CS406: Compilers

HA-1

190010008

1) Regular Expression that matches valid roll number (aa0bbccc) =>

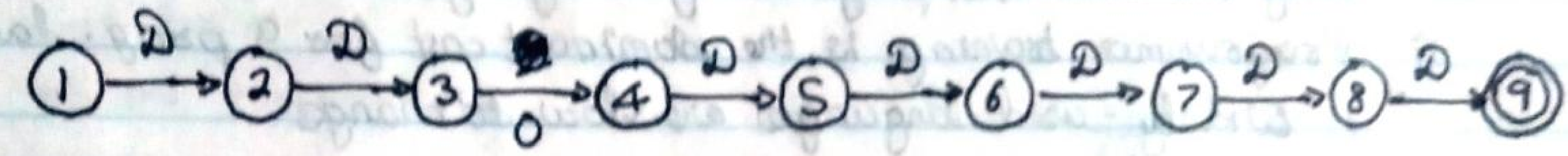
[0-9][0-9]0[0-9][0-9][0-9][0-9][0-9]

OR

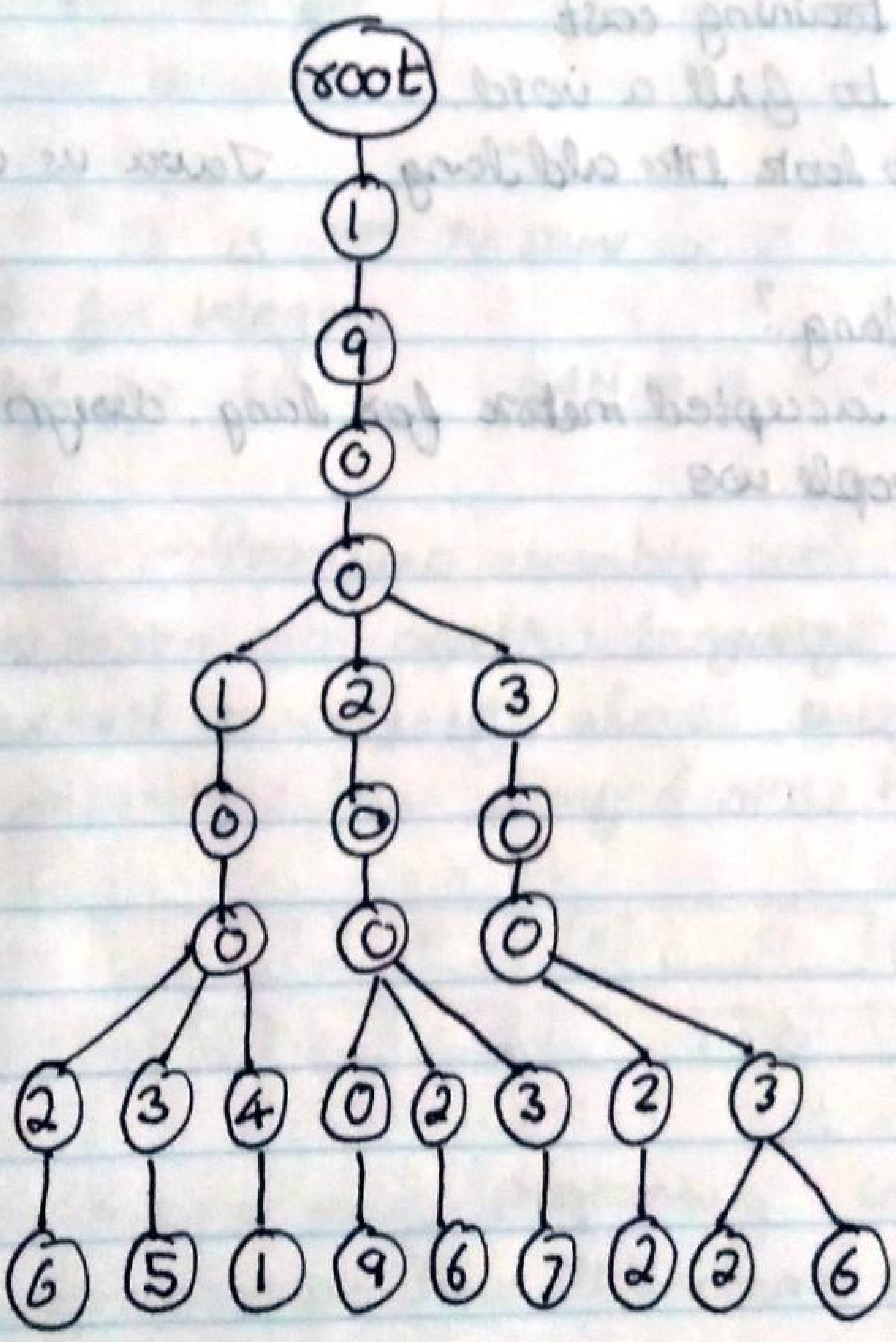
~~DDODDDDDD~~ where D = [0-9] ~~[illegible]~~

