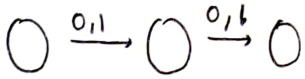
Assignment - 0 - COMPILER DESIGN - Revision of AFLL :-

PRIYA MOHATA
PE2UG19CS301
SECTION-E

(Q1) $(0+1) \cdots n$ times

States:-



} n+1 times.

∴ (ii) n+1 states correct option.

(Q2) Answer Regular, Non Regular

(Q3) String = 1101

(i) $110^*(0+1) \Rightarrow$ 11 any number of 0's (≥ 0) (0 or 1) $\Rightarrow 1101 \in 110^*(0+1)$

(ii) $(10)^*(01)^*(00+11)^* \Rightarrow 1101 \notin (10)^*(01)^*(00+11)^*$

(iii) $1(0+1)^*101 \Rightarrow 1101 \in 1(0+1)^*101$

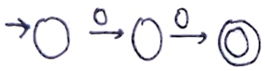
(iv) $(00+(11)^*01)^*)^* \Rightarrow 1101 \in (00+(11)^*01)^*$

∴ Answer ⇒ (ii)

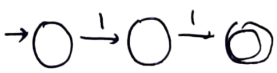
(Q4) Either ends with 00 or 11

(i) Explanation ⇒ If we go by traditional cartesion product method.

Ends with 00 ⇒

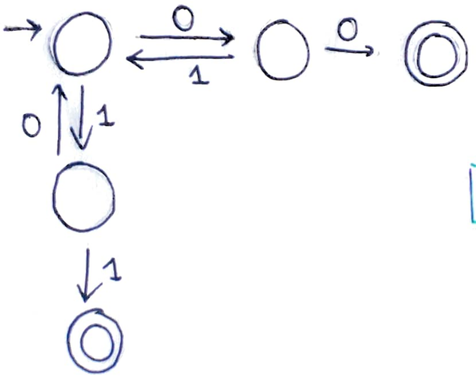


Ends with 11



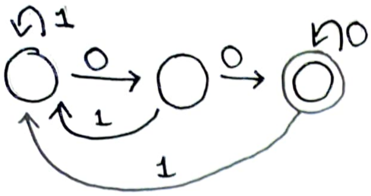
Max states = 9

Minimal states



∴ Answer ⇒ (ii) 5

(Q5)



Answer (ii) 3

(Q6) For strings :- $n_a(w) \bmod 4 = 0$

$n_b(w) \bmod 5 = 0$

For this ⇒ remainders are 0,1,2,3 ⇒ n=4

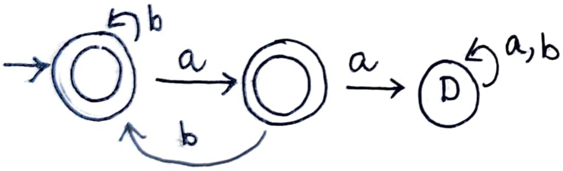
For this ⇒ remainders are 0,1,2,3,4 ⇒ m=5

∴ n * m = 20 ⇒ minimum number of states

∴ Answer (i) 20

(Q7) Answer (iii) For every NFA there is a DFA and vice versa.

(Q8) $L = \{\lambda, bbb\ldots, abba, aba, a, aba\ldots\}$



Answer (ii) 3

(Q9) (i) $(pq)^*$ ⇒ pq ⇒ belongs to this ×

(ii) $(qp)^*$ ⇒ ~~qppq~~ qpqp ⇒ ∴ $pq \in (qp)^*$ ×

(iii) $(p^* q^*)$ ⇒ pq ⇒ $pq \in p^* q^*$ ×

(iv) $q^* p^*$ ⇒ ∴ $pq \notin q^* p^*$

∴ Answer (iv)

(Q10) Ex: ~~of~~ a valid string = { 0011, 1100, 11000, 00110, 110100 } (4)

(i) $(0+1)^* 0011 (0+1)^* + (0+1)^* 1100 (0+1)^*$ ⇒ looks at two consecutive 1's and 0's

③

(1) 0011 ✓ (3) 11000 ✓ (5) 110100 ✗

(2) 1100 ✓ (4) 00110 ✓

↳ although its valid

these two also must be consecutive

∴ its wrong.

