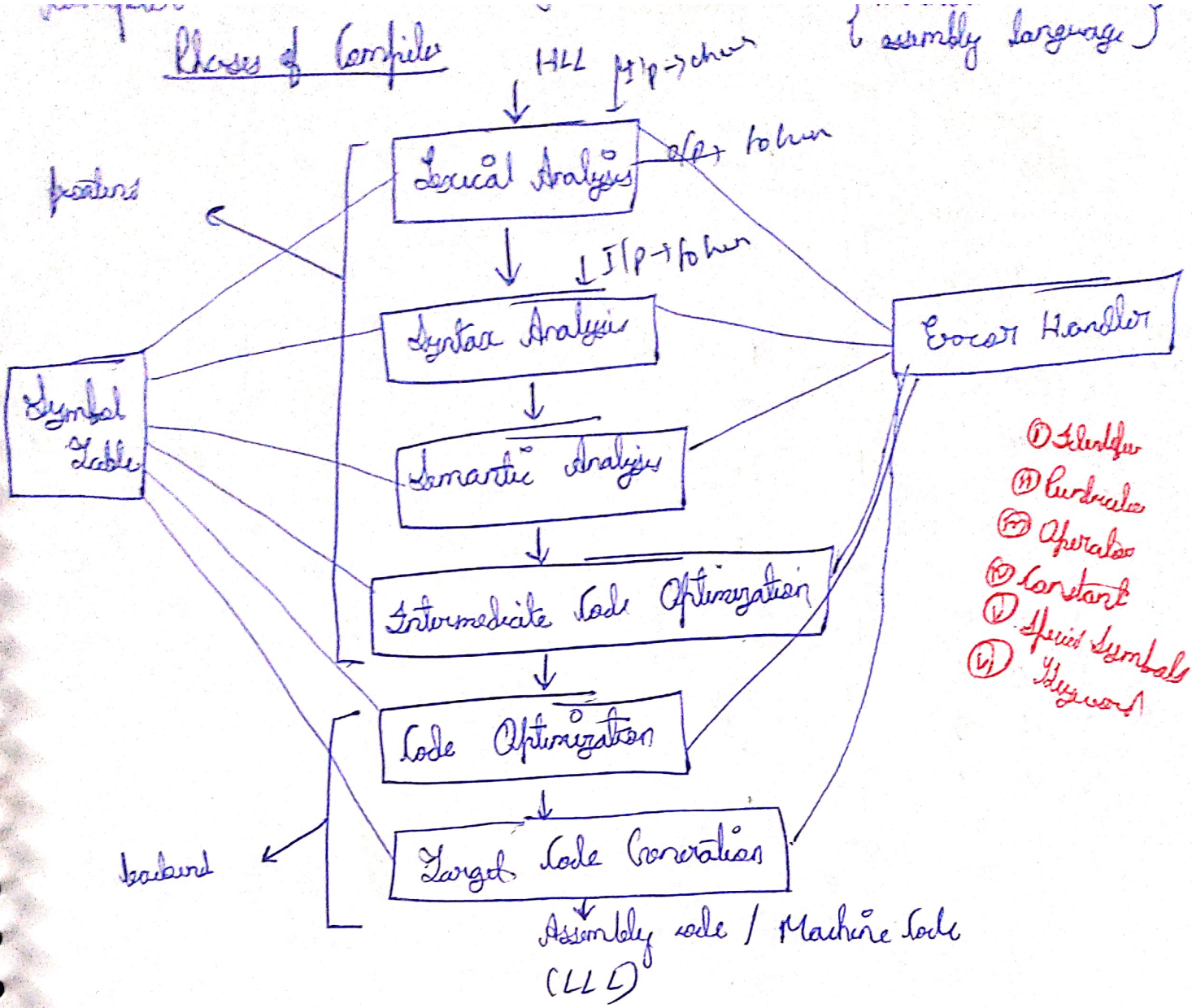
Phases of Compiler



First of all in HLL we get a stream of code. The preprocessor will remove the header files. This stream of character of code goes to lexical analysis and generates stream of tokens as an output. For eg we have $y = 2 * x$, so the lexical analysis will convert this code to tokens.

$y = 2 * x$

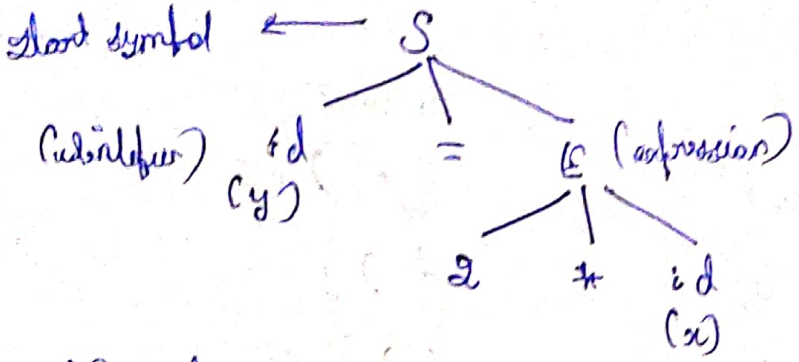
identifier, assignment, constant, operator, identifier

It removes comments, white space. Also called lexemes. It uses DFA for counting characters to tokens.

type, scope, size ⟶ all the information are saves in symbol table

Syntax analysis {very imp phase} - checks whether tokens are syntactically correct or not. We create parse tree & check whether grammer is ambiguous or not.

$y = 2 * x$



Semantic analysis - in this we check logical errors in compile time. In this we use SDT (syntax directed translation). With parse tree we apply actions, & this is defined in SDT.

Intermediate code generator - {very imp phase} In this we use 3 address code. It converts above parse tree into 3 address code.

eg $z = a + b + c$

$t = a + b$

$t_1 = t + c$ } 3 address code

$z = t_2$

{ all errors are handled by error handler

Code Optimization - In this we usually talk about global & local optimization. It divides code into block, inside the block we have local optimization & outside the block we have global optimization.

