Unit V

* Push Down Automata *

As FA is the mathematical model for FSM similarly push down automata (PDA) is a mathematical model for (PDM) push down machine.

Formal defn:- A Pushdown automata M is denoted as $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

Where,

Q = finite set of states

'Γ': stack symbol — Σ = i/p alphabet

Γ = stack alphabet

q_0 = intial state

b blank — i.e ~~set of~~ $q_0 \in Q$

$Z_0 = z_0 \in \Gamma$ It is particular stack symbol called as start symbol (probably consider as b (blank) i.e. blank in many designs).

F = set of Final states $F \subseteq Q$

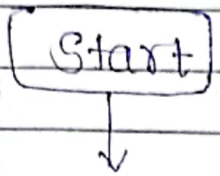
δ = Mapping from $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$

* Difference betn FSM & PDM.

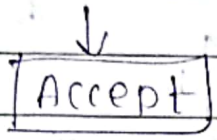
1) FSM does not have memory to remember arbitarily long sequence of inputs but pdm can have memory to remember long sequences of inputs.
2) FSM is less powerful than PDM
3) PDM is more powerful than FSM But less powerful than Turing machine. (TM)

* Pictorial Representation for PDA:-

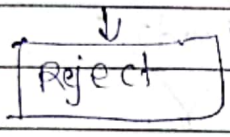
PDA can be represented by some flowchart like notation all these pictorial representation for diff. types of states of PDA are given below

i) 

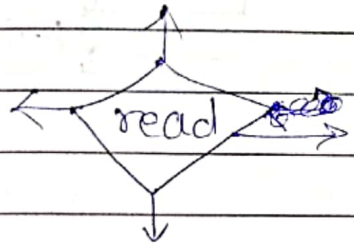
Start the process

ii) 

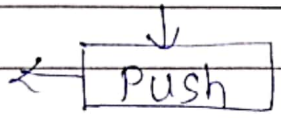
Accept half state

iii) 

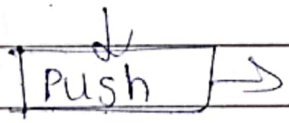
reject half state.

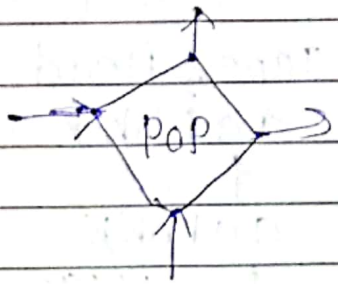
iv) 

Read i\p ~~state~~ symbol from tape

v) 

Push a <letter> i.e some symbol on to stack & Proceed



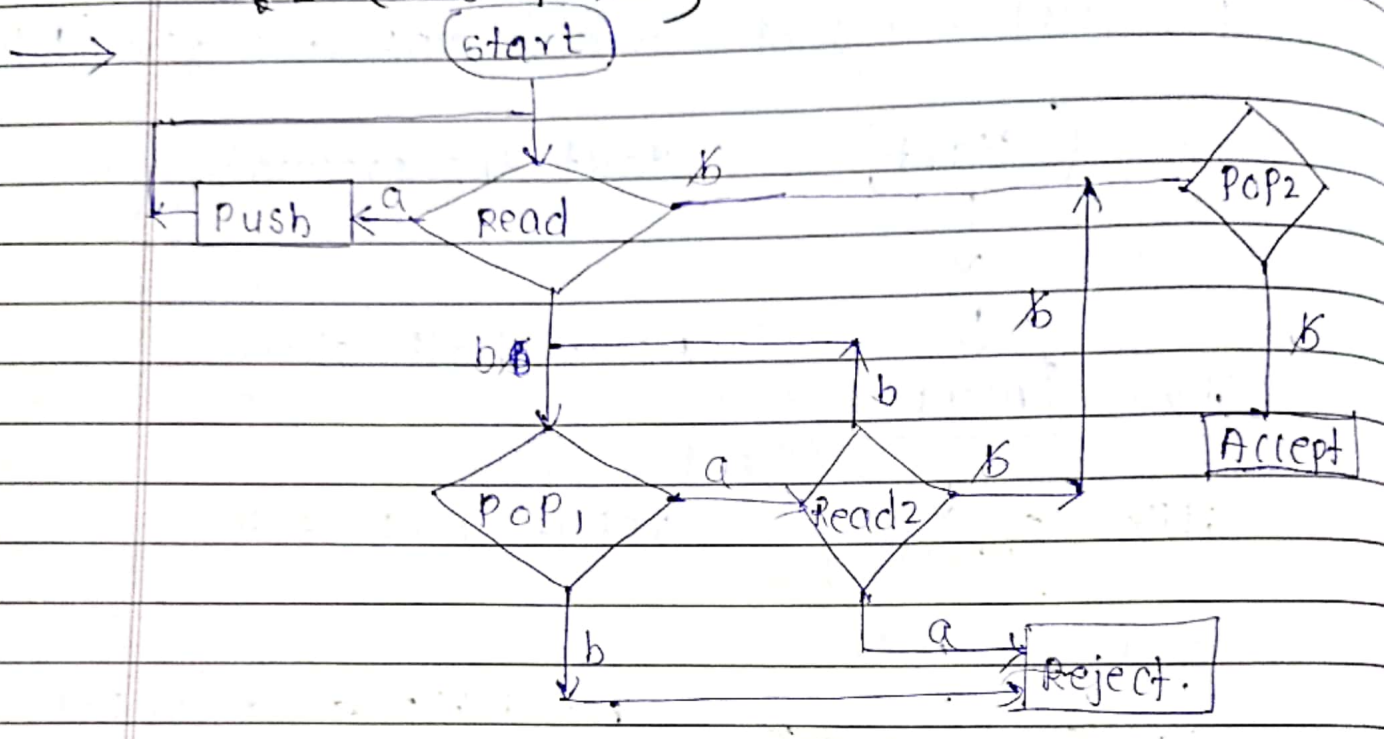
vi) 

Pop a symbol from the stack.

