CISS362: Introduction to Automata Theory, Languages, and Computation
Test 1 Part A

The following instructions on defining a DFA or an NFA must be followed.

Here's an example on how to define an NFA:

```
automata:nfa
sigma:a,b
states:q0,q1,q2,q3,q4
start:q0
accept:q0,q1
transitions:
q0,a,q0
q0,b,q1
q1,e,q3
```

The letter `e` is used for ϵ. (None of the Σ in this test will use `e`.)

Here's an example on how to define a DFA:

```
automata:dfa
sigma:a,b
states:q0,q1
start:q0
accept:q1
transitions:
q0,a,q0
q0,b,q1
q1,a,q1
q1,b,q0
```

Q1. Our alphabet is $\Sigma = \{a, b, c\}$.

1. T or F or M: $1 + 1 = 2$
2. T or F or M: a is a regular expression
3. T or F or M: $a \cup b$ is a regular expression
4. T or F or M: $a \cdot \cup c$ is a regular expression
5. T or F or M: $a \cup^* b$ is a regular expression
6. T or F or M: $\{c\}$ is a regular expression
7. T or F or M: $c \cdot \emptyset$ is a regular expression
8. T or F or M: $\epsilon \cdot \epsilon \cdot \epsilon$ is a regular expression
9. T or F or M: $\emptyset^*$ is a regular expression
10. T or F or M: $a^*)$ is a regular expression
11. T or F or M: $a \cdot b \cup c$ is a regular expression
12. T or F or M: a^b is a regular expression
13. T or F or M: $a \in L(a \cup b)$
14. T or F or M: $ab \in L(a^* \cup b^*)$
15. T or F or M: $ab \in L((a \cup b)^*)$
16. T or F or M: $a \in L(a \cdot \emptyset)$
17. T or F or M: $ab \in L(a \cdot (a \cup b) \cdot c^*)$
18. T or F or M: $ab \in L((a \cup b) \cdot (b \cup c))$
19. T or F or M: $ab \in L((a \cup \overline{b}))$
20. T or F or M: $a^4b^2 \in L(a^* \cup b)L(a \cup b^*)$

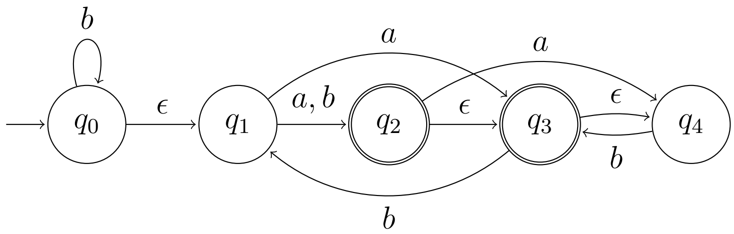
SOLUTION ON NEXT PAGE ...

SOLUTION.

Modify the file `q01.tex`. Use the letter `t` or `f` or `m`. I have already completed the first question for you.

```
1:t
2:t
3:t
4:m
5:m
6:f
7:t
8:t
9:t
10:m
11:t
12:m
13:t
14:f
15:t
16:f
17:t
18:t
19:m
20:t
```

Q2. For the NFA N given below, using the subset construction, construct a DFA M that accepts the same language accepted by N. Do not include states which are not reachable from the initial state of your DFA.



SOLUTION.

Modify the file `q02.tex`.

```
automata: dfa

sigma: a, b

states: {q0, q1}, {q0, q1}, {q0, q1, q2, q3, q4}, {q2, q3, q4},
{q1, q3, q4}, {q4}, {} ,{q3, q4}, {q1, q2, q3, q4}

start: {q0, q1}

accept: {q0,q1,q2,q3,q4},{q2,q3,q4},{q1,q3,q4},{q1,q2,q3,q4},{q3,q4}

transitions:
{q0,q1},a,{q2,q3,q4}
{q0,q1},b,{q0,q1,q2,q3,q4}

{q0,q1,q2,q3,q4},a,{q2,q3,q4}
{q0,q1,q2,q3,q4},b,{q0,q1,q2,q3,q4}

{q2,q3,q4},a,{q4}
{q2,q3,q4},b,{q1,q3,q4}

{q1,q3,q4},a,{q2,q3,q4}
{q1,q3,q4},b,{q1,q2,q3,q4}

{q4},a,{}
{q4},b,{q3,q4}

{},a,{}
{},b,{}

{q3,q4},a,{}
{q3,q4},b,{q1,q3,q4}

{q1,q2,q3,q4},a,{q2,q3,q4}
{q1,q2,q3,q4},b,{q1,q2,q3,q4}
```

Q3. Design an NFA that accepts $\{a, ab, bab\}^*$.

SOLUTION.

Modify the file `q03.tex`.

```
automata: nfa

sigma: a, b, e

states: q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10

start: q0

accept: q0, q6

transitions:
qo, e, q1
q0, e, q3
q0, e, q6

q1, a, q2

q2, e, q10

q3, a, q4

q4, b, q5

q5, e, q10

q6, b, q7

q7, a, q8

q8, b, q9

q9, e, q10

q10, e, q0
```

Q4. Recall that the "complement construction" works for a DFA, i.e., if you exchange

$$\text{accept} \leftrightarrow \text{non-accept states}$$

the resulting DFA will accept the complement of the language accepting by the original DFA.

Does it work with NFAs? In other words, if you exchange

$$\text{accept} \leftrightarrow \text{non-accept states}$$

for an NFA, will the resulting NFA accept the complement of the language accepting by the original NFA? If it works, prove it. If it does not, provide a minimal counterexample. (Minimal in this case means the one with least number of states.)

SOLUTION.

Modify `q04.tex`.

```
False.

Let NFA accept nothing
automata: nfa
sigma: a, b
states: q0
start: q0
accept:
transition:

Then, the compliment of this NFA accepts the empty string (e)
automata: nfa
sigma: a, b
states: q0
start: q0
accept: q0
transition:

The DFA for the original language is
automata: dfa
sigma: a, b
states: q0
start: q0
accept:
transition:
q0, a, q0
q0, b, q0

The compliment of the DFA accepts a*b*
automata: dfa
sigma: a, b
states: q0
start: q0
accept: a0
transition:
q0, a, q0
q0, b, q0

Then the compliment of the NFA doesn't equal the compliment of the DFA,
and therefore this operation isn't the same for NFA as DFA
```