$t = 2 * x$ ⟶ time consuming

$t = x + x$ ⟶ optimized code

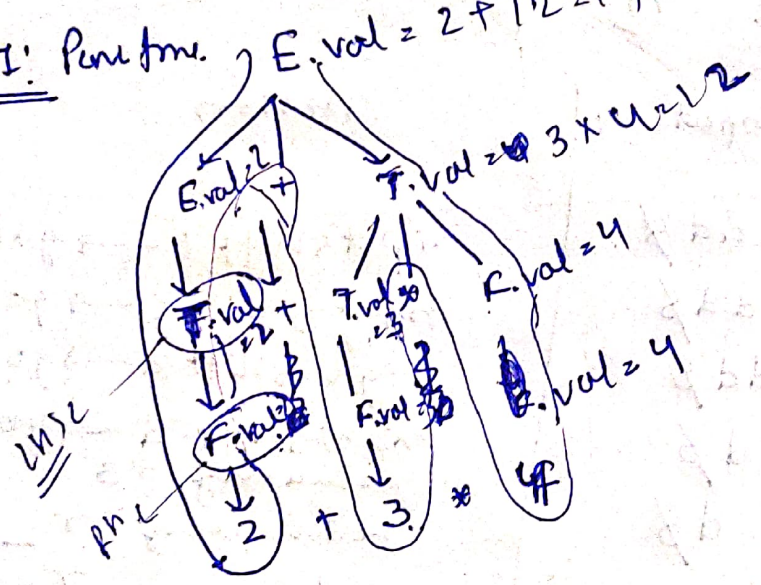
Target code generator - {machine code generation}

SDT (Syntax directed Translation)

Ques: $E \rightarrow E + T$ { E.val = E.val + T.val }
$E \rightarrow T$ { E.val = T.val }
$T \rightarrow T * F$ { T.val = T.val * f.val }
$T \rightarrow F$ { T.val = F.val }
$F \rightarrow num$ { f.val = num }.

String: 2+3*4

Step 1: Parse tree.



Step 2: Scan parse Tree → Bottom up fusion.
→ Top down fusion

top down
left to right.

Step 3: Eval = 2+12=14

Qu 2. $E \to E+T$ { printf("+") }.

$E \to T$ { }

$T \to T*F$ { printf("*") }

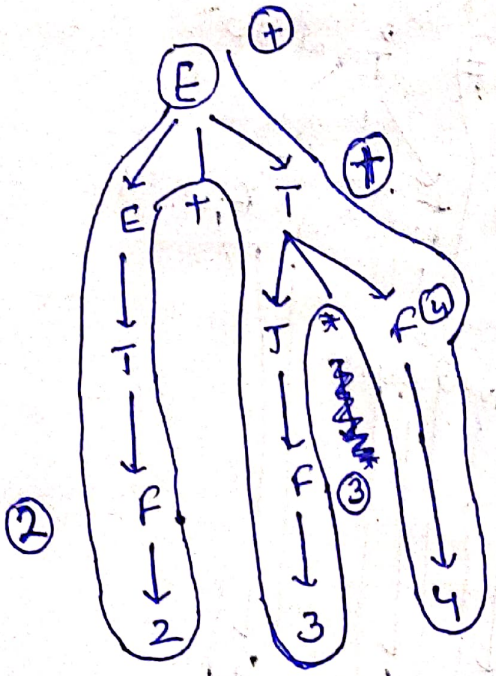
$T \to F$ { }

$f \to num$ { printf("num") }

String 2+3*4.

O/p.

Ans Steps:



O/p: 234*+ { string is converted to infix to postfix }

Qu 3: $E \to E+T$ { E.val = E.val * T.val }

$E \to T$ { E.val = T.val }

$T \to T*F$ { T.val = T.val + F.val }

$T \to F$ { T.val = F.val }

$F \to num$ { F.val = num }

I/p 2+3*4

↓

2+(3*4) = 14

Those grammers tells u below will solved first only. Not according meth.

Qus-4- $E \to E * T$ { E.val = E.val * T.val }

$E \to T$ { E.val = T.val }

$T \to T \otimes F$ { T.val = T.val + F.val }

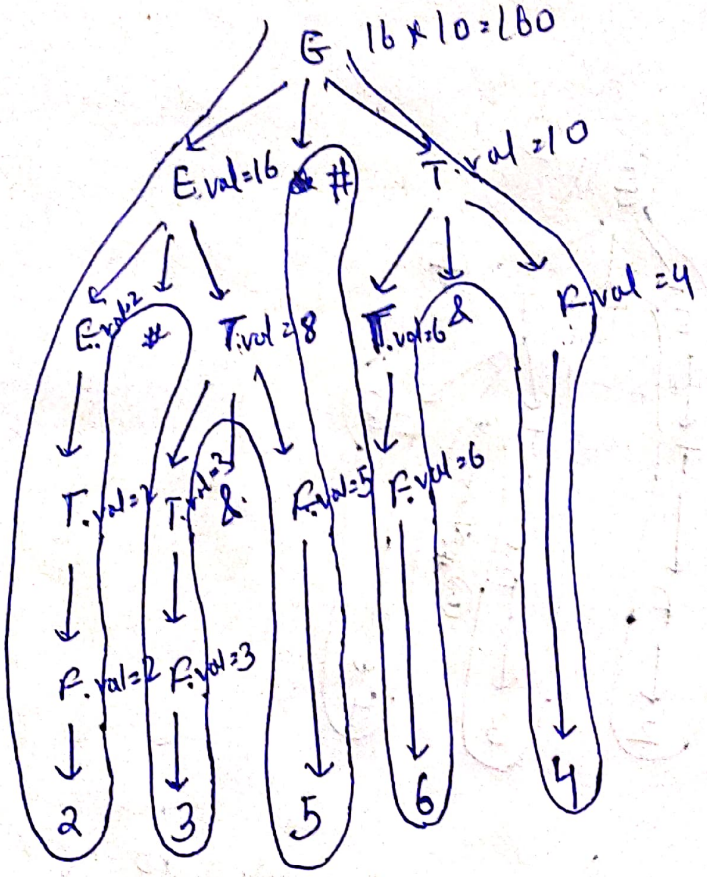
$T \to F$ { T.val = F.val }

$F \to num$ { F.val = num }.