Ex:]

* Construct a PDA accepting a language

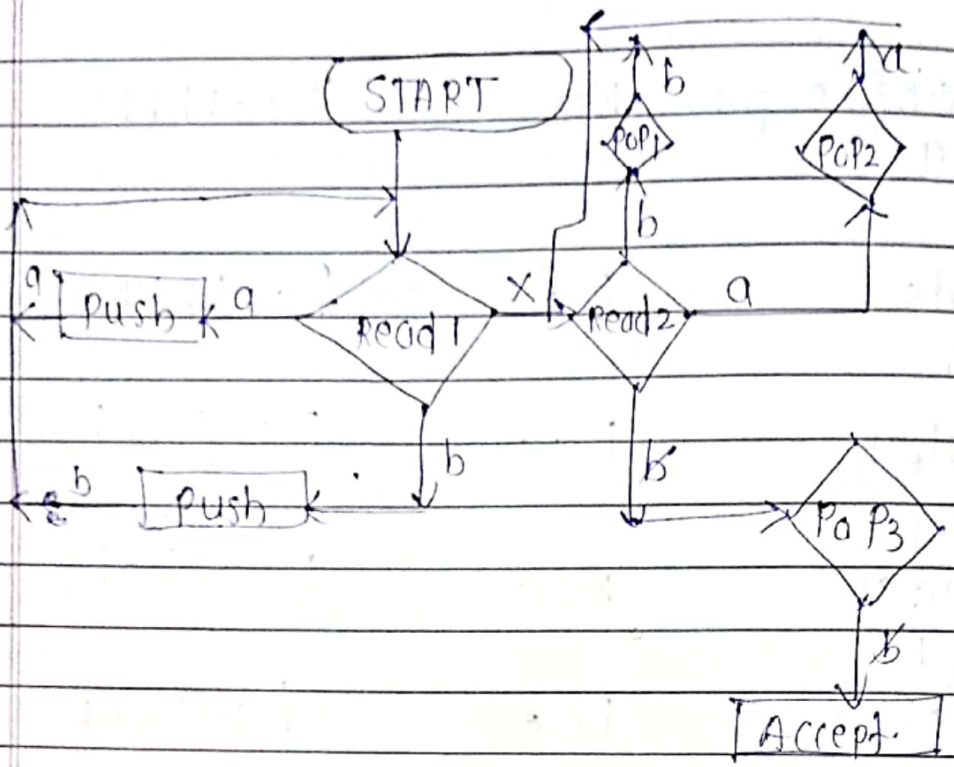
$L = \{a^n b^n \mid n \geq 0\}$



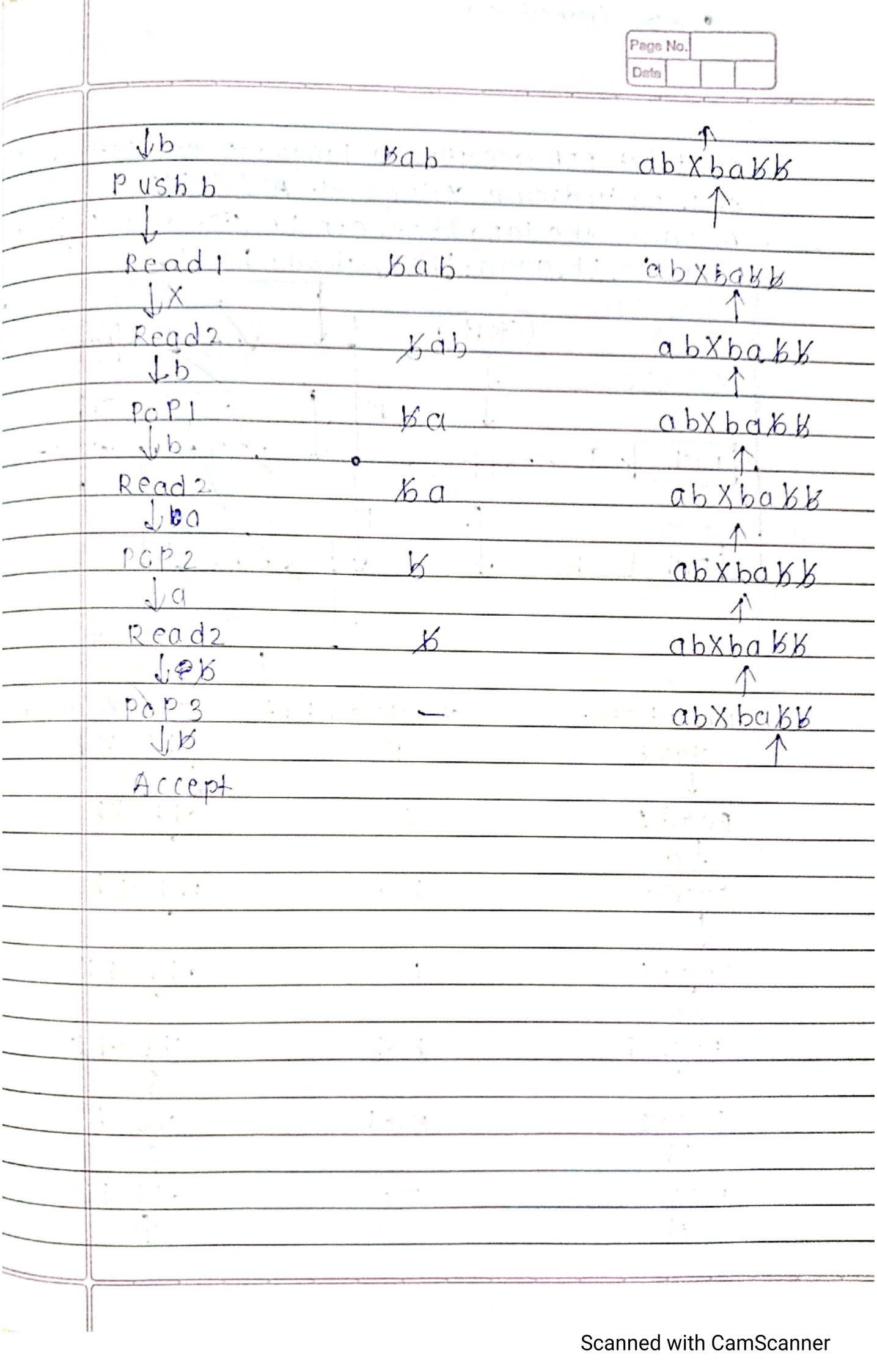
Current State	Stack Content	Tape & Head
start	$	a a b b $ $
↓		↑
Reads a	$	a a b b $ $
		↑
Reads a	$	a a b b $ $
		↑
Reads b	$	a a b b $ $
		↑

