che riconosce il linguaggio è il seguente



Trasformiamo l'automa in una grammatica regolare che genera lo stesso linguaggio usando la tecnica vista a lezione.

```
<0> ::= a <1> | a | b <2>
<1> ::= a <1> | a | b <1> | b
<2> ::= b <2> | a <3> | a
<3> ::= b <2> | a <3> | a
```

ESERCIZIO 2 (6 punti)
Dato l'alfabeto $\Lambda = \{\mathtt{a}, \mathtt{b}\}$, definire una grammatica libera che genera il linguaggio

$$\mathcal{L} = \left\{\mathtt{aba}^n\mathtt{b}^m\mathtt{a}^n \mid n, m > 0\right\}$$

Soluzione

```
S ::= abA
A ::= aAa | aBa
B ::= bB | b
```

ESERCIZIO 3 (6 punti)
Dimostrare formalmente che il linguaggio dell'esercizio precedente **non è regolare**.

Soluzione
Per dimostrare che il linguaggio non è regolare utilizziamo il pumping lemma mostrando 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 $k \in \mathbb{N}$ e consideriamo, tra le tante possibili, la stringa

$$w = aba^{k+1}ba^{k+1}$$

Chiaramente $w \in \mathcal{L}$ e $|\ w\ | = 2k + 5 \geq k$. Siano dunque x, y, z tali che:

(a) $w = xyz$

(b) $y \neq \epsilon$

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

Consideriamo i seguenti due casi, che coprono tutte le possibili suddivisioni di w che soddisfano (a), (b) e (c).
(i) $x = \epsilon \vee x = a$. In questo caso, essendo $y \neq \epsilon$, la stringa xy^0z non appartiene ovviamente a $\mathcal{L}$ in quanto non è del tipo $ab\alpha$.

(ii) $x = aba^t$, $y = a^s$ e $z = a^{k+1-t-s}ba^{k+1}$, con $t + s \leq k - 2$ e $s > 0$. Anche in questo caso la stringa xy^0z non appartiene a $\mathcal{L}$. Infatti:

$$xy^0z = aba^t a^{k+1-t-s}ba^{k+1} = aba^{k+1-s}ba^{k+1}$$

e $s > 0 \Rightarrow k + 1 - s < k + 1$.

N.B. Negli esercizi che seguono non è consentito utilizzare comandi come `break` e `goto`, né utilizzare il comando `return` all'interno del corpo di un ciclo `while` o `for`.

ESERCIZIO 4 (6 punti)
Scrivere in `C` una funzione con intestazione

```
int check (int a[], int dim, int n, int m)
```

che, dati un array `a`, la sua dimensione `dim` e due interi `n` ed `m`, restituisce 1 se il valore massimo in `a` occorre nell'array più di `n` volte e meno di `m` volte; restituisce 0 altrimenti.

Soluzione
Utilizzando una iterazione determinata, calcoliamo il valore massimo dell'array e quante volte occorre nell'array stesso.

```
int check (int a[], int dim, int n, int m)
{
  int i; int massimo; int occ_massimo;
  massimo=a[0];
  occ_massimo = 1;

  for (i=1; i<dim; i++)
     if (a[i]>massimo)
         {  massimo = a[i];
            occ_massimo = 1;
         }
     else
         if (massimo == a[i]) occ_massimo = occ_massimo+1;

  return (occ_massimo > n && occ_massimo < m);
}
```

ESERCIZIO 5 (6 punti)
Scrivere in `C` una funzione

`int maxocc (int a[], int dim)`

che dato un array `a` e la sua dimensione `dim`, restituisce il valore che occorre nell'array il maggior numero di volte. Nel caso siano più di uno i valori che occorrono il maggior numero di volte, la funzione restituisce quello che occorre per primo. Ad esempio, dato l'array

60	**20**	30	10	**20**	40	50	80

la funzione deve restituire il valore 20. Dato invece l'array

60	**20**	30	10	**20**	40	**60**	80

la funzione deve restituire il valore 60.

```
int maxocc (int a[], int dim)
{
  int i; int valore; int occ_valore;
  occ_valore = 0;

  for (i=0; i<dim; i++)
     { int j; int occorrenze;
       occorrenze = 1;
       for (j=i+1; j<dim; j++)
         if (a[i]==a[j]) occorrenze = occorrenze + 1;
       if (occorrenze > occ_valore) { valore = a[i]; occ_valore=occorrenze; }
     }
   return (valore);
}
```