Ans 4.

String 2#385#624

Ans.



Shortcut

2#385#624

$2 * (3+5) * (6+4)$

$2 * (8) * (10)$

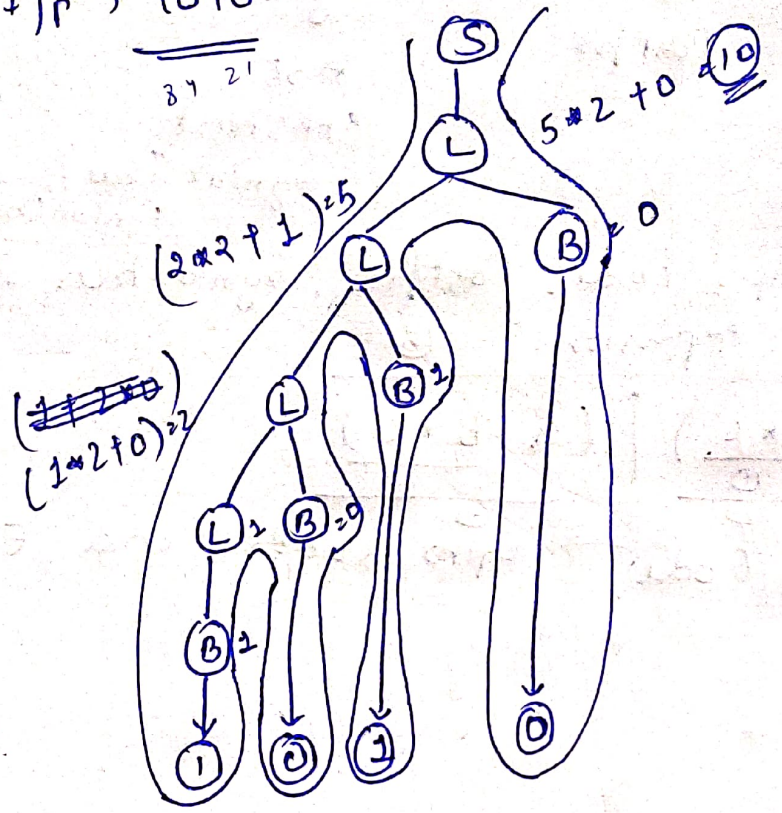
$16 * (10) = 180$ Ans

Qu =.

$S \to L$	{ S.dv = L.dv }
$L \to LB$	{ L.dv = L.dv * 2 + B.dv }
$L \to B$	{ L.dv = B.dv }
$B \to 0$	{ B.dv = 0 }
$B \to 1$	{ B.dv = 1 }.

Ans =

Inp → 1010



① SDT is a CFG (Context free grammar)

② Evaluate order of Sgnt Semantic rules used to

③ Semantic rules embedded in right side of production.

Intermediate Code Generation

- Linear Form
  - 3 address code
  - Postfix
- Tree form
  - AST (Abstract Syntax Tree) / Syntax Tree
  - DAGs (Directed Acyclic graph)

Q) Write 3 address code, Postfix, Syntax tree & DAGs for following expression

$$n = (a + b * c) \,/\, (a - b * c)$$

(operator numbering: * ①, + ②, * ③, − ④, / ⑤)

Sol: Step 1: 3 address Code → upto 3 address in a Expression.

$t_1 = b * c$

$t_2 = a + t_1$

$t_3 = b * c$

$t_4 = a - t_3$

$t_5 = t_2 / t_4$

$n = t_5$

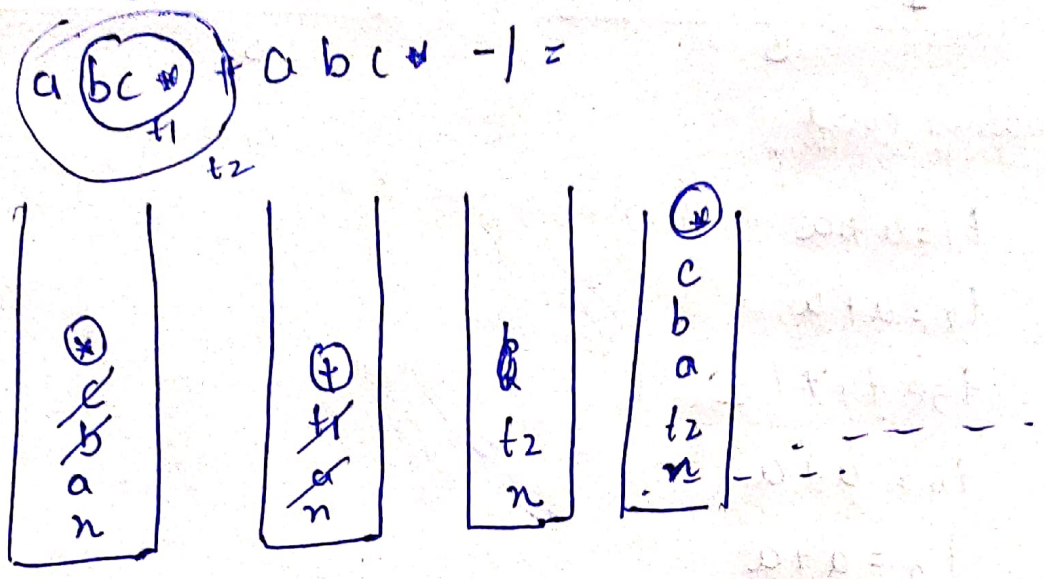
Step 2: Postfix — abc * + abc * - /

n abc * + abc * - / =

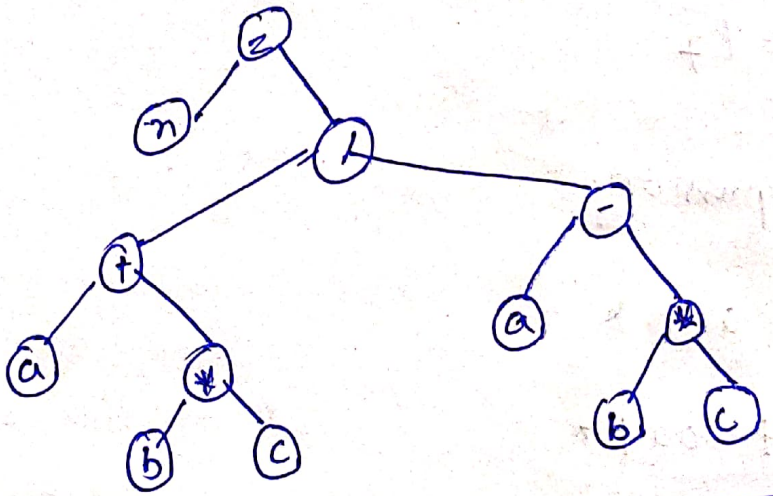
Postfix and 3address code are similar expression

↓ on solving postfix expression we get 3 address code.