↓'a Push a ↓	Xa	↑ aabbXX
Reads ↓a	Xa	↑ aabbXX
Push a ↓	Xaa	↑ aabbXX
read s ↓b	Xaa	↑ aabbXX
POP ↓a	Xa	↑ aabbXX
Read 2 ↓b	Xa	↑ aabbXX
POP 1 ↓a	X	↑ aabbXX
Read 2 ↓b	X	↑ aabbXX
POP 2 ↓b	—	↑ aabbXX
Accept		

* Construct PDA which can accept the language.
L:- { X, aXa, bXb, aaXaa, abXba, baXab, bbXbb, aaaXaaa... }.



current State	Stack content	Tape & Head
Start	$	a b X b a b $
↓		↑
Read 1	$	a b X b a $ $
↓ a		↑
Push a	$ a	a b X b a $ $
↓		↑
Read 1	$ a	a b X b a $ $

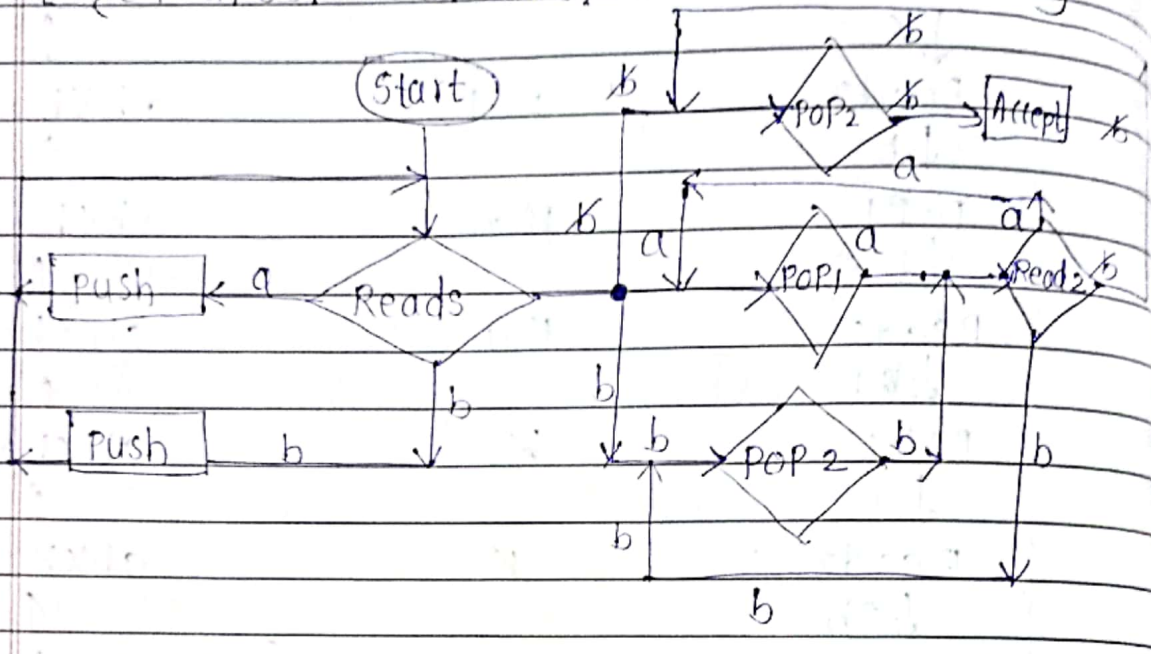


• → Connector.

Page No.
Date

* Construct PDA accepting a language consisting of even palimdrome strings of A's & b's

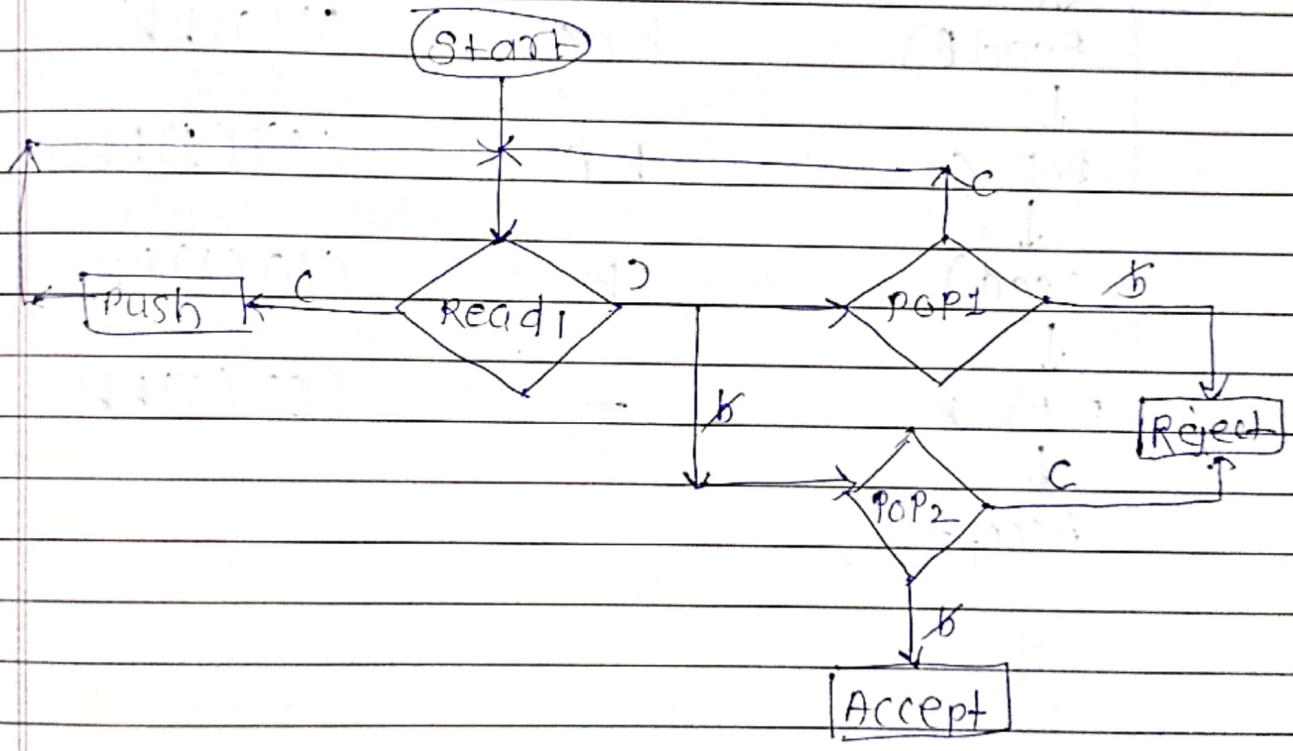
→ Assume the language can be listed as follows
L = {ϵ, aa, bb, aaaa, bbbb, abba, baab, ...}