(ii) $(0+1)^* (00(0+1)^*) 11 + 11 (0+1)^* 00) (0+1)^*$

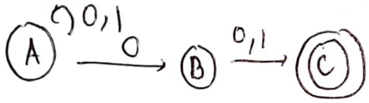
(1) 0011 ✓ (3) 11000 ✓ (5) 110100 ✓

(2) 1100 ✓ (4) 00110 ✓

∴ (B) is the answer choice

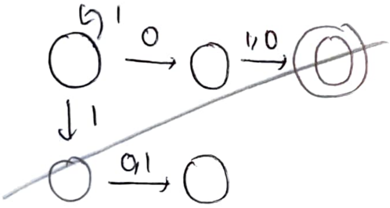
Answer (ii)

(Q11) NFA :-

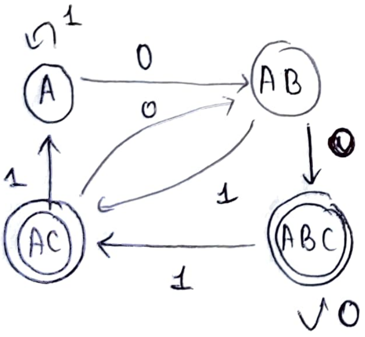


∴ Answer (ii) 4,3

DFA :-



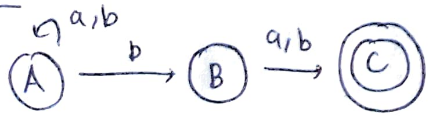
DFA:



(Q12) Answer: 4

(This question is similar to the previous one)

NFA :-



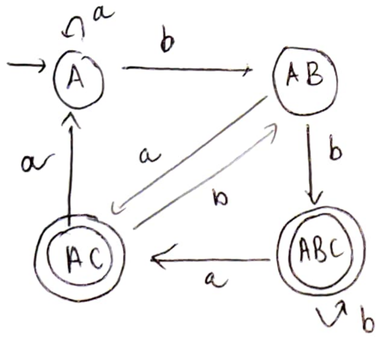
DFA: NFA table:-

	a	b
→ A	A	B, A
B	C	C
* C	Φ	Φ

DFA:

	a	b
→ A	A	A, B
A, B	A, C	A, B, C
* A, C	A	A, B
* A, B, C	A, C	A, B, C

DFA:



Q13 (a) $a(a+b)^* a$

(b) $L = \{a, aaa \dots\} \Rightarrow b^*(ab^*ab^*)^* ab^*$

~~$b^*(aa)^* a b^*$~~ X

(c) $L = \{ab, abab, baba \dots, bbaba \dots\}$
λ, bbb...

↳ $(b+ab)^*(a+\lambda) + (a+\lambda)(b+ba)^*$

Q14

(A) Decimal numbers:- (1.41, 0.41, 3.41 ....
1, 2, 3, ---
+1.41, -1.41 ..., .34..)

↳ ~~^\d+\.?\d* $~~

→ ^[+ -]?(\d+\.?\d* | \.\d+)$

number can be positive/negative

optional

cases like 1.41 → optional
↓
>1 digit

for cases like .34 ...

(B) Variable name in C;

↳ can't start with number
↳ only A-Z, a-z and underscore

^[A-Z a-z _]{1} [A-Z a-z 0-9 _]* $

first character

rest of the characters.

(C) Number in exponential form

L = { 1.54e-10 , 1.54e+10 , 1e10 ... }

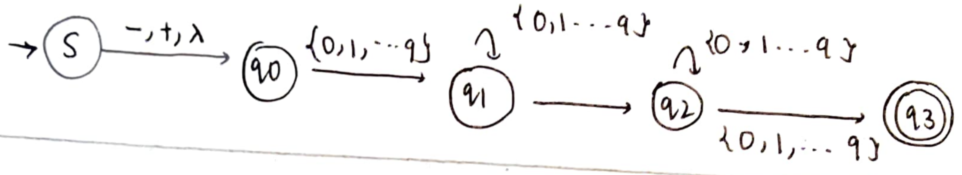
Regex ⇒ \d*\.?\d* e[+,-]?\d+

d.