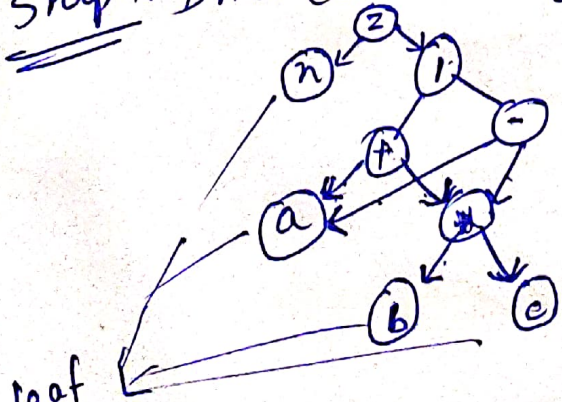
x a bc * + a b c * - / =



Step3: AST ≈ ST (Syntax Tree)



Step4: DAG (Directed Acyclic graph) (One variable one time a, b, c)



No. of nodes in DAG = ? (9)

No. of internal nodes = (5)

No. of leaf Nodes = (4) = (No. of distinct variables) i.e, a, b, c

No. of Edges = (10)

Ques.2. Write 3 addrss code, postfix, Syntax Tree, DAG for the following expression.

$$x = \frac{\overbrace{(\underbrace{(a+a)}_{1} * \underbrace{(a+a)}_{2})}^{3} / \overbrace{(\underbrace{(a+a)}_{4} * \underbrace{(a+a)}_{5})}^{6}}{7}$$

Ans.2

Step1: 3 addrss code

$t_1 = a+a$

$t_2 = a+a$

$t_3 = t_1 * t_2$

$t_4 = a+a$

$t_5 = a+a$

$t_6 = t_4 * t_5$

$t_7 = t_3 / t_6$

$x = t_7$

Step2: postfix Expression.