current state	stack content	Tape & Head
start	$	abbaXX
↓		↕
Read 1	$	abbaXX
↓a		↑
Push a	$a	abbaXX
↓		↑
Read 1	$a	abbaXX
↓b		↑
Push b	$ab	abbaXX
↓		↑
Read 1	$ab	abbaXX
↓b		↑
POP2	$a	abbaXX
↓ab		↑

Read 2	$ a	a a b b a $ $
↓a		↑
Pop 1	~~$ $~~ $	a b b a $ $
↓a		↑
Read 2	$	a b b a $ $
↓b		↑
Pop 3	~~$~~ —	a b b a $ $
↓$	~~$~~	
Accept		

* construct PDA that check ~~where~~ well-formedness of parenthesis.



current state	stack content	Tape & Head
start	$	( ) ( )) $ $
↓		↑
Read 1	$ (	( ) ( )) $ $
↓(		↑
Push (	$ (	( ) ( )) $ $
↓		↑

DPDA: From every state there is an unique transition on a particular symbol

Page No.
Date

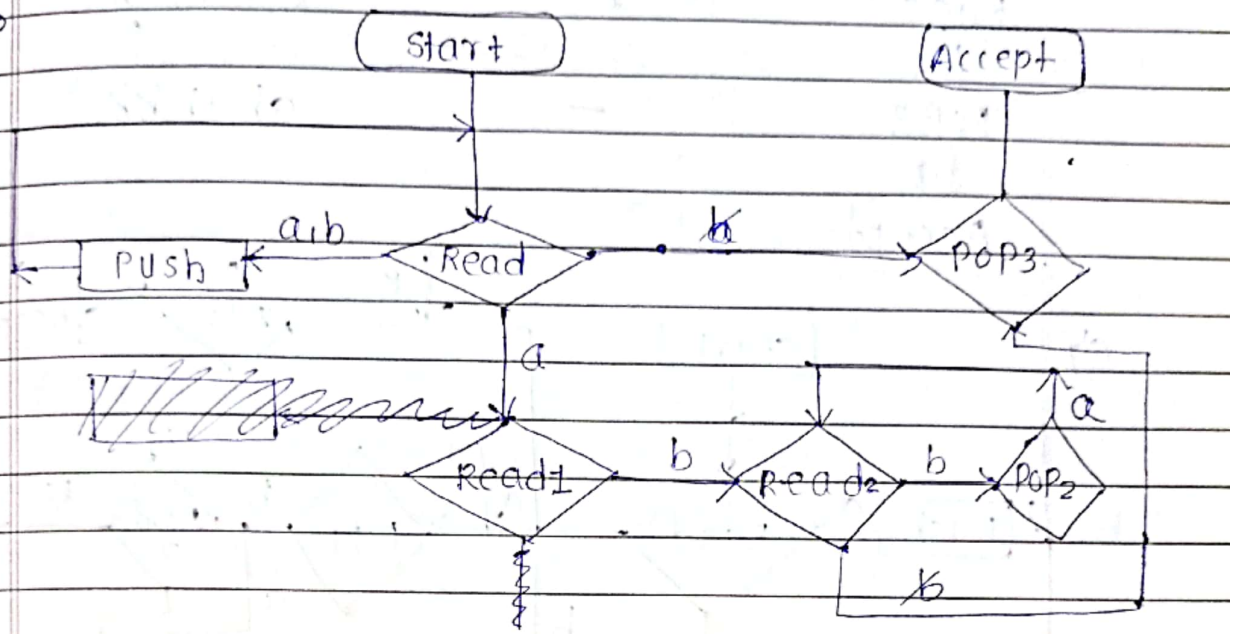
Read (	$	(()())$$
↓ C		↑
Push C	$C	(()())$$
↓		↑
Read (	$CC	(()())$$
↓)		↑
pop (	$C	(()())$$
↓ (		↑
Read (	$C	(()())$$
↓ C		
Push C	$CC	(()())$$
↓ C		
Read)	$CC	(()())$$
↓		
pop C	$C	(()())$$
↓		
Read)	$	(()())$$
↓		
pop $	—	(()())$$
↓		
Accept		

NDPA:- From every state there may be more than on transition on a particular symbol

Page No.
Date

* Construct NDPA which accept the set of all strings containing equal no. of a's & b's.

→



current state	stack content	Tape & Head
start	$	abab$$
↓		↑
Reads	$	abab$$
↓a		↑
Pusha	$a	abab$$
↓		↑
Read1	$a	abab$$
↓b		↑
Pushb	$ab	abab$$
↓		↑
Reads	$ab	abab$$
↓a		↑
Pop1	$a	abab$$
↓b		↑
Read2	$	abab$$
↓b		↑