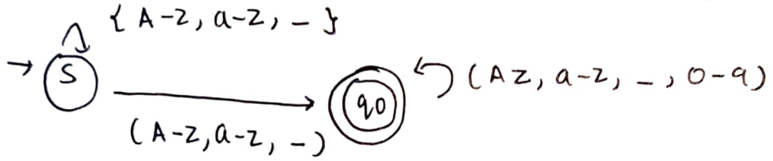
(D) A relational expression consisting of 1 operator.

Regex ⇒ (expression)[>< >=<=!](expression)

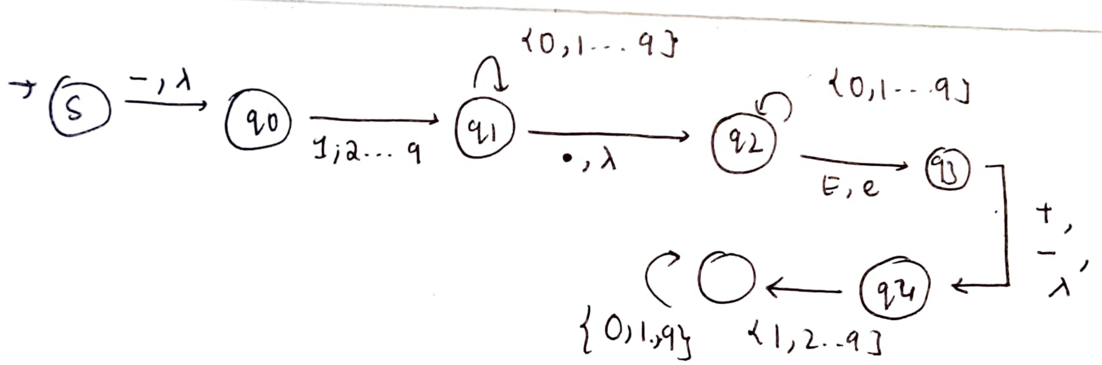
(15) (A)



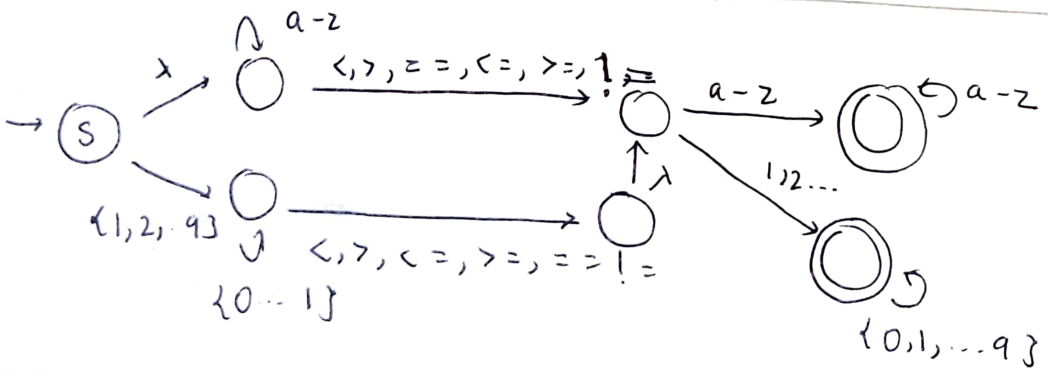
(B)



(C)



(D)



(416)

(A) Variable Declaration in C Language :-

↳ int a   or   int a, b, c . . .

type ↙ id ↓

a, b, c : list of id's. separated by a comma.