LHS: aa+ aa+*

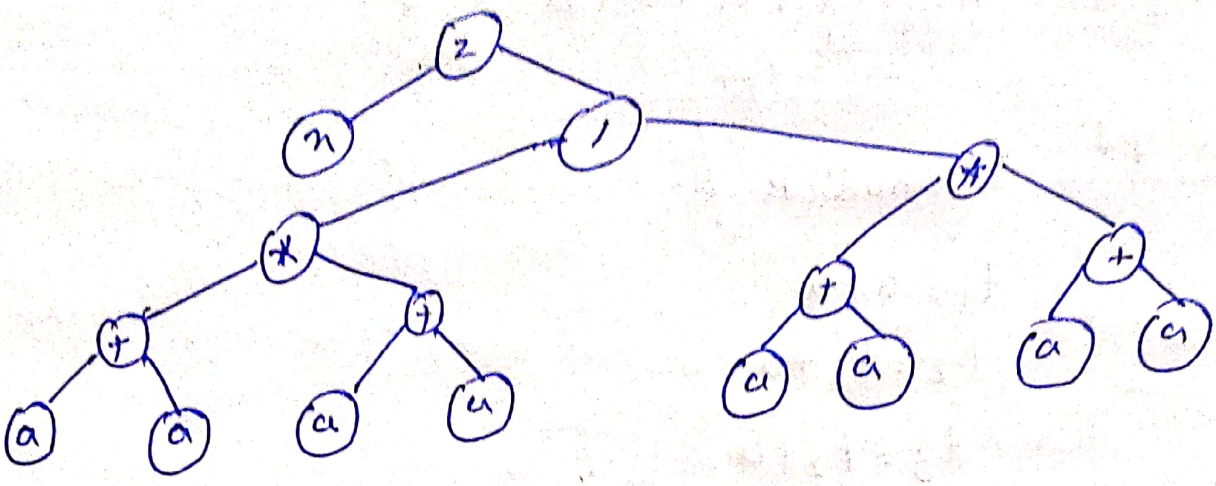
RHS: aa+ aa+*

LHS / RHS

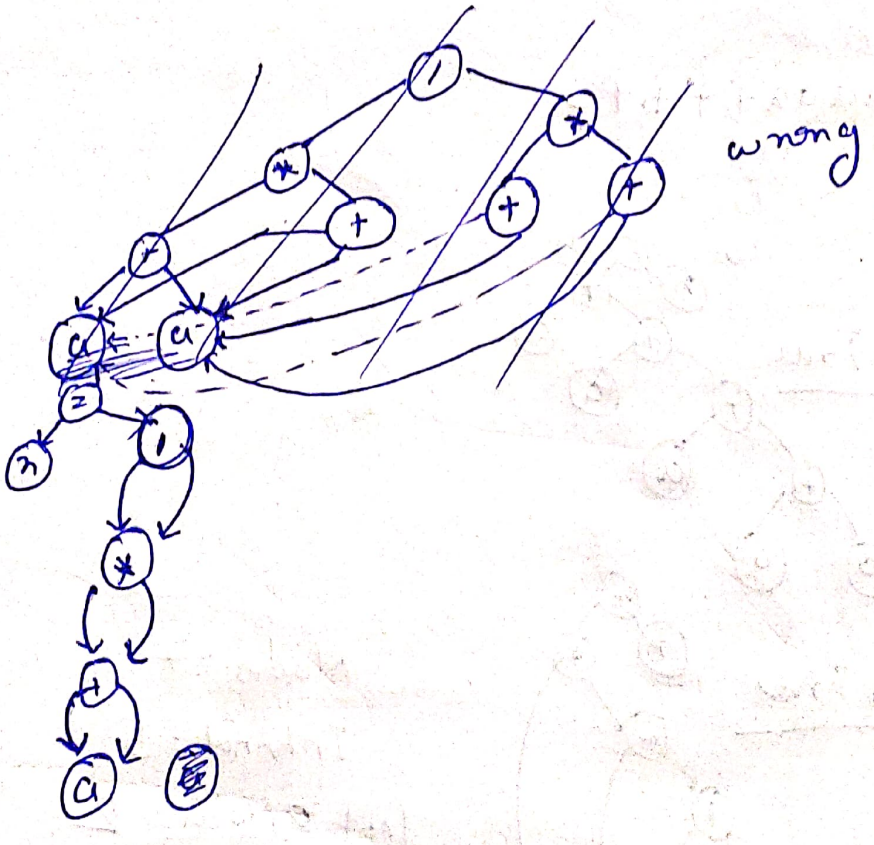
aa+ aa+* / aa+ aa+*

x aa+aa+*aa+aa+*/ =

Step 3: Syntax Tree.



Step 5: DAG.



wrong

Nodes = 6

Internal Node = 4

Leaf Node = 2

Edges = 8

Qu -3. Write 3 addrss code, postfix - - -

Ans 3- $n = (((a+a)+a)+a)+a)$

5
1
2
3
4

Step1: Intermediate

$t_1 = a + a$

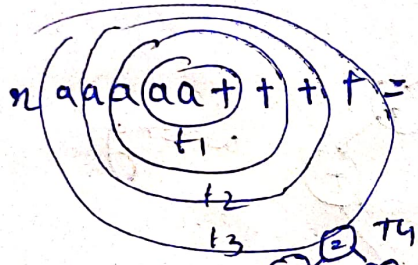
$t_2 = t_1 + a$

$t_3 = t_2 + a$

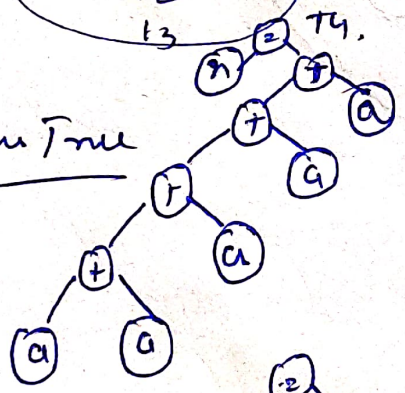
$t_4 = t_3 + a$

$t_5 = t_4$

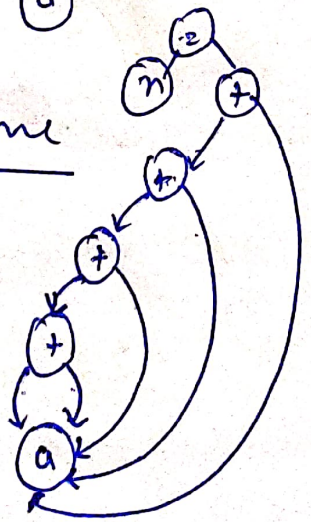
Step2:



Step3: parse Tree



Step4: DAG Tree



Nodes = 7

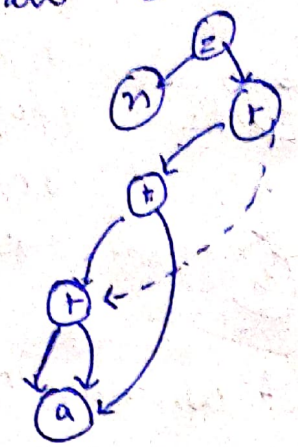
Internal = 5

Leaf Nodes = 2

Edges = 10

Qu-4 Draw DAG for $n = (((a+a)+a)+(a+a))$

Ans-4



Nodes = 6
Leaf Nodes = 2
Internal = 4
Edges = 8

Qu-5. Find min. no of nodes & edges in DAG for following 3 addr Code? → only income of min follow the 3 steps.

Ans-5

$a = b + c$
$c = a + d$
$d = b + c$
$e = d - b$
$a = e + b$

Step 1: In this type of case first of all create an Expression from bottom to up.

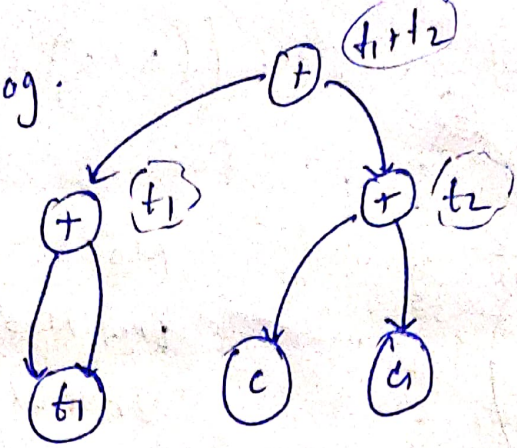
$a = e + b$
$\downarrow$
$d - b + b$
$\downarrow$
$b + c$
$\downarrow$
$b + a + d$
$\downarrow$
$b + b + c + d$

Expression: $b + b + c + d$

Step 2: follow, write Optimized three address code for $b+b+c+d$

$t_1 = b + b$
$t_2 = c + d$
$a = t_1 + t_2$

Step3: Create dag.