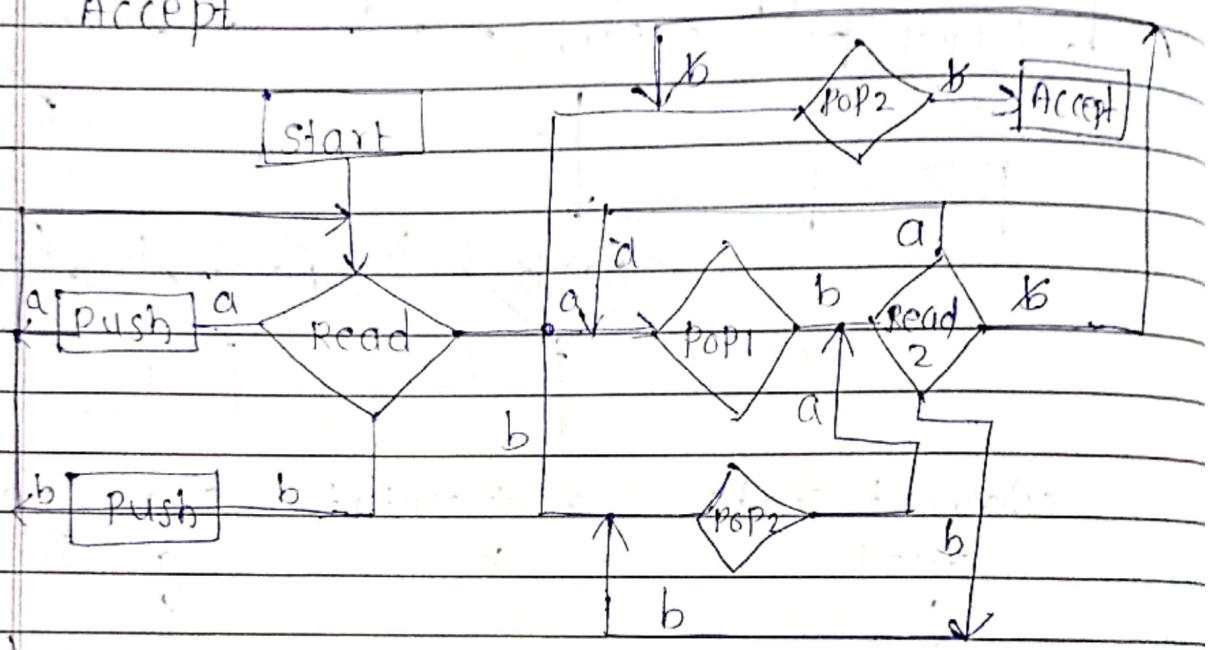
pop 2 ↓a — Read 2 ↓b — pop 3 ↓b — Accept

abaabbb

abaabbb

ababbb

abababb



ababab

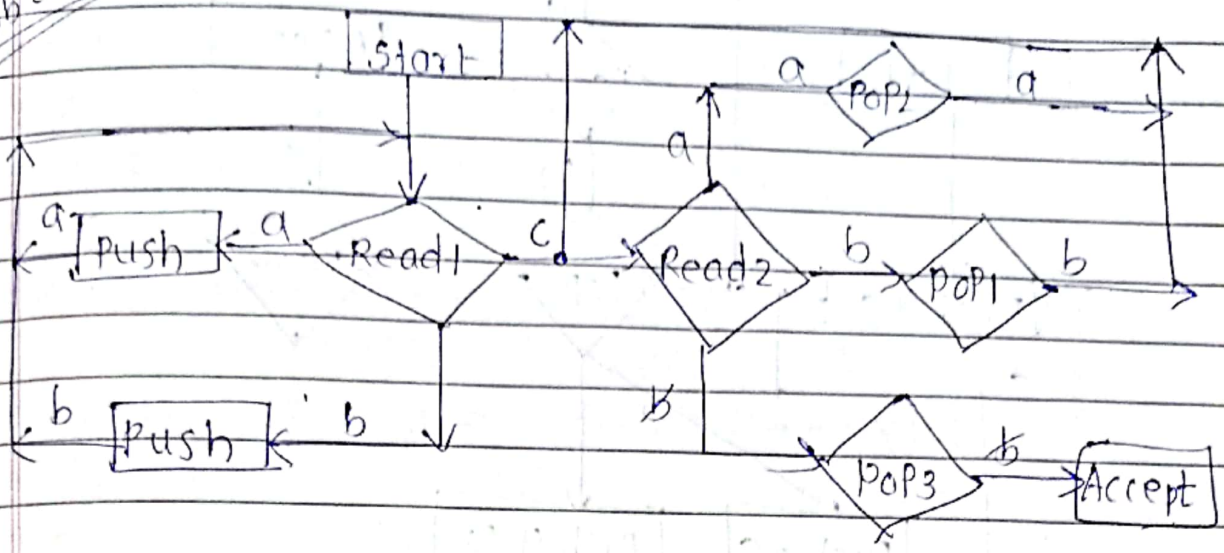
1)* construct PDA which accepting the foll. languages.
$L = \{ a^n b^n c^n \mid n \geq 0 \}$.

2) * construct PDA which accepting the foll. language
$L = \{ w c w^R \mid w \in \{a, b\}^* \}$. by final state

Page No.
Date

abbcbba

2)



* CFG & PDA

* Construct PDA that accepted language generated by CFG $S \rightarrow SS \mid aa$

OR

$L = \{a^{2n} \mid n > 0\}$

→ Step1> convert the foll. above grammer into CNF

let 1st P:R

$S \rightarrow SS$

It is already in CNF

Take 2nd P.R.

$S \rightarrow aa.$

Assume $R_1 \rightarrow a$

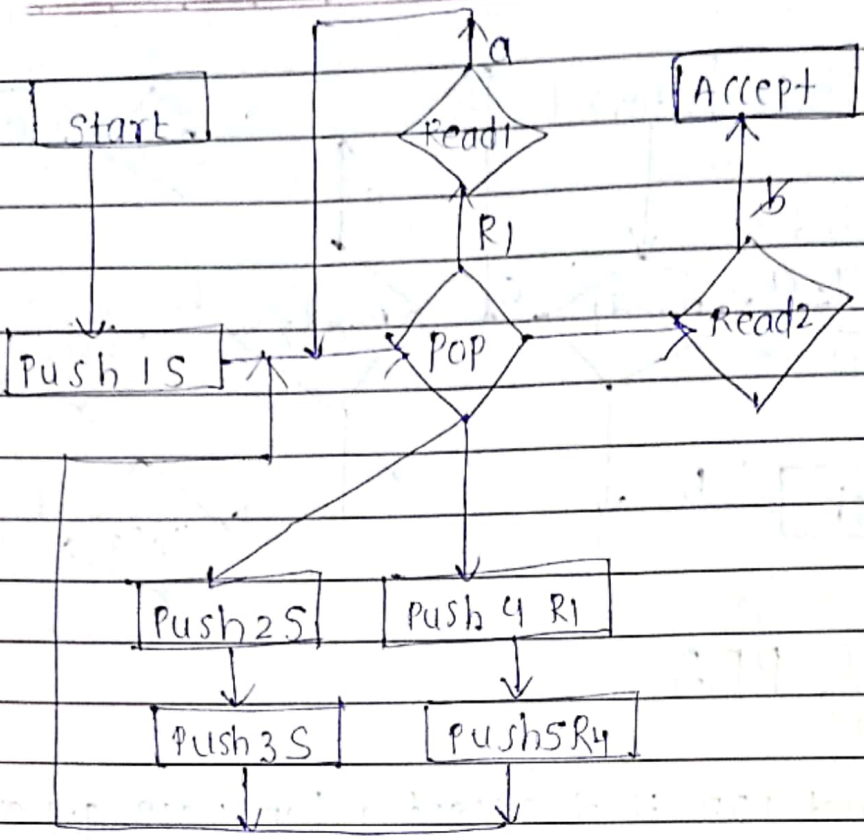
$S \rightarrow R_1R_1$

The final grammer is.

