CS302- HOMEWORK 1

Elif Cemre DURGUT - 26493

Question 1)

Complement of NFA is not equal to the NFA that accepts the language L^c.
Therefore, I should first convert the NFA to DFA, then complement the graph.

Table for NFA:

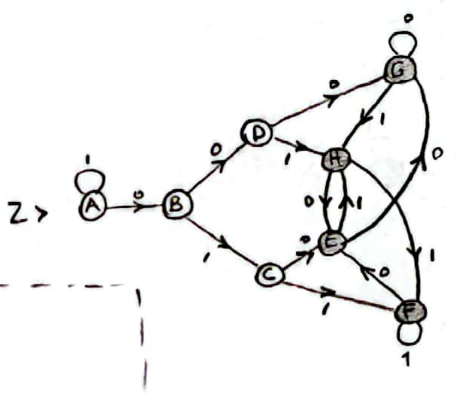
	0	1
$\rightarrow q_0$	q_0, q_1	q_0
q_1	q_2	q_2
q_2	q_3	q_3
$* q_3$	q_3	q_3

Table for DFA:

	0	1
$\rightarrow q_0$	q_0q_1	q_0
q_0q_1	$q_0q_1q_2$	q_0q_2
q_0q_2	$q_0q_1q_3$	q_0q_3
$q_0q_1q_2$	$q_0q_1q_2q_3$	$q_0q_2q_3$
$* q_0q_1q_3$	$q_0q_1q_2q_3$	$q_0q_2q_3$
$* q_0q_3$	$q_0q_1q_3$	q_0q_3
$* q_0q_1q_2q_3$	$q_0q_1q_2q_3$	$q_0q_2q_3$
$* q_0q_2q_3$	$q_0q_1q_3$	q_0q_3

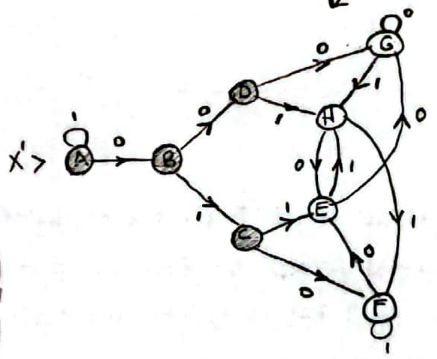
$\Rightarrow$

	0	1
.A	B	A
B	D	C
C	E	F
D	G	H
E	G	H
F	E	F
G	G	H
H	E	F



DFA diagram

Complement of DFA



//

2.2.5 a)

Number of possible such strings

two 0's $\rightarrow \frac{5!}{3!.2!} = 10$

three 0's $\rightarrow \frac{5!}{3!.2!} = 10$

four 0's $\rightarrow \frac{5!}{4!.1!} = 5$

five 0's $\rightarrow 1$

26

It will be hard to draw 26 possible cases.
But after drawing, it should loop back to the initial state to check other blocks.

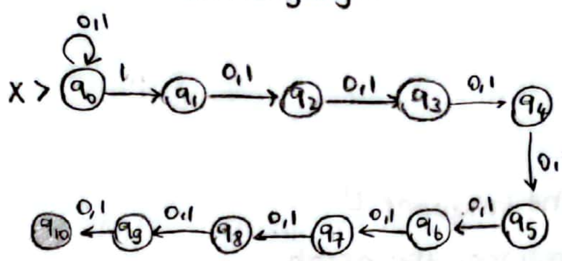
I assumed that
For ex:

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
0	0	1	1	1	1	1	1	0	0

we only check $s_1s_2s_3s_4s_5$ and $s_6s_7s_8s_9s_{10}$
but not $s_2s_3s_4s_5s_6$ because it doesn't say "any" in the question. $\rightarrow$ If we need to check those too, it will be even more complicated to draw!

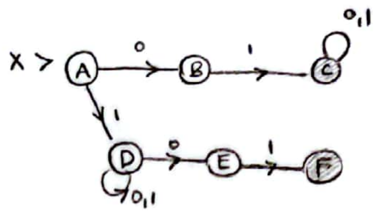
2.2.5

b) NFA of such language:



→ There are 10 states in NFA, so there are 2^{10} possible states for DFA. Therefore, it is very hard to draw DFA.

c) i) Let's start with NFA:



ii) Convert to DFA:

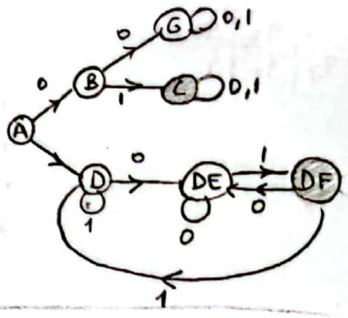
NFA Table:

	0	1
→ A	B	D
B	∅	C
* C	C	C
D	D,E	D
E	∅	F
* F	∅	∅

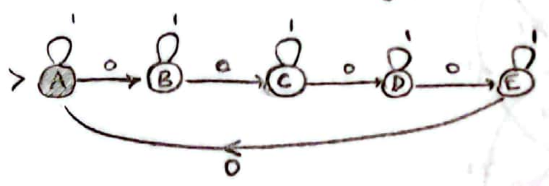
DFA Table:

	0	1
→ A	B	D
B	G	C
D	DE	D
G	G	G
* C	C	C
DE	DE	DF
* DF	DE	D

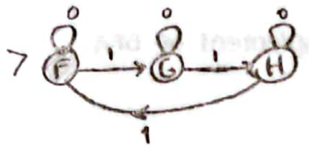
DFA Diagram:



d) The number of 0's divisible by five:



The number of 1's divisible by three:

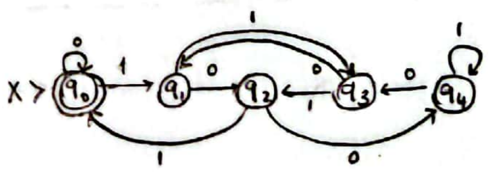


2.2.6 b)

For this, I can first calculate the DFA for left to right case and then reverse it.
The trick is to remember the remainder from the previous state. If the next bit is 0, this multiplies the number by 2; if the next bit is 1, this multiplies by 2 and add 1.

q_i : the state where the remainder is i

	0	1
→* q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4



⇒ I thought that it would work if I reverse this (not complement!). However, I tried and it does not work.

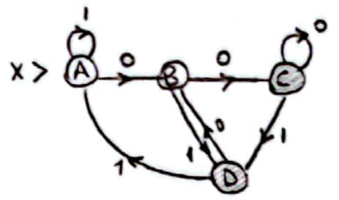
2.3.3

	0	1
→ p	pq	p
pq	pqrs	pt
* pqrs	pqrs	pt
* pt	pq	p

⇒

	0	1
→ A	B	A
B	C	D
* C	C	D
* D	B	A

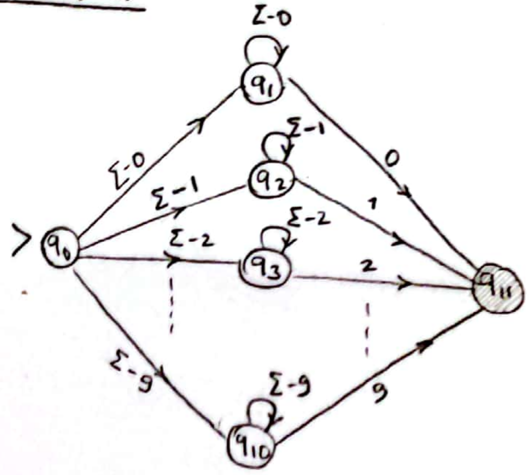
⇒



Informal definition:

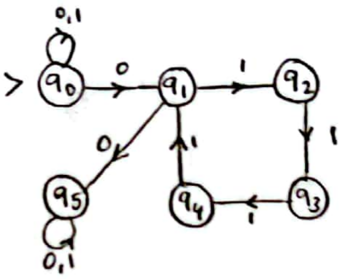
The string over the alphabet {0,1} and ends with 00.

2.3.4 b)



2.3.4 c)

It says "there are" in the question. Therefore, it is enough to find such one pattern:



2.5.2

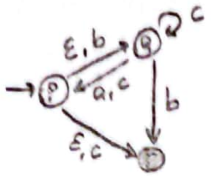
a)

ϵ-closure (p) = {p, q, r}

ϵ-closure (q) = {q}

ϵ closure (r) = {r}

b) strings of length three or less:



{b, c, bb, bcb, cb, abb, aab, acb, ab, ccb, cbb, cab, ∅}

c) ϵ-NFA to NFA:

NFA table:

	a	b	c
→ p	q, r	q, r	q, r
→ q	p	r	q
→* r	∅	∅	∅

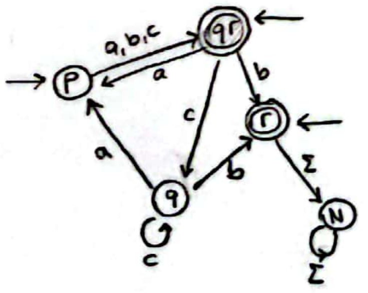
↓

p, q, r are all start states because of ϵ.

⇒ DFA table:

	a	b	c
→ p	qr	qr	qr
* qr	p	r	q
→* r	N	N	N
→ q	p	r	q
N	N	N	N

⇒ DFA diagram:



p
qr
r
q
N
a, b, c
a
b
c
Σ