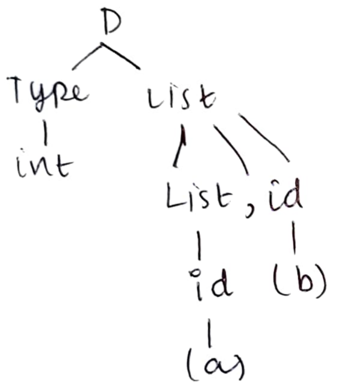
CFG:

$D \rightarrow Type\ List$

$List \rightarrow id.|\ List,\ id$

$Type \rightarrow int\ |\ float\ |\ double\ |\ char\ |\ short\ |\ boolean$

(B) Parse int a, b ;



(C) If-else loop generation :-

if condition then
    statement
else
    statement

S → if condn then S |
S ⟶ ~~else S~~ if condn then
    S else S | { statement }

S → if condn then else S | { statement }

(D) (a) S → S1 | S2
S1 → a S1 bb | S1 b | b
S2 → a S2 bb | a S2 | a

(b) S → aSbbb | S1
S1 → a bbb

(Q17)

Ambiguous Grammar

→ A grammar is ambiguous when there exists a string w that belongs to the grammar and there exists 2 different left most derivations or 2 right most derivations for the string (i.e two different parse trees)

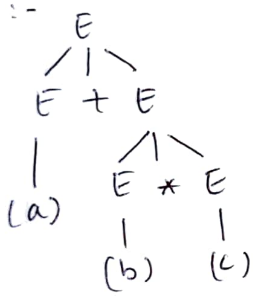
Ex (B)

Σ = { +, *, /, -, ( ), var, %, constant, ^ }

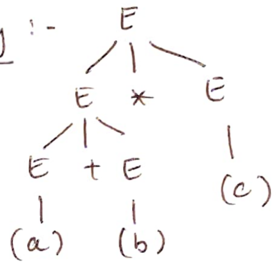
E → E+E | E * E | E / E | E - E | (E) | E % E | constant | var | E ^ E

Here w = a + b * c

One way :-



Second way :-



(A) Dangling-else problem :- ↗ if condn then ~~statement~~ {if condn then statement else statement

↳ **Grammar:**

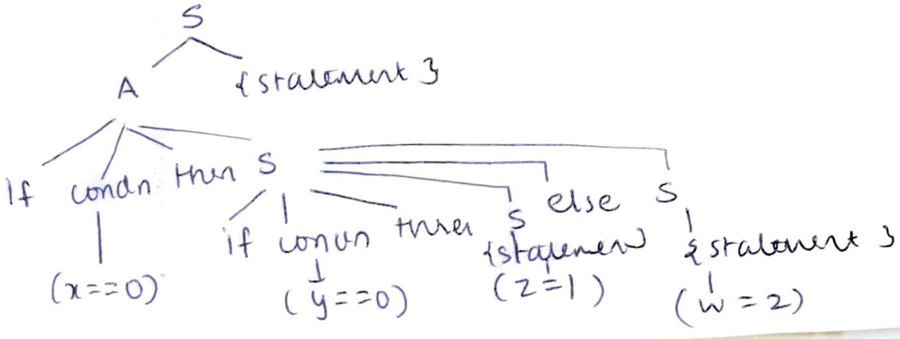
~~S → if condn then S | if condn {statement }~~

S → A | {statement }
A → if condn then S | if condn then S else S

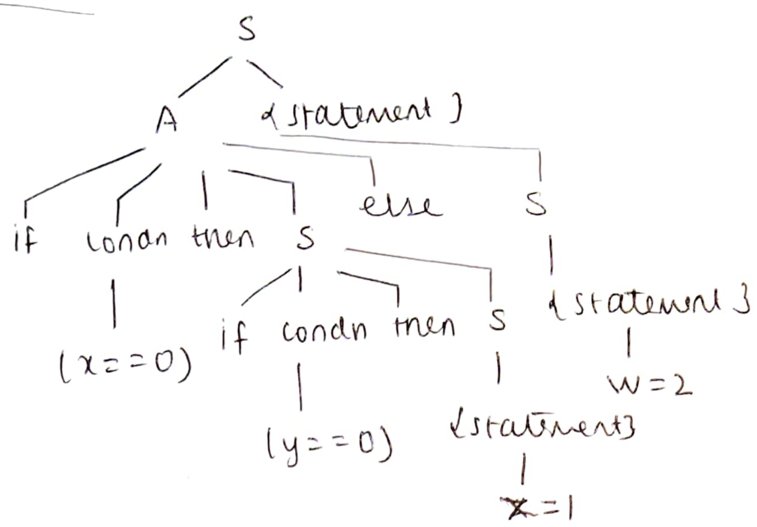
Ex:

if x==0 then if y==0 then z=1 else w=2

Parse Tree-1



Parse Tree-2



(Q18) What is a sentence? What is a sentential form?

Ex: ~~S → SS~~

Ex: $S \to aSb \mid bSa \mid SS \mid \lambda$ $w = abba$

$S \to SS$
$S \to aSbS$
~~S → aS~~ $S \to abS$
$S \to abbSa$
$\boxed{S \to abba}$

These are sentential form.

↳ this is sentence

Sentential form ⇒ is any string derivable from the start symbol.
Sentential form $\in (V \cup T)^*$

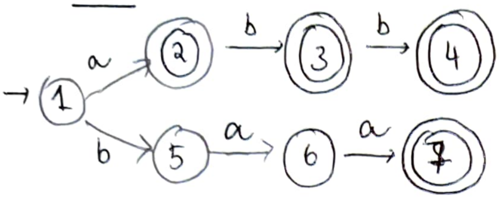
Sentence ⇒ is a sentential form consists only of terminals.

(Q19) What are the different parsing techniques?

Parsing Techniques:-

1. Top down parsing
2. Bottom up parsing
3. Universal parsing

(20) NFA :



Q = { 1, 2, 3, 4, 5, 6, 7 }

Σ = { a, b }

F = { 2, 3, 4, 7 }

q_0 = { 1 }

NFA transition table

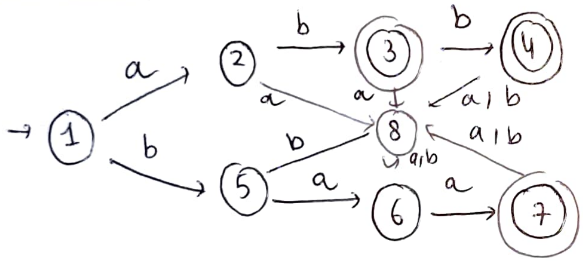
	a	b
1	2	5
2	Φ	3
3	Φ	4
4	Φ	Φ
5	6	Φ
6	7	Φ
7	Φ	Φ

State transition table for DFA

	a	b
1	2	5
2	8	3
3	8	4
4	8	8
5	6	8
6	7	8
7	8	8
8	8	8

State 8 ⇒ Dead state

Equivalent DFA :-



Method to convert NFA to DFA ⇒ Subset Construction Method

(21)

Context Free Grammar	Regular Expressions
- Lexical rules are difficult in case of context free grammar. - Notations are complex to understand. - Difficult to construct the recognizer.	- Lexical rules are quite simple in case of Regular Expressions. - Notations are easy to understand. - Used to construct efficient recognizer.

(22) A: ~~Unambgi~~ Unambiguous Grammar.