DDODDDDD

2)



6)

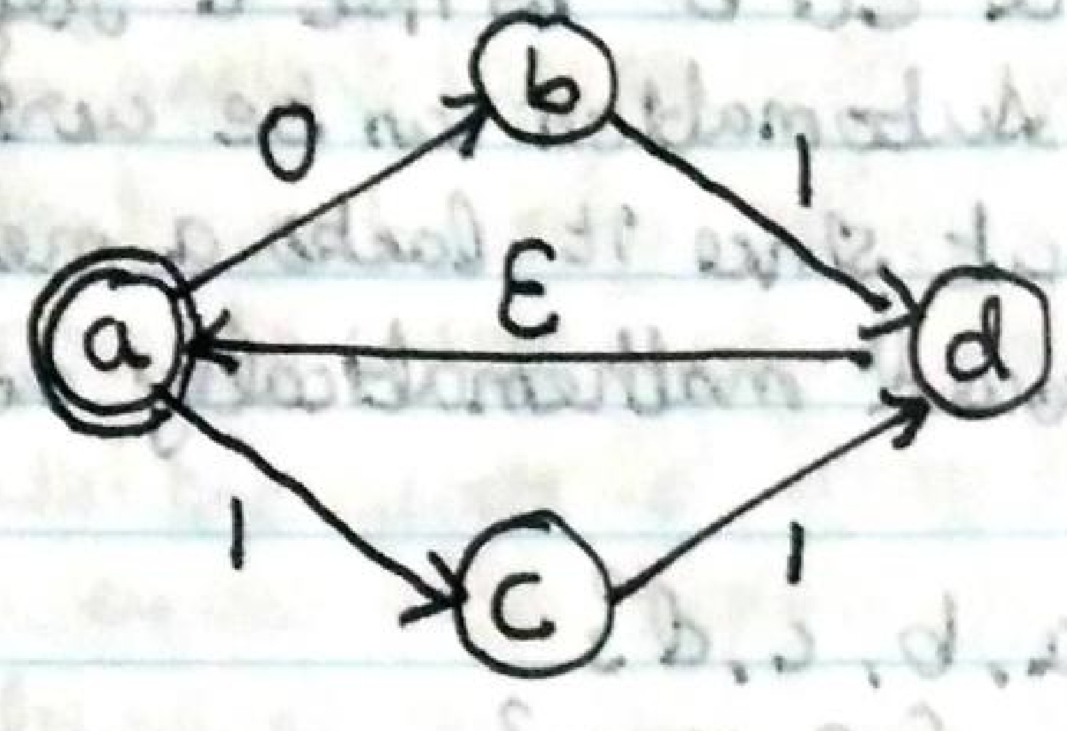


CSE	ECE	MMAE
190010026	190020009	190030022
190010035	190020026	190030032
190010041	190020037	190030036

3) Regular Expression => $((((01)+|(11)+)))^*$

Strings belonging to the above regex = ε, 01, 11, 0111, 0101, 1111, 010111, etc

NFA =>

