بسم الله الرحمن الرحيم

International Islamic University Chittagong

Department of Computer Science and Engineering

B.Sc. in Computer Science & Engineering

Mid-Term Examination (Spring 2018)

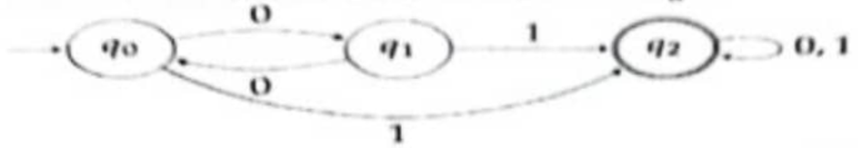
Course Code: CSE-3609 (Theory of Computing)

Time: 1:30 Hours | **Full marks: 30**

[Answer any 3 (three) of the following questions.]

1. a) What are the purposes of "theory of Computation"? Mention the formal definition of a finite automaton with proper example. **3**

   b) What will be the transition function of the following DFA? **3**



   Write down 1 string that will be accepted by this DFA and 1 other string that will not be accepted. What is the language of this DFA?

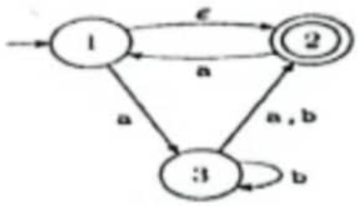
   c) Give state diagrams of DFAs recognizing the following languages. The alphabet is {0,1} **4**

   i) {w | w starts and ends with the same symbol} ii) { w | w ends with 01}

2. a) What are the differences between an NFA and a DFA? **3**

   b) Prove that every NFA can be converted to an equivalent one that has a single accept state. **3**

   c) **4**



   Convert the following NFA to DFA.

3. a) Write down regular expressions for the following languages: **3**

   (i) {w|w begins with a 1 and ends with a 0}

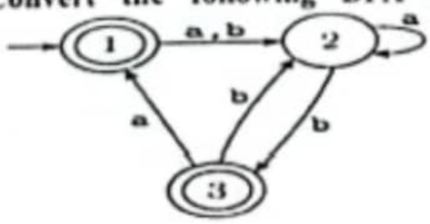
   (ii) {w|w contains at least one 1} (iii) { w|w contains at most two 0's}.

   b) Convert the following regular expressions to NFA: **3**

   (i) ab* U abb (ii) (a U b)*aba

   c) Define generalized nondeterministic finite automaton (GNFA). **4**

   Convert the following DFA to its equivalent regular expression.



4. a) Prove that "The class of regular languages is closed under the concatenation operation." **3**

   b) Describe the four components of a context free grammar. **3**

   c) Draw Turing machine for deciding language B = {anb2ncn | n > 0}. **4**

END

SPRING 2018

Mid term

Tahiya

1 (a)

The main Purpose ~~is~~ ~~to~~ ~~d~~ of theory of computation is to develop a formal mathematical model of Computation that reflects the real world computers.

A finite automation is a 5 tuple $(Q, \Sigma, \delta, q_0, F)$ where,

Q: Finite set of states

Σ: Finite set of alphabets

δ: $Q \times \Sigma$ transation ~~mod~~-table

q_0: $q_0 \in Q$ the starting state.

F: $F \subseteq Q$ Set of accepted states

Ex:-

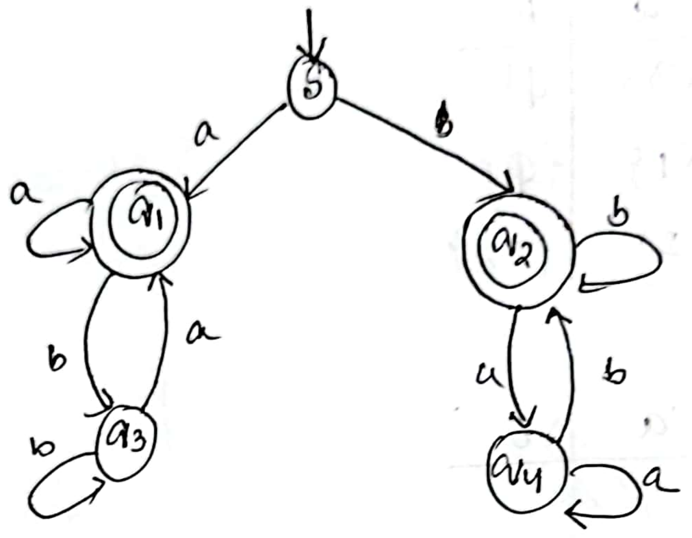
1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. $\delta =$