$S \rightarrow SS$

$S \rightarrow R_1R_1$

$R_1 \rightarrow a$



current state	stack content	Tape & Head
start	$	a a a a b b
↓		↑
push 1 S	$ S	a a a a b b
↓		↑
Pop	$	a a a a b b
↓ S		↑
Push 2 S	$ S	a a a a b b
↓		↑
Push 3 S	$ S S	a a a a b b
↓		↑
Pop	$ S	a a a a b b
↓ S		↑
Push 4	$ S R1	a a a a b b
↓ R1		↑
Push 5	$ S R1 R1	a a a a b b
↓ R1		↑

POP ↓R1	$ S R1	a a a a $ $
Read 1 ↓a	$ S R1	a a a a $ $
POP ↓R1	$ S	a a a a $ $
Read 2 ↓a	$ S	a a a a $ $
POP ↓S	$	a a a a $ $
Push $ R1 ↓	$ R1	a a a a $ $
Push S R1 ↓	$ R1 R1	a a a a $ $
POP ↓R1	$ R1	a a a a $ $
Read 1 ↓a	$ R1	a a a a $ $
POP ↓R1	$	a a a a $ $
Read 1 ↓a	$	a a a a $ $
POP ↓b	—	a a a a $ $
Read 2 ↓$	—	a a a a $ $
Accept		

* Construct PDA that accept the language generated by CFG $S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$

→ 1) Delete all ε-productions from the above grammer after deleting ε-production we get

$S \to aSa$

$S \to bSb$

$S \to a$

$S \to b$

2) Identify the nullable non-terminal in the given grammer 'G' we form the production as

$S \to \epsilon$

S is a nullable non-terminal.

3) Identify nullable non-terminal in step 1 we should occur on right hand side of the production rule.

$S \to aSa$

$S \to bSb$

$S \to a$

$S \to b$.

put the value of nullable non-terminal S on RHS is null we got new production rule

$S \to aa$

$S \to bb$

$S \to a$

$S \to b$.

4) So the final grammer without ε-productions is as follow

$S \to aa \mid aSa$

$b \to bb \mid bSb$

$S \to a \mid b$.

convert to CNF

$S \to aa$
$S \to aSa$
$b \to bb$
$b \to bSb$
$S \to a$
$S \to b$

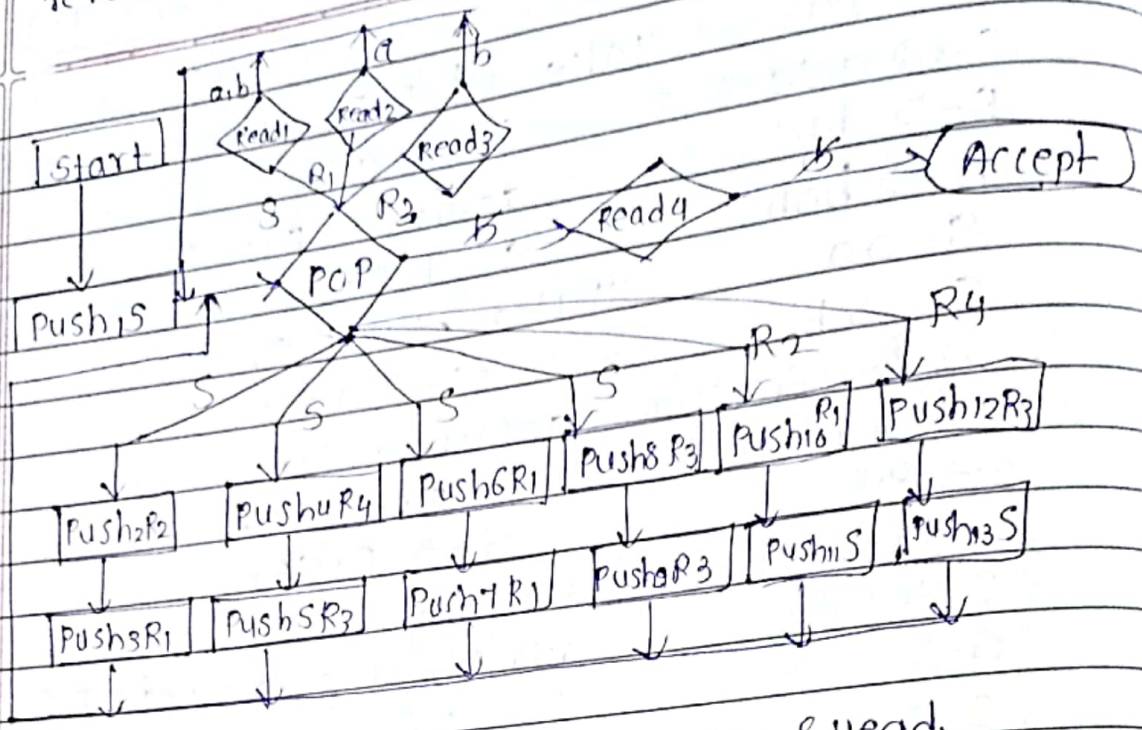
$S \to R_1R_2$
$S \to R_3R_4$
$S \to R_1R_1$
$S \to R_3R_3$
$S \to a$
$S \to b$
$R_1 \to a$
$R_2 \to SR_1$
$R_3 \to b$
$R_4 \to SR_3$

Take 1st P.R.
$S \to aa$.
assume $R_1 \to a$.
$S \to R_1R_1$

Take 2nd P.R.
$S \to aSa$
assume
$R_1 \to a$.
$S \to R_1SR_1$

similarly
our final grammer is
$S \to R_1R_2$
$S \to R_3R_4$
$S \to R_1R_1$
$S \to R_3R_3$
$S \to a$
$S \to b$
$R_1 \to a$
$R_2 \to SR_1$
$R_3 \to b$
$R_4 \to SR_3$.