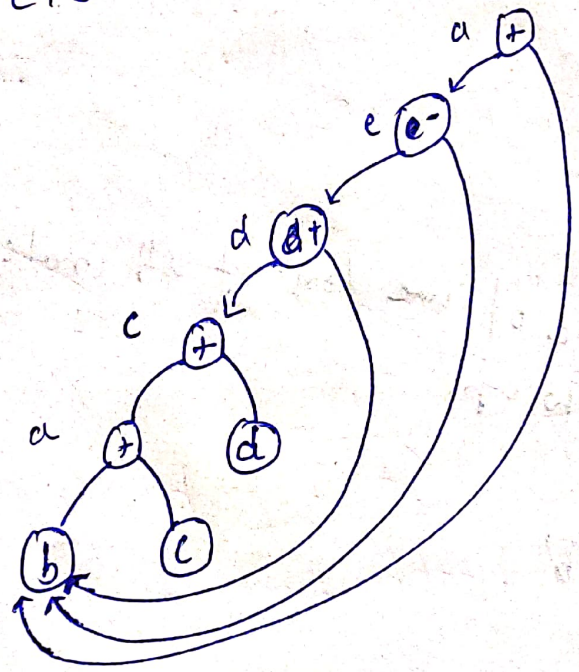
Nodes = 6
Leaf = 3
Internal = 3
Edges = 6

Que-6 Same Question as 5 but without using min Nodes draw above question DAG?

Ans-6

$a = b + c$ ✓
$c = a + d$ ✓
$d = b + c$ ✓
$e = d - b$ ✓
$a = e + b$ ✓

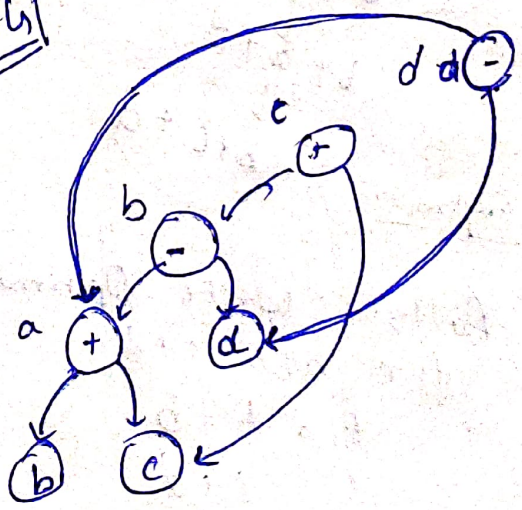
DAG



Que-7

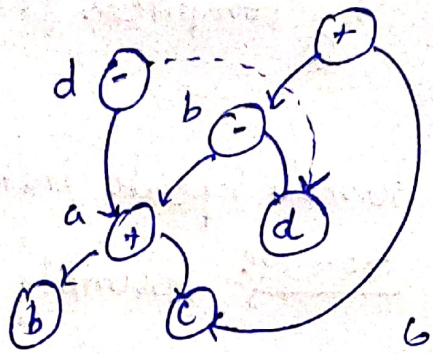
$a = b + c$ ①
$b = a - d$ ②
$c = b + c$ ③
$d = a - d$ ④

DAG



Ans-7

DAG 2



3 | 6 | 4

Qu-7 $x = a + a * (b - c) + (b - c) * d$

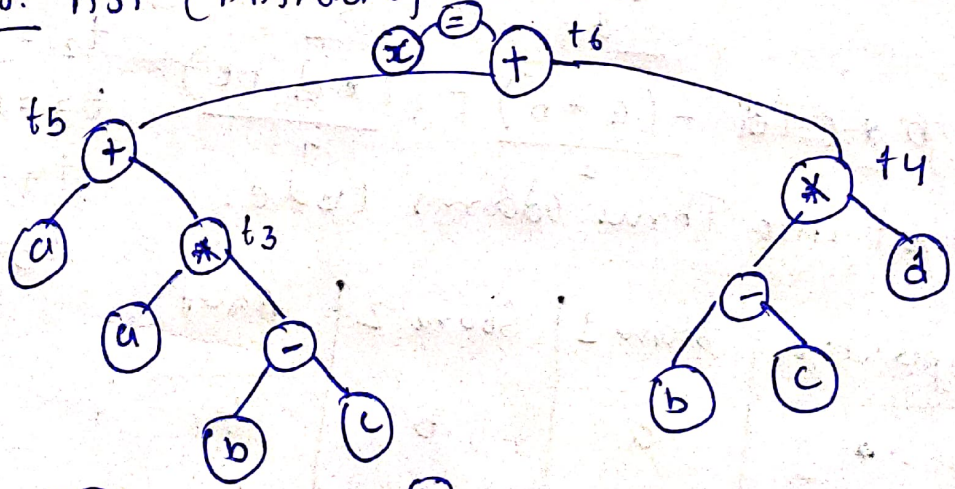
(b-c): 1, 5 | (b-c): 2

My:

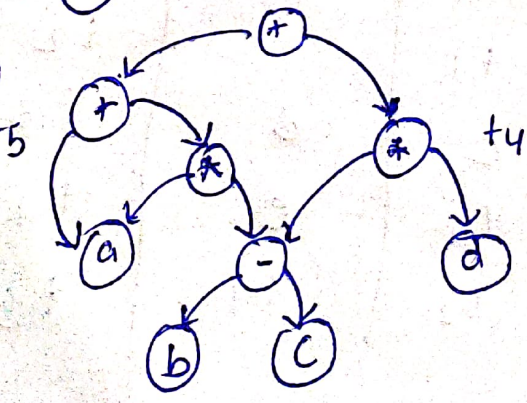
$t_1 = b - c$
$t_2 = a * t_1$
$t_3 = a + t_2$
$t_4 = b - c$
$t_5 = t_4 * d$

$t_1 = b - c$
$t_2 = b - c$
$t_3 = a * t_1$
$t_4 = t_2 * d$
$t_5 = a + t_3$
$t_6 = t_5 + t_4$
$x = t_6$.

Step 3: AST (Abstract Syntax Tree).



Step 4: DAG



Three address code (TAC)

(1) Intermediate code used by compilers for optimizing codes.

(2) Expression broken down into several seperate instruction

(3) These seperate instruction translate Assembly language.

(4) At most three operands.

(5) Combination of assignment & binary operation.

(6) Atmost one operator at RHS of expression.

Eg $x = y + z$

$t_1 = y * z$

$t_2 = x + t_1$

t_1 & t_2 are compiler generated temp name.