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

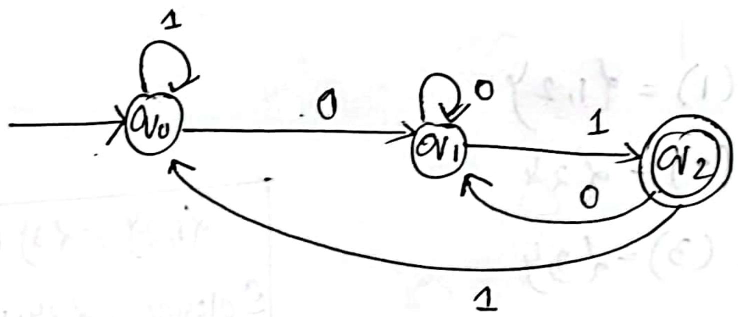
4. q_1 is the start state.
5. $F = \{q_2\}$.

1(c)

①



②



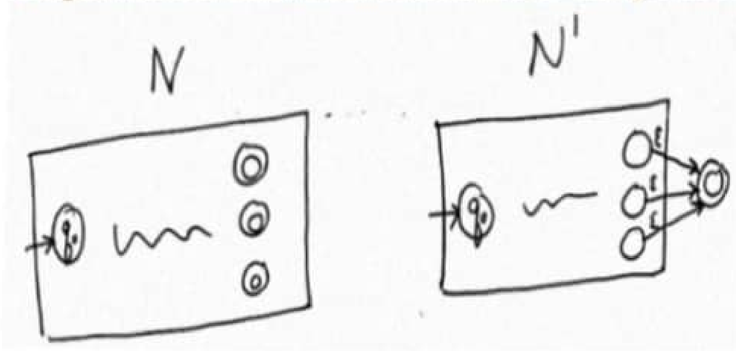
2(a)

Deterministic Finite Automata	Non-Deterministic Finite Automata
Each transition leads to exactly one state called as deterministic	A transition leads to a subset of states i.e. some transitions can be non-deterministic.
Accepts input if the last state is in Final	Accepts input if one of the last states is in Final.
Backtracking is allowed in DFA.	Backtracking is not always possible.
Requires more space.	Requires less space.
Empty string transitions are not seen in DFA.	Permits empty string transition.
For a given state, on a given input we reach a deterministic and unique state.	For a given state, on a given input we reach more than one state.
DFA is a subset of NFA.	Need to convert NFA to DFA in the design of a compiler.
$\delta : Q \times \Sigma \rightarrow Q$ For example – $\delta(q0,a)=\{q1\}$	$\delta : Q \times \Sigma \rightarrow 2^Q$ For example – $\delta(q0,a)=\{q1,q2\}$

2(b)

1.11 Prove that every NFA can be converted to an equivalent one that has a single accept state.

Plan: given an NFA N, convert it to the NFA N', which has a single state, as shown.



Proof: Let N= $(Q, \Sigma, \delta, q_0, F)$. Consider the NFA N'= $(Q\cup\{q_{accept}\}, \Sigma, \delta', q_0, \{q_{accept}\})$ where δ' is defined by:

$$\delta'(q,x) = \begin{cases} \delta(q, x) & \text{for } q\in Q\setminus F \text{ and } x\in \Sigma_\varepsilon \\ \delta(q, x) & \text{for } q\in F \text{ and } x\in \Sigma \\ \delta(q, x) \cup \{q_{accept}\} & \text{for } q\in F \text{ and } x=\varepsilon \\ \{\} & \text{for } q=q_{accept} \text{ and } x\in \Sigma_\varepsilon \end{cases}$$

Clearly N' recognizes the same language as N, yet N' only has a single accept state. Thus, given an arbitrary NFA, it can be converted to an equivalent one that has a single accept state.