RHS of production rule are push in reverse order



current state	stack content	Tape & Head.
start	⊥	abab⊥
↓		↑
Push1S	⊥S	abab⊥
↓		↑
POP	⊥	abab⊥
↓S		↑
Push2R2	⊥R2	abab⊥
↓		↑
Push3R1	⊥R2R1	abab⊥
↓		↑
POP	⊥R2	abab⊥
↓R1		↑
Read2	⊥R2	abab⊥
↓a		↑
POP	⊥	abab⊥
↓R2	⊥	↑
*Push10R1	⊥R1	abab⊥
↓		↑

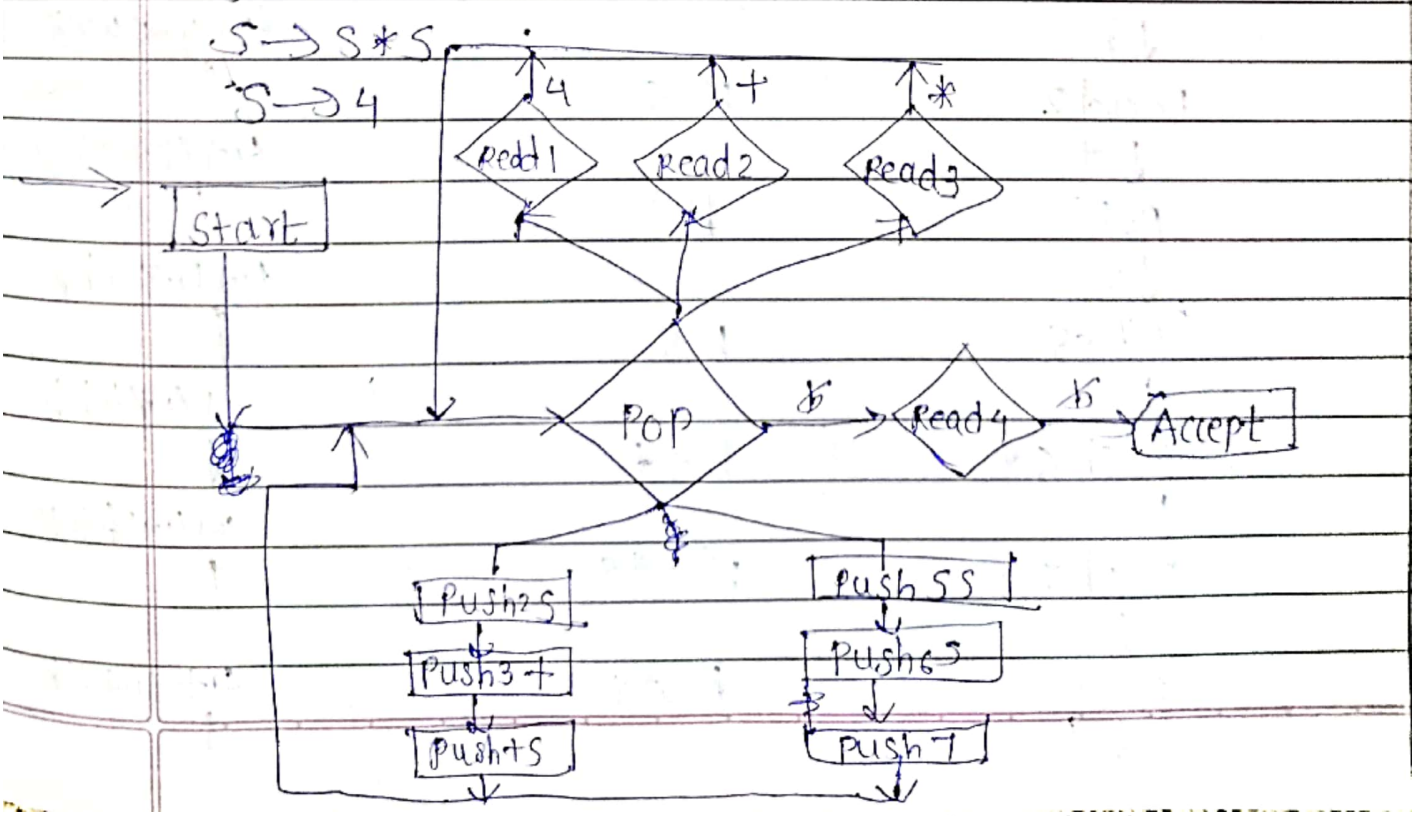
Push 11 S	ß R S	abaßß
↓		↑
Pop	ß R 1	abaßß
↓ S		↑
Read 1	ß R 1	abaßß
↓ b		↑
Pop	ß	abaßß
↓ R 1		↑
Read 2	ß	abaßß
↓		↑
Pop	—	abaßß
↓ ß		↑
Read 4	—	abaßß
↓ ß		
Accept		

* Construct the PDA that accept the language generated by CFG

S → S + S

S → S * S

S → 4



current state	stack content	Tape & Head
start	$	4+4*4$$
↓		↑
push1S	$S	4+4*4$$
↓		↑
POP	$S	4+4*4$$
↓		↑
push2S	$S	4+4*4$$
↓		↑
push3+	$S+	4+4*4$$
↓		↑
push4S	$S+S	4+4*4$$
↓		↑
POP	$+S	4+4*4$$
↓S		↑
Read1	$S+	4+4*4$$
↓4		↑
		4+4*4$$
POP	$S+	↑
↓+		4+4*4$$
Read2	$S	↑
↓+		4+4*4$$
POP	$	↑
↓S		4+4*4$$
push5S	$S	↑
↓		4+4*4$$
push6*	$S*	↑
↓		4+4*4$$
push7S	$S*S	↑
↓		4+4*$$
POP	$S*S	↑
↓3		