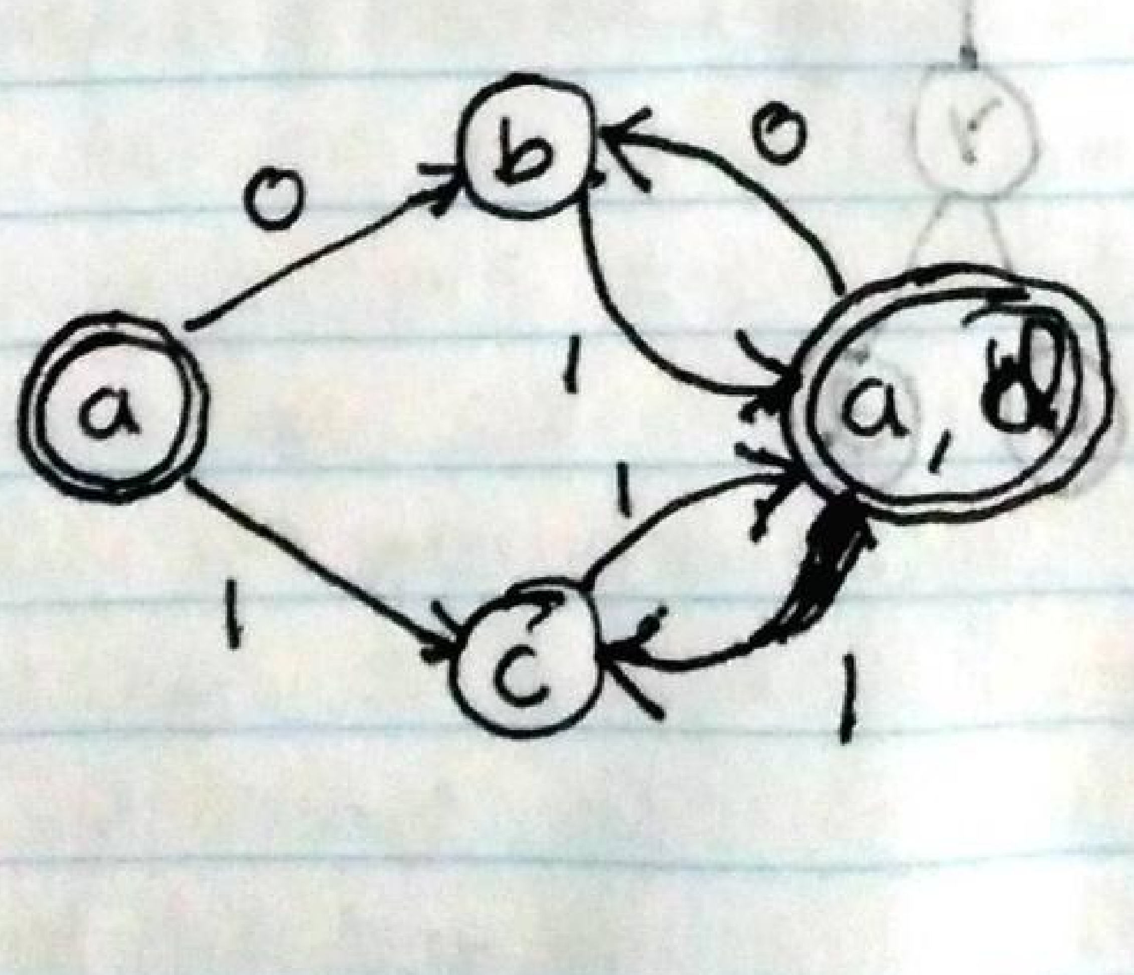


a is the initial as well as the final state

4) Transition Table =>

Character / State	ε	0	1
a	a	b	c
b	b	–	d
c	c	–	d
d	a,d	–	–
a,d	a,d	b	c

Equivalent DFA =>



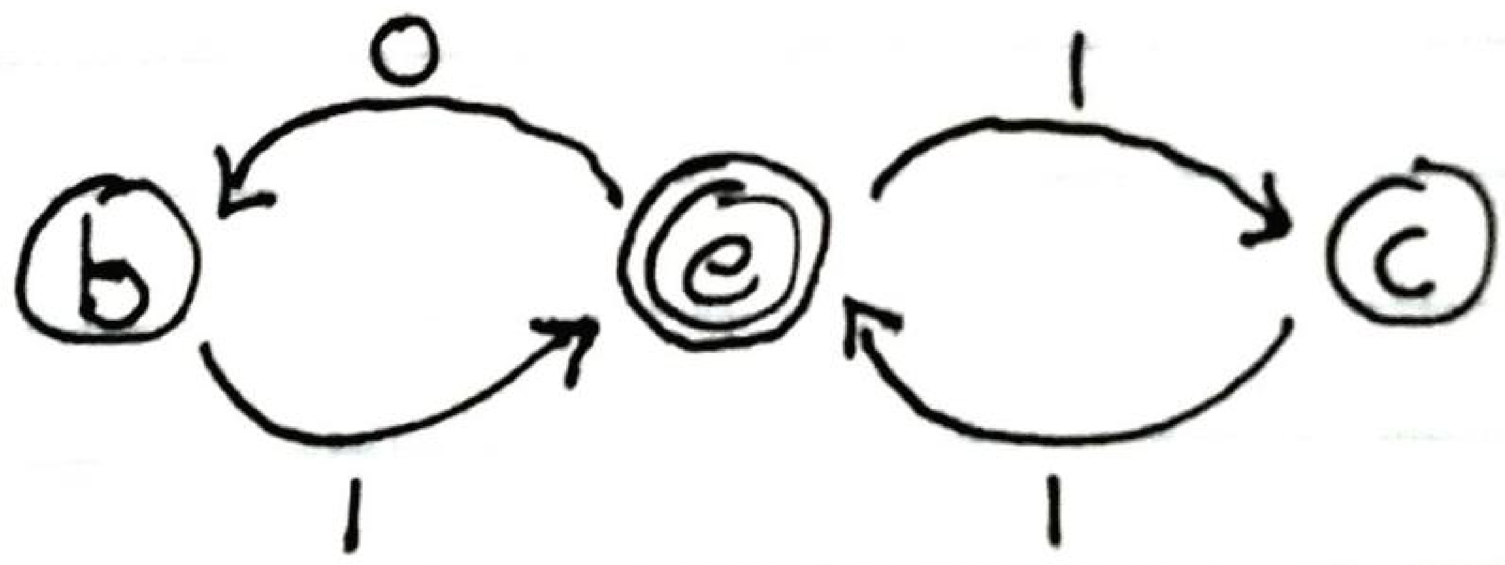
a is the initial state & (a) and (a,d) are the final state.