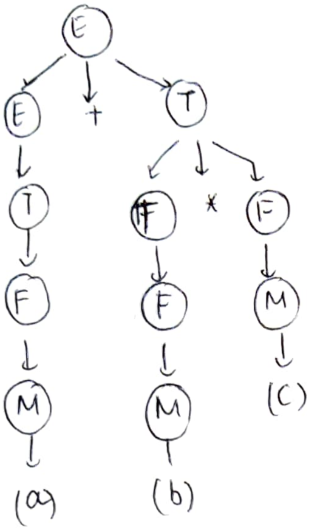
E → E + T | E - T | T

T → T * F | T / F | F

F → M ^ F | M

M → (E) | id | NUM

$w = a + b * c$



$E \overset{LM}{\Rightarrow} E + T$

$\Rightarrow T + T$

$\Rightarrow F + T$

$\Rightarrow M + T$

$\Rightarrow a + T$

$\Rightarrow a + T * F$

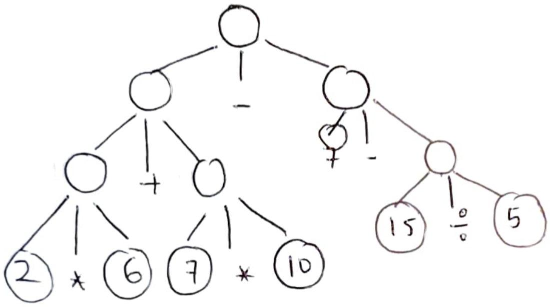
$\Rightarrow a + F * F$

$\Rightarrow a + \cancel{M} * F$

$\Rightarrow a + b * M$

$\Rightarrow a + b * c$

22(B)



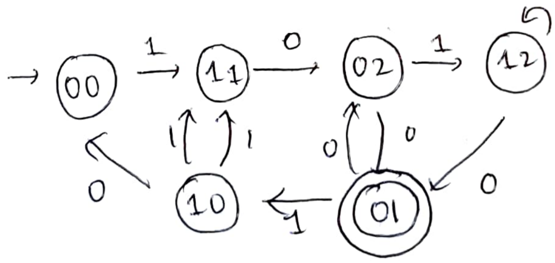
(23)

(A) ~~A~~ W mod 3 = 0 and W mod 2 = 1

$\div 3 \leq \begin{matrix}0\\1\\2\end{matrix}$ $\div 2 < \begin{matrix}0\\1\end{matrix}$ } total states = 6.

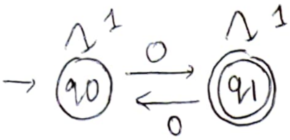
2	3	Binary	Decimal
0	0	0000	0
1	1	0001	1
0	2	0010	2
1	0	0011	3
0	1	0100	4
1	2	0101	5
0	0	0110	6
1	1	0111	7

Diagram :-

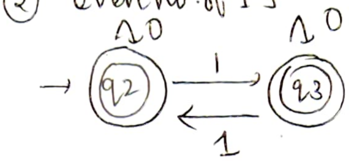


(23)(B)