2(e)

NFA

	a	b
1	{3}	{φ}
2	{1}	{φ}
3	{2}	{2, 3}

DFA

	~~a~~	~~b~~
~~1~~		

ε closure (1) = {1, 2}

ε closure (2) = {2}

ε closure (3) = {3}

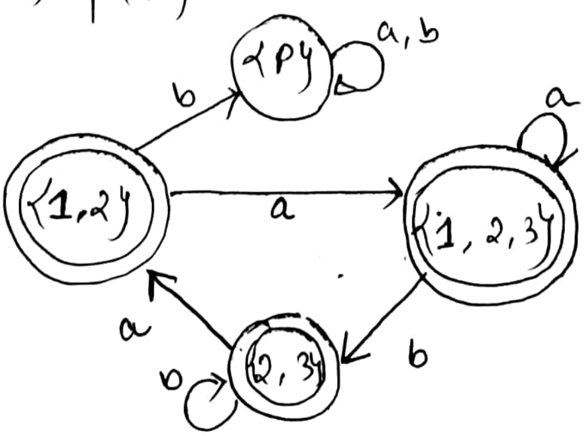
a	a	b
{1, 2}	{1, 2, 3}	{P}
{1, 2, 3}	{1, 2, 3}	{2, 3}
{2, 3}	{1, 2, 3}	{2, 3}
{P}	{P}	{P}

$\{1,2\} = \{3\} \cup \{1\}$

ε closure $= \{3\} \cup \{1\}$

$\Rightarrow \{3\} \cup \{1,2\}$

$\Rightarrow \{1,2,3\}$



3(a)

① $R = 1(0+1)^*0$

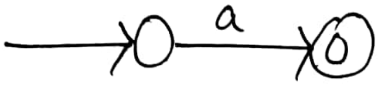
② $R = 0^* 1(011)^*$

③ $R = 1^* . 1^*01^*01^*$

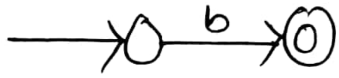
3(b)