Types of Three Address code

(1) Quadruples ⇒ 4 fields.

add → easy to rearrange for global code

disadv. easily access the temporary values using symbol tbl.

operator | S1 | S2 | distination / Result

Eg ~~a := b * c + d~~

$a = -(a * b) + (c * d + e)$

$t_1 = a * b$

$t_2 = -t_1$

$t_3 = c * d$

$t_4 = t_3 + e$

$t_5 = t_2 + t_4$

Step 1: Convert into Three address code.

	operator	Source 1	Source 2	Result
0	*	a	b	t_1
1	-	t_1		t_2
2	*	c	d	t_3
3	+	t_3	e	t_4
4	+	t_2	t_4	t_5

diss

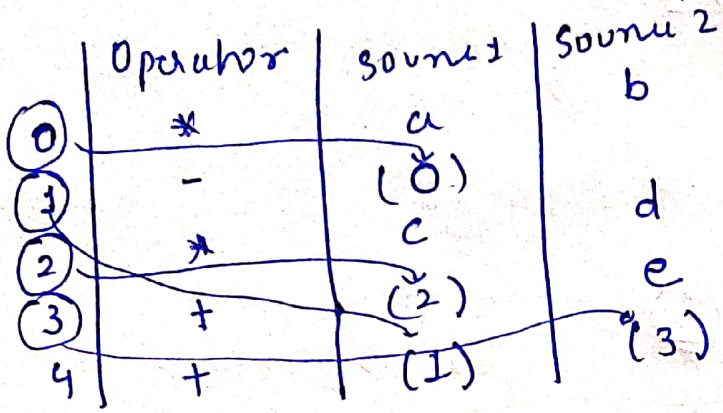
(1) inconvenient Space complexity

(2) Certain jobs of temporary

(3) slower.

(2) Triples. → 3 fields → operator
→ Soure 1
→ Soure 2

Use's Reference as to another triple value



Diss ⇒ (1) Difficult to optimize the code

(2) difficult to rearrange the code.

(3) Implicit

(3) Indirect triples.

Storing Index of memory mainly indirect addrss.

Make uses of pointers.

or directly storing index of memory

Diss easier to rearrange code

1001 (0)
1002 (1)
1003 (2)
1004 (3)
1005 (4)

Backpatching: leaving the labels empty and filling as from later is called backpatching.

Eg1 if (a<d) then t=1 else t=0.

(1) if (a<b) goto 4
(2) t=0
(3) goto 5
(4) t=1
(5)

Eg 2 if (a<b) && (c<d) then t=1 else t=0

1) if (a<b) goto (4)
2) t=0
3) goto (7)
4) if (c<d) goto (6)
5) goto (2)
6) t=1
7)

Eg 3 for (i=1; i<=n; i++) {
n = a+b*c;
}

1) i=1
2) if (i<=n) goto (4)
3) goto (9)
4) $t_1 = b * c$
5) $t_2 = a + t_1$
6) $n = t_2$
7) $i = i + 1$
8) goto (2)
9)

✱ Boolean Expression Three addrss code.

If a < b AND c > b
then p = q + r

Soln.
1) if (a < b) goto 3
2) goto 7
3) if (c > b) goto 5
4) goto 7
5) $t_1 = q + r$
6) $p = t_1$
7)

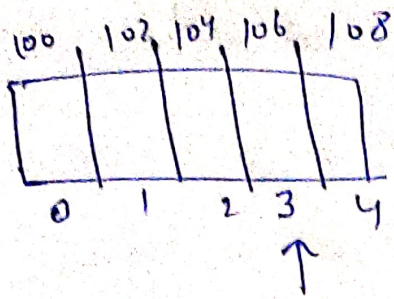
✱ Array's Three Addrss Code.

$A[i] = t_2[t_1]$

t_2 = addrss A - width (generally 4 bytes)
t_1 = i × width (generally 4 bytes)

Eg
1. a[i] = 10

$t_1 = i \times 4$
t_2 = addrss(a) - 4
$t_2[t_1] = 10$



$A[3] = Base + (3-0) \times 2$

$A[3] = 100 + 6 = 106$

Convenchions

t_1 = base address of Array.

$t_2 = i * 2$

$t_3 = t_1[t_2]$

Qu:

```
int A[10], B[0]
int n=0, i
for(int i=0; i<10; i++){
    n = n + A[i] * B[i];
}
```

Ans

1) $n = 0$
2) $i = 0$
3) if ($i >= 0$) goto 16
4) t_1 = base address of array
5) $t_2 = i * 2;$
6) $t_3 = t_1[t_2]$
7) t_4 = base address of array
8) $t_5 = i * 2;$
9) $t_6 = t_4[t_5];$
10) $t_7 = t_3 * t_6$
11) $t_8 = n + t_7$
12) $n = t_8$
13) $t_9 = i + 1$
14) $i = t_9$
15) goto 3

Ambigious Grammar

N.T only child
V = { E } by hit try method
T = { +, *, id }

Eg $E \to E+E \mid E*E \mid id$

word = id + id * id

Left most derivation (LMD)

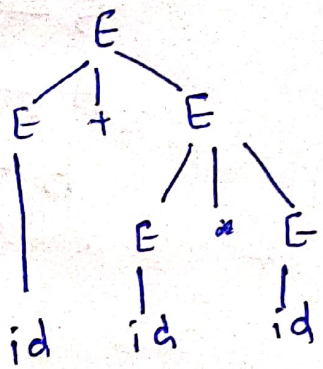
Replace ~~LHS~~ ~~Right~~ Left most symbol of Rhs.

$E \to \textcircled{E}+E$

$\Rightarrow id+E$

$\Rightarrow id+E*E$

$\Rightarrow$ id + id * id



Right Most Derivation (RMD)

$E \to E+\textcircled{E}$

$E \Rightarrow E+E*E$

$E+E*id$

$E+id*id$

id + id * id

Ambigious grammar : If more than one parse Tree exist of ~~strin~~ single string either using LMD or RMD then ambigious gramm.

when A string that can be derived from multiple LMD or RMD.

③rd phase compiler (SDT) is part of 3rd part of compiler.