5) In order to convert DFA to reduced DFA, we look into the transition table to find which states could be merged together. We see that state (a) & (a,d) could be merged.

State Transition Table after merging -

State \ Character	ε	0	1
a-a,d-M	a_a,d_M	b	c
b	b	-	a_a,d_M
c	c	-	a_a,d_M

Lets rename the state a_a,d_M as state e



e is the initial as well as the final state

7) Strings with even number of characters having n number of a's followed by cd followed by n number of b's where $n \geq 0$.

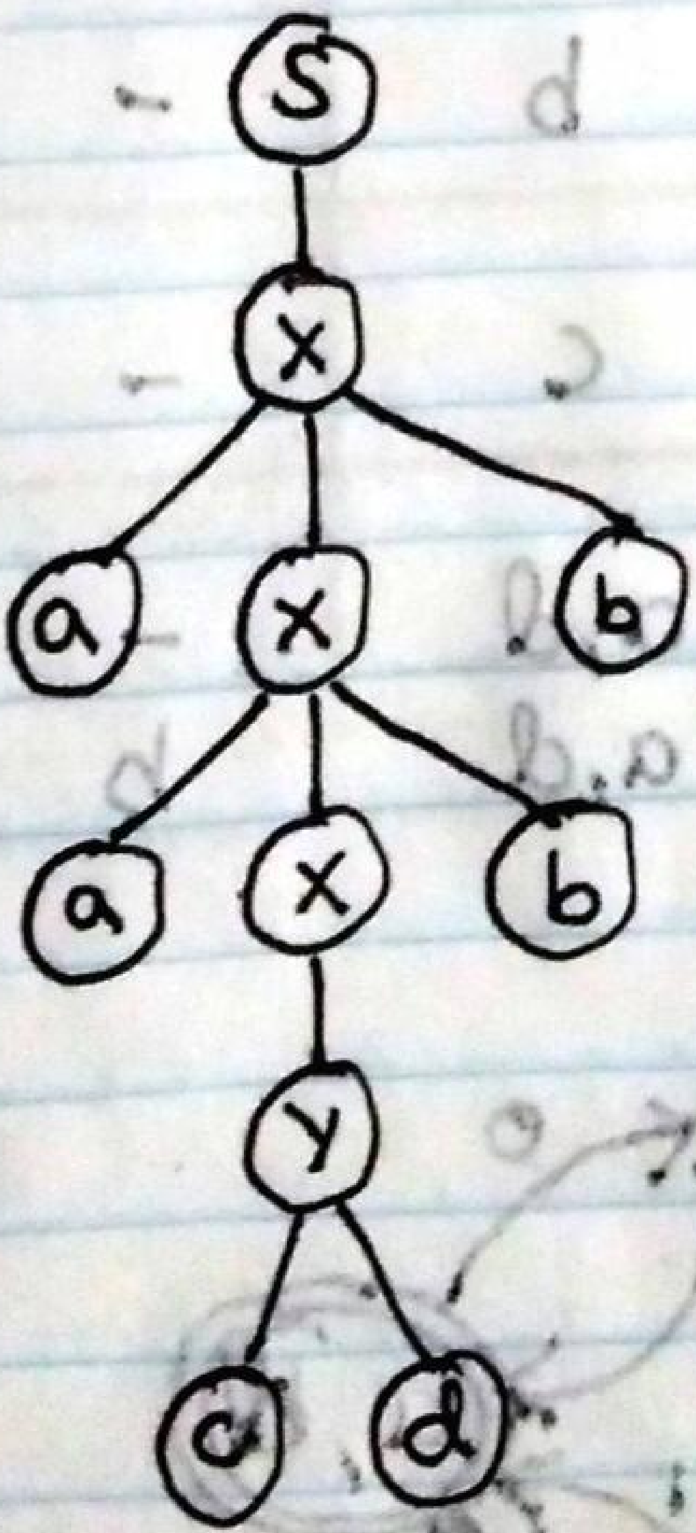
8) The language $a^n cd b^n$ is not a regular expression since no FiniteState Automation can be written that validates & accepts this input since it lacks a memory component. This can also be proved mathematically using the pumping lemma.

9) Terminals : { a, b, c, d }
Non-Terminals : { S, X, Y }

10)

Step	Production Used
S → X	S → X
X → aXb	X → aXb
aXb → aaXbb	X → aXb
aaXbb → aaYbb	X → Y
aaYbb → aacdbb	Y → cd

Parse Tree :


S
X
a
X
b
a
X
b
Y
c
d