Automat finit nedeterminist

Functia de tranzitie - variante

- matrice de stringuri

- Tranzitie obj (stare_plecare, symbol, stare_sosire) ex q0 0 q1, q2 0 q3, q7 0 $…

- tuple

- map / unordered_map

	a	b	c	
q0	q1, q4	q2	q3, q2	q0a , q1 q0, aq4
Q1	Q2	Q1	-	
Q2	q2	q1	q4	
Q3	q2	q1	q0	
Q4	q2	q1	q0	

{q0}, aababababaaac

{q1,q4}, abababababaaac

- **Backtracking** (testare toate variantele prin care pot ajunge, daca toate sunt false sau blocaj, atunci e neacceptat. Daca vreuna dintre ramuri este true, atunci este acceptat. Repet procesarea pana cand am terminat de verificat toate cazurile SAU pana cand am gasit o ramura cu true). – **AFN classic (5p)**
- **simulare lambda inchidere (3p)**

  {q1, q4}, **a**abababababaaac

  {q2}, bababababaaac

  {q1}, abababababaaac

  …

  {……} lambda

  Cand e acceptat? Trebuie sa contina una dintre starile finale! acceptat

Să se implementeze un AFN (automat finit nedeterminist) atfel: se citesc din fișier elementele componente ale AFN-ului Q, Σ, δ, q_0, F. Se testează cuvinte succesiv cuvinte.

1. Se cere creearea unei clase AFN (alta decât clasa principală – aviz celor cu Java, C#). În funcția principală main se declară un obiect de tip AFN, apoi se citesc pe rând cuvinte de la tastatură (într-un do – while) și pentru fiecare se verifică dacă e sau nu acceptat.

Membrii clasei vor fi: Stari, Sigma, Delta, StareInit, Finale

Printre metodele clasei trebuie să existe:

(1) afisare() - afișarea frumoasă a automatului (tabel sau grafic) **0.5p**
(2) verifica() – verificări legate de corectitudinea automatului (de exemplu: starea inițială și stările finale să existe in mulțimea de stări, Delta să fie definită pentru stări existente și caractere existente) **1p**

(3) accepta(cuvant) - verifică dacă cuvântul dat ca parametru este acceptat de către automat și afișează: "accepta" - dacă este cuvânt acceptat, "neacceptat" - dacă nu este accepta , "blocaj" - dacă automatul se blochează pe parcurs. (**3p** / **5p)**

Construcția corectă a clasei (membrii publici/privați):- **1p**

Citirea elementelor automatului se poate face printr-o funcție membră sau printr-o funcție externă, care sa preia ca parametru un automat.

2. Citirea din fișier a elementelor AFN-ului - **1p**

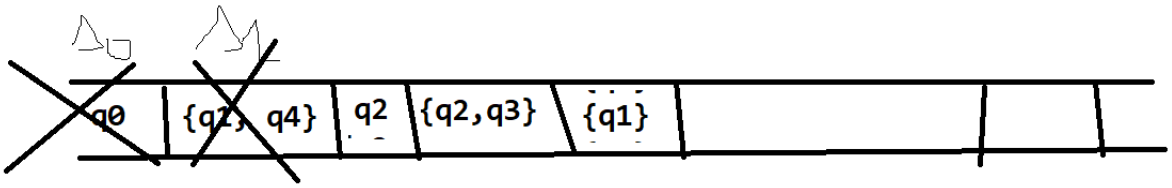
3. Posibilitatea de a verifica mai multe cuvinte, fără a reporni algoritmul (do-while) – **0.5p**

Un punct din oficiu

Un algoritm funcțional care doar citește și afișează elementele automatului - nota 3

Alternativa AFN -> AFN

- Teorema
- **Coada**



q0, a -> {q1, q4}
q0, b -> q2
q0, c ->{q2,q3}

{q1,q4}, a -> {q2}
{q1,q4}, b -> {q1}
{q1,q4}, c -> {q0}