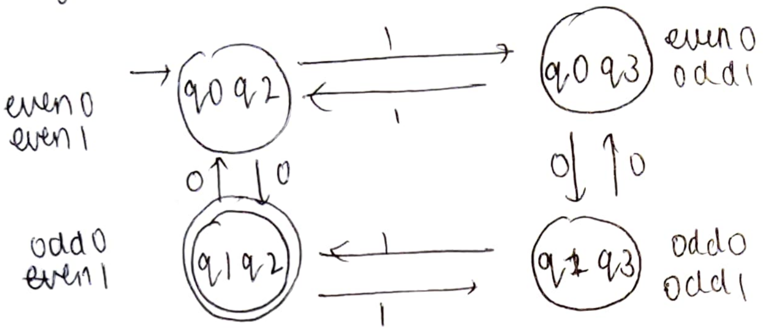
① ~~even~~ odd no. of 0's



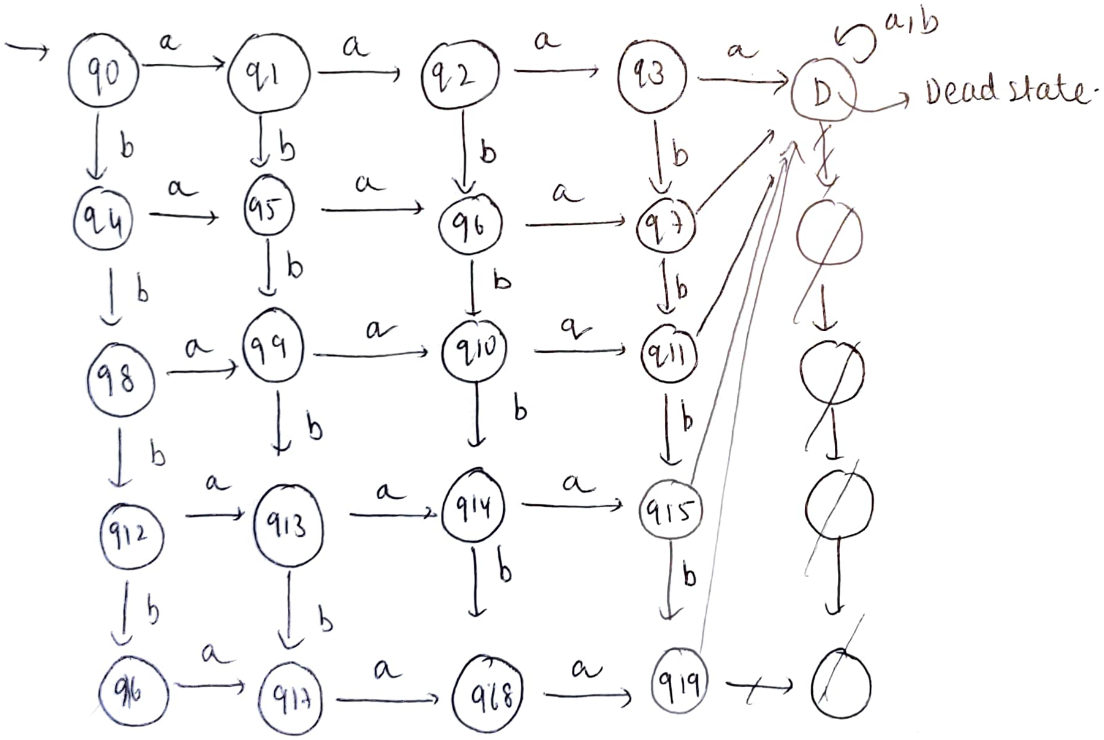
② even no. of 1's



Merge



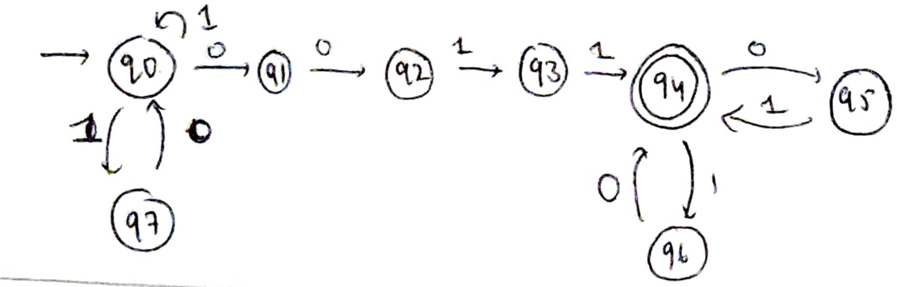
(23)(C) Exactly 3 a's and more than 4 b's



(Q24) $(10+1)^* \quad 00\phi1 \; (10+01)^*$

(a)

<u>NFA</u>



(b) $S \to 1S \mid 10S \mid 0A$

$A \to 0B$

$B \to 1C$

$C \to 1D \mid \lambda$

$D \to 01D \mid 10D \mid \lambda$

(a) 00110 = 14

(c) (b) 13 → 001101

(c) 00101 · 13

(Q5) (A) (a) 14 = 001110

$S \to ABC$

$\to BC$

$\to 0011C$

$\to 00110C$

$\to 00110$.

sentential forms

⇒ ABC

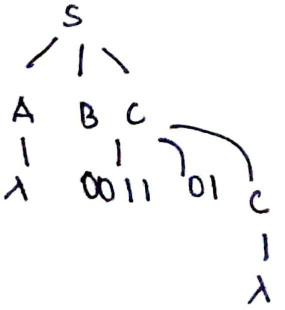
⇒ BC

⇒ 0011C

⇒ 00110C

⇒ 00110

b, c) 13 = 001101



S
A
B
C
λ
0011
01
C
λ