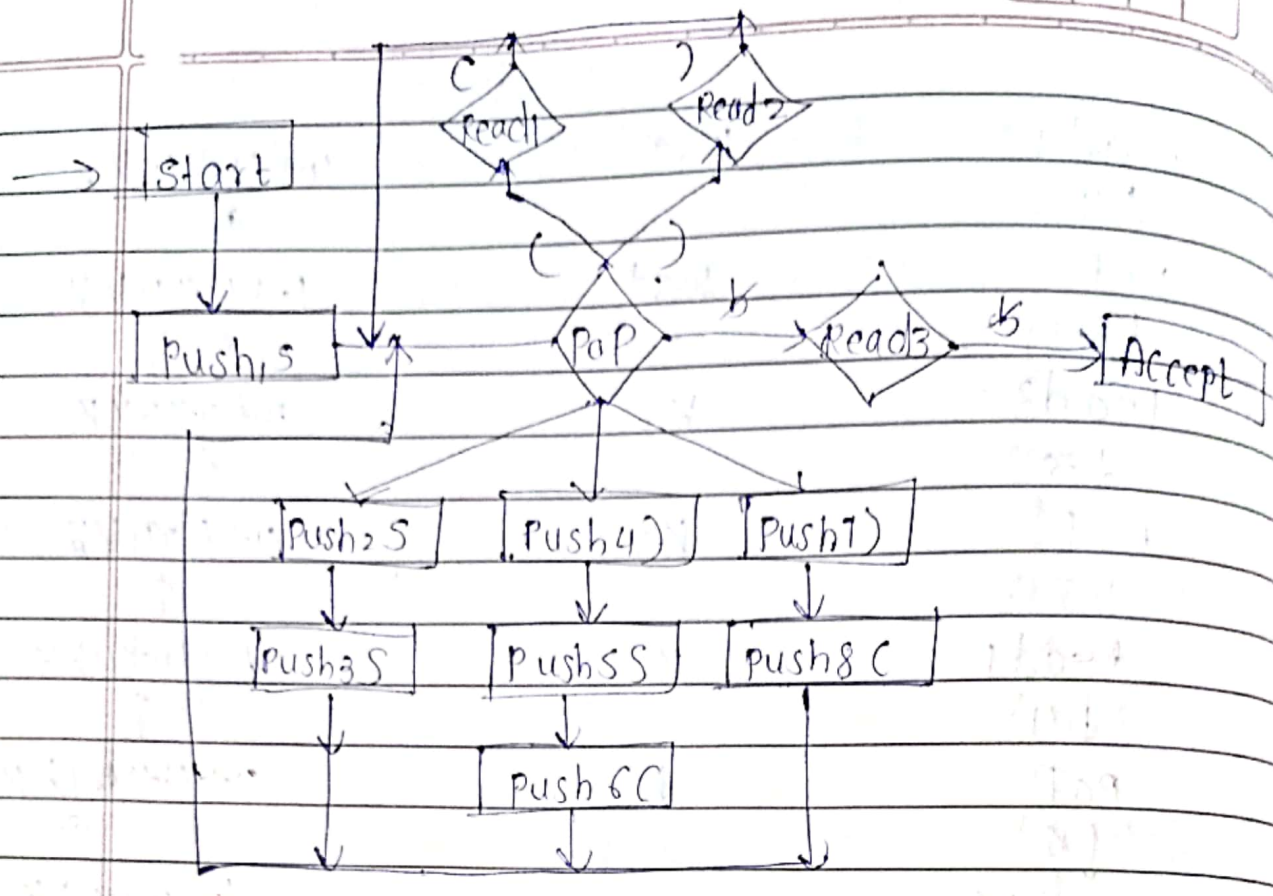
Read 1	$ S *	4 + 4 * 4 $
↓ 4		↑
pop	$ S *	4 + 4 * 4 $
↓ +		↑
Read 3	$ S	4 + 4 * 4 $
↓ *		↑
pop	$ S	4 + 4 * 4 $
↓ S		↑
Read 1	$	4 + 4 * 4 $
↓ 4		↑
pop	$	4 + 4 * 4 $
↓ $		↑
Read 4	—	4 + 4 * 4 $
↓		↑
Accept		

* Construct a PDA that accept the language generated by CFG

S → SS

S → (S)

S → ()



Current state	Stack content	Tape & Head
Start	b	(()())bb
Push1 S	bS	(()())bb
Push4)	bS)	(()())bb
Push S5	bS)S	(()())bb
Push6 (	bS)S(	(()())bb
Push		
Start	b	(()())bb
Push1 S	bS	(()())bb
POP	b	(()())bb
S		

Push$_4$ S ↓	ƀ (̶) S	(()(())ƀb ↑
Push$_5$ S ↓	ƀ) S	(()(())ƀƀ ↑
push$_6$ S ↓	ƀ) S (	(()(())ƀƀ ↑
Pop ↓ C ↓	ƀ) S	(()(())ƀƀ ↑
Read 1 ↓	ƀ) S	(()(())ƀƀ ↑
Pop S ↓ S	ƀ)	(()(())ƀƀ ↑
push$_2$ S ↓	ƀ) S	(()(())ƀƀ ↑
Push$_3$ S ↓	ƀ) S S	(()(())ƀƀ ↑
Pop ↓ S	ƀ) S	(()(())ƀƀ ↑
Push 7) ↓	ƀ) S S	(()(())ƀƀ ↑
Push 8 (↓	ƀ) S	(()(())ƀƀ ↑
Pop ↓ C	ƀ) S)	(()(())ƀƀ ↑
Read 1 ↓ C	ƀ) S) (	(()(())ƀƀ ↑
Pop ↓	ƀ) S)	(()(())ƀƀ ↑
Read 2 ↓	ƀ) S	(()(())ƀb ↑

POP	$)	( ) ( )) $ $
↓S		↑
push (	$))	( ) ( )) $ $
↓		↑
Push & (	$)) (	(( ))) $ $
↓		↑
POP	$))	(( ) ( )) $ $
↓(		↑
Read 1	$))	(( ) ( )) $ $
↓(		↑
POP	$)	(( ) ( )) $ $
↓)		↑
read 2	$)	(( ) ( )) $ $
↓)		↑
POP	$	(( ) ( )) $ $
↓(		↑
read 2	$	(( ) ( )) $ $
↓)		↑
POP	—	(( ) ( )) $ $
↓$		↑
Read 3	—	(( ) ( )) $ $
↓$		↑
Accept		