① $ab^* \cup abb$

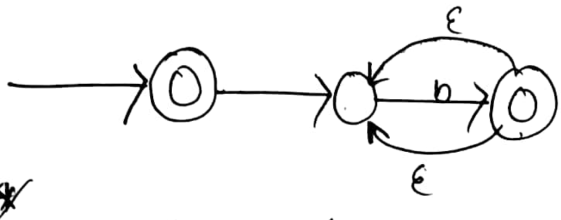
a



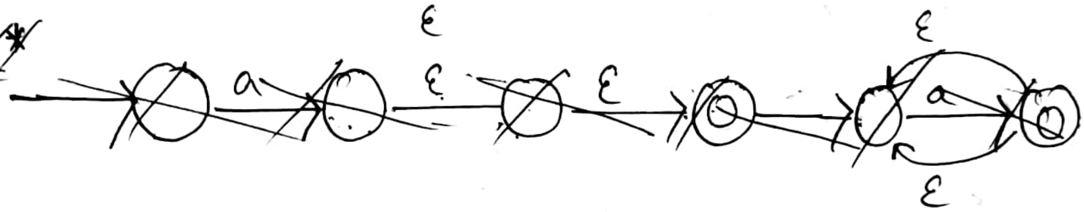
b



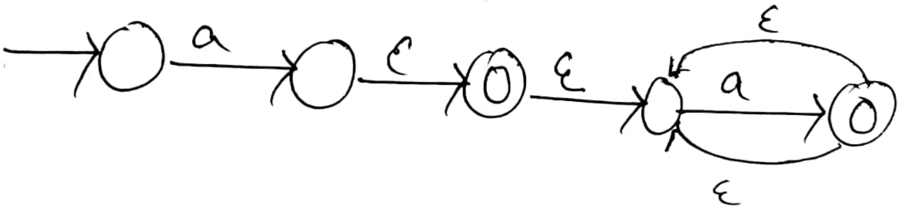
b*



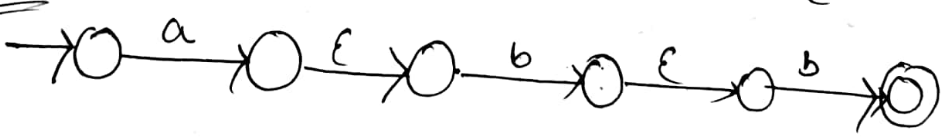
~~ab*~~



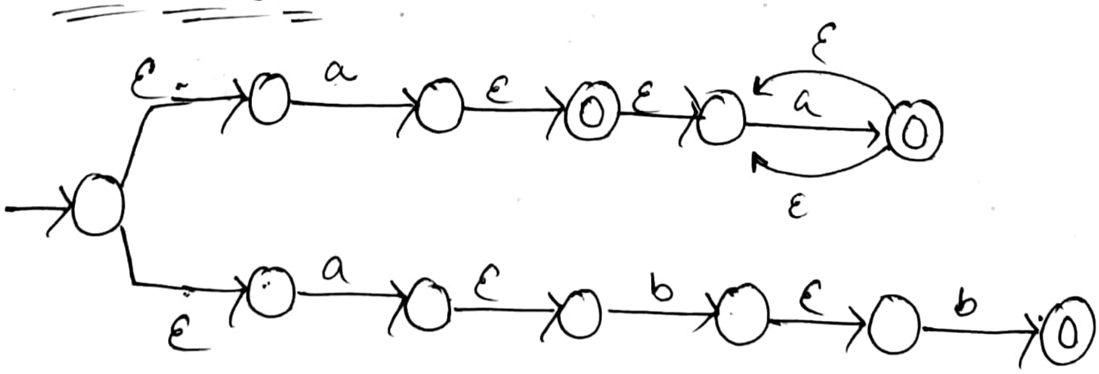
ab*



abb



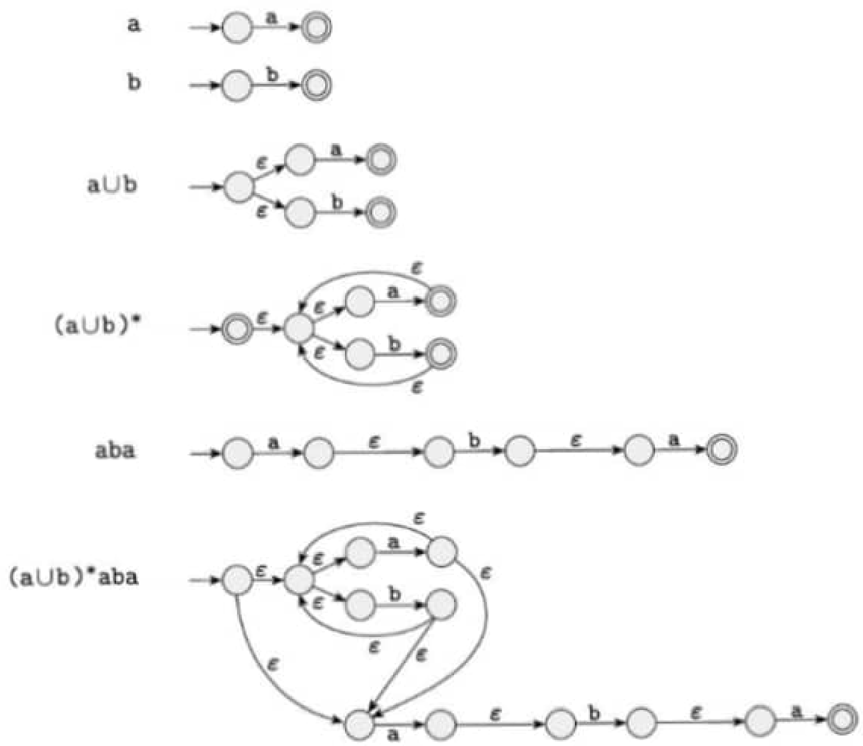
$ab^* \cup abb$



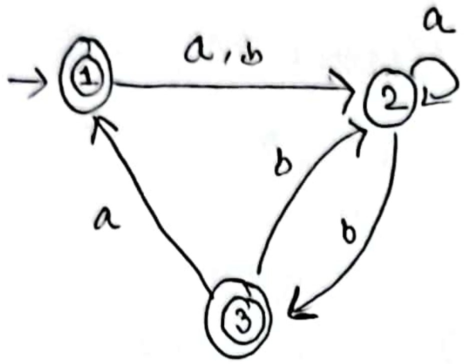
3(b)

EXAMPLE 1.58

In Figure 1.59, we convert the regular expression $(a \cup b)^*aba$ to an NFA. A few of the minor steps are not shown.

