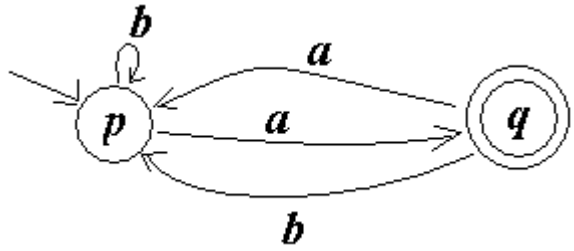
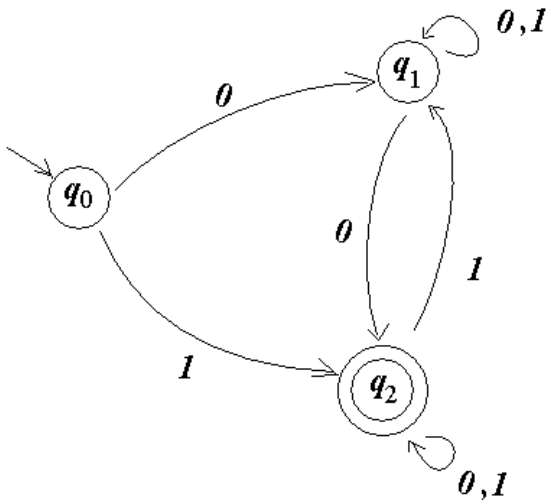
Automate finite

1.1 Probleme cu AF

1. Sa se reprezinte tabelar urmatoarele automate finite.
Sint ele AFD sau AFN?





2. Sa se reprezinte sub forma de graf automatul finit: M=(Q,Σ, δ,q_0,F)

	0	1	
q0	q2	q1	1
q1	q3	q0	0
q2	q0	q3	0
q3	q1	q2	0

Verificati apoi, bazandu-va pe graful obtinut, ca:
a) secventele 1010, 1100 sunt acceptate de automat
b) secventa 1011 nu este acceptata de automat

3. Sa se construiasca un AF care accepta

a) $L = \{ a\ a\ a \}$, $\Sigma = \{ a \}$

b) $L = \{ w_1\ a\ a\ a\ w_2 \mid w_1, w_2 \in \{a, b\}^* \}$

c) cuvinte peste alfabetul {0,1} cu proprietatea ca:
orice cuvant al limbajului contine cel putin 2 zerouri consecutive

d) cuvinte peste alfabetul {a, b, c} cu proprietatea ca:
primul simbol al cuvantului este acelasi cu cel cu care se termina cuvantul

e) cuvinte peste alfabetul {a, b, c} cu proprietatea ca:
exista un simbol in cuvant care mai apare cel putin o data in cuvant

f) $L = \{c^{3n}, n \in N^*\}$

g) limbajul ce contine secvente peste $\Sigma=\{a,b\}$ cu nr. par de simb. **a** si nr.par de simb **b**

h) limbajul ce contine secv. peste $\Sigma =\{a,b\}$ cu nr. impar de simb. **a** si impar de simb **b**

i) $L=\{ 1^n 0^m 1\ u \mid n>=0, m>=1, u \in \{0,1\}^* \}$

j) $L = \{ 0^n 1^m 0^q \mid n, m, q >= 1\}$

k) $L = \{ 0^n 1^m 0^q \mid n, m >= 1, q >= 0\}$

l) $L = \{ 0\ (10)^n 01^m \mid n>=0, m>=0\} \cup \{(10)^n 01^m \mid n>=1, m>=0\}$ cu cel mult 4 stari

m) $L=\{ 0^m 1^n \mid m,n \in N\} \cup \{1^p 0^q \mid p,q \in N\}$

n) $L = \{ w_1\ a\ a\ w_2 \mid w_1 \in \{b, ab\}^*, w_2 \in \{a, b\}^* \}$

1.2 Structuri de date pentru automate finite

Descrieti o modalitate de reprezentare pentru AF
(ganditi cateva modalitati)

ex:

- AF care are ca alfabet multimea caracterelor reprezentabile in calculator (alfabet fixat)
- structura de date care are in vedere ca operatia cea mai frecventa cu AF
  este verificare acceptare secventa

```
StateMachine
  description: String /** A description of the state machine */
  startState :State
  states: Set<State>  /** ?? can be List, Map, …; redundancy??!*/

State
  description: String  /** The name of the state */
  isAcceptState: Boolean
  transitions: MultiMap<Symbol, State> /** ?? can be List, Map */
```