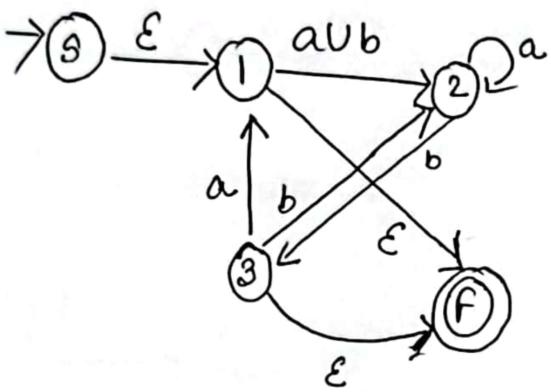


3 (e)



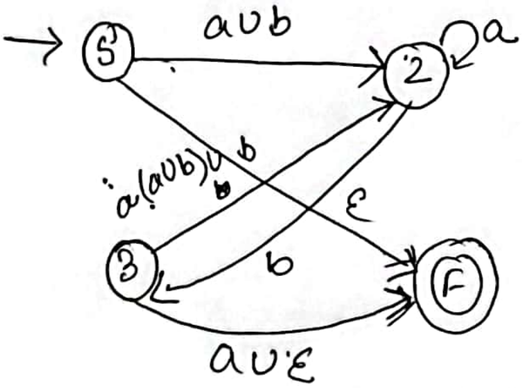
a, b
a
b
a
b
1
2
3

1)



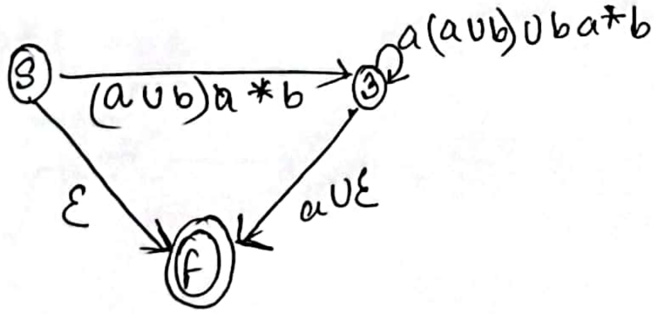
S
ε
1
a∪b
2
a
b
a
b
ε
3
F
ε

2)



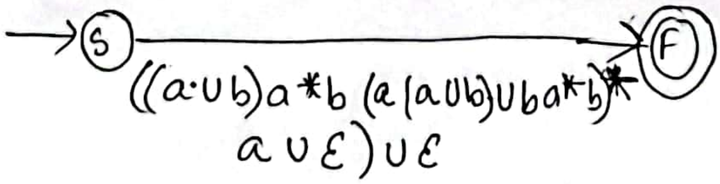
S
a∪b
2
a
a(a∪b)∪b
ε
3
b
F
a∪ε

3)



S
(a∪b)a*b
3
a(a∪b)∪ba*b
ε
a∪ε
F

4)



S
F
((a∪b)a*b (a(a∪b)∪ba*b)*
a∪ε)∪ε

DEFINITION 1.64

A ***generalized nondeterministic finite automaton*** is a 5-tuple, $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where

1. Q is the finite set of states,
2. Σ is the input alphabet,
3. $\delta \colon (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathcal{R}$ is the transition function,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

4(a)

\# The class of regular languages is closed under the concatenation operation.

Let

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1 and

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2

Constructing $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$

2. The state q_1 is the same as the start state of N_1.

3. The accept states F_2 are the same as the accept states of N_2.

4. Def^n of δ: $q \in Q$ and any $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

4(b)

A context-free grammar has four components:

A set of non-terminals (V). Non-terminals are syntactic variables that denote sets of strings. The non-terminals define sets of strings that help define the language generated by the grammar.

A set of tokens, known as terminal symbols (Σ). Terminals are the basic symbols from which strings are formed.

A set of productions (P). The productions of a grammar specify the manner in which the terminals and non-terminals can be combined to form strings. Each production consists of a non-terminal called the left side of the production, an arrow, and a sequence of tokens and/or on- terminals, called the right side of the production.

One of the non-terminals is designated as the start symbol (S); from where the production begins.