↓ PT (parse Tree)

SA → Semantic Analysis phase

↓

Annotated parse Tree (CFG + Semantic Rule)

$S \rightarrow S\#A \mid A$ { S.val = S.val * A.val }

$A \rightarrow A \& B \mid B$ { A.val = A.val + B.val }

$B \rightarrow id$ { B.val = id.lval; }

SDT

- Top-down
  - S
  - A S {I} Colony with Rule
  - Start from top only.
- Bottom up.
  - S
  - A S
  - Not with Rule
  - Start from top only.

Types of SDT

① S-Attributed SDT
- Based on synthesized attributes
- LHS: parent
- RHS: child
- ② Bottom up parsing
- ③ semantic Rule written in RHS of rightmost position in RHS

A → XYZ {printf}

parent taking value from child is known as Synthesized.

② L-Attributed SDT
- Based on synthesized & inherited attribute (parent, Left sibling values)
- ② use Top down parser.
- ③ Semantic Rules written anywhere in RHS

child taking values from parent inherited. also from + Sibling values.

Eg check whether below Attributes are are S-ATT SDT or L-ASDT

Inherit Inherit Synthesized.

① $A \to LM \{ L.i = l(A.i), M.i = m(L.s), A.s = f(M.s) \}$

② $A \to QR \{ R.i = r(A.i), Q.i = q(R.s); A.s = f(Q.s) \}$

Inherit Inherit Synthesized.

Q taking value from R sibling ∴ Not L SDT

Not in S-A SDT

Not in L-A SDT

(5) th phase of Compiler.

Intermediate code generation. (LA SA SA) different (LA SA SA) different

① machine Independent

P1 Processor ICG

P2 processor ICG

②

Methods for Intermediate Code Generation.

~~① Machine Independent~~

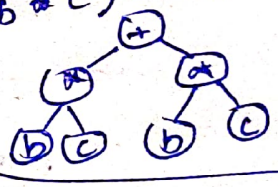
① Abstract Syntax Tree (AST)

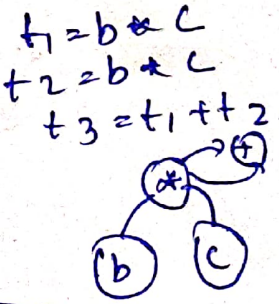
② Direct Acyclic graph (DAG)

③ postfix

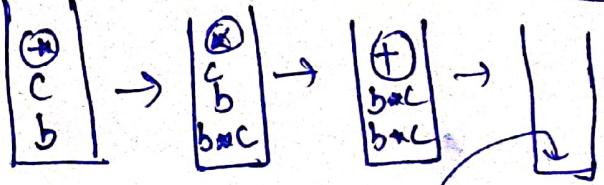
④ 3-Address Code → popular.

Eg (b*c) + (b*c)

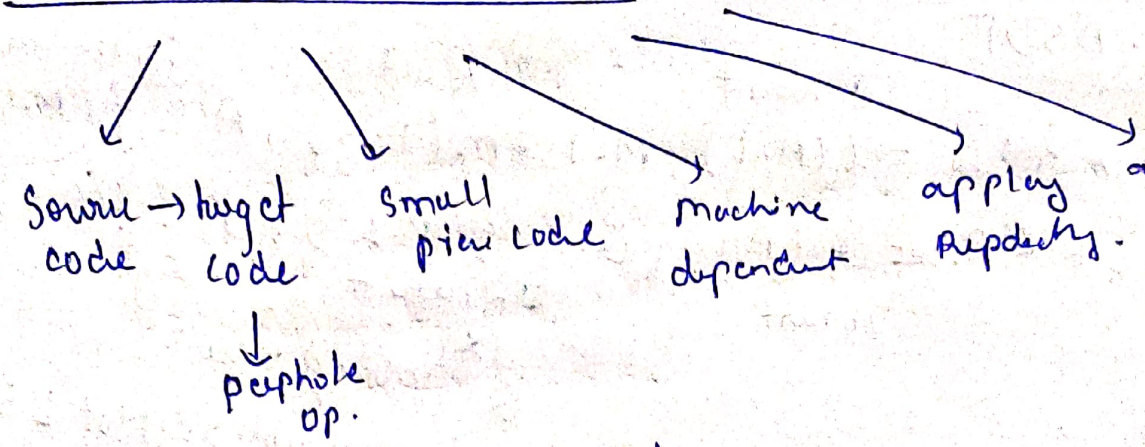




bc* t



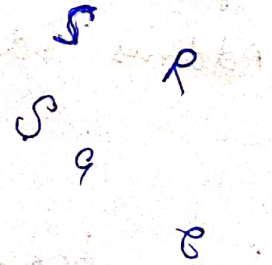
Peephole optimization.



Refers ss (Screen short)

1. Redundant load & store Removal — Mov a, b / Mov b, a.
2. Strength Reduction ~~Operation~~ — $n^2 = n * 2$, $n * 2 = $ left shift
3. Simply algebric Expression. — $a = a + 0$, $a = a * 1$ } don't use this type of instruction
4. Replace slower with faster instruction { Add R, #1 → INC(R)
5. Dead ~~De~~code Elimination

```
int void (void) {
    muir
        int c;
        c = a * 10;
        return 0;
        b = 30;
Removal { b = b + 10;
Y. The unused code
```



(5) In phase of compiler.

Code Optimization

Platform dependent

→ peephole Optimization

using piplining
- → Instruction level parallism
- → data level parallism
- → Redundant Resources

Platform Independent.

→ Loop optimization.
loop unrolling, code motion/movement, frequency reduction, loop jamming

→ constant folding (x = 3.14)

→ constant propogation

→ common subexpression elimination.

a = b * c

c = bac

Reuse a

Compile value dedo rather than run time

Loop optimization

loop

① Code Motion (frequency Reduction)

use misstatement only.

```
a=100
while(a>0){
    c=e*f
    if(a%2==0){
        print(i);
    }
}
```

not required. inside of loop.

② Loop fusion.

```
for(i=0; i<100; i++){
    a[i]=i;
}
for(i=0; i<100; i++){
    a[i]=i;
}
```

merge these both loops

③ loop unrolling

```
for(int i=0; i<100){
    printf("hiksha");
}
```

↓

```
printf("hiksha")
;
;
100 printf("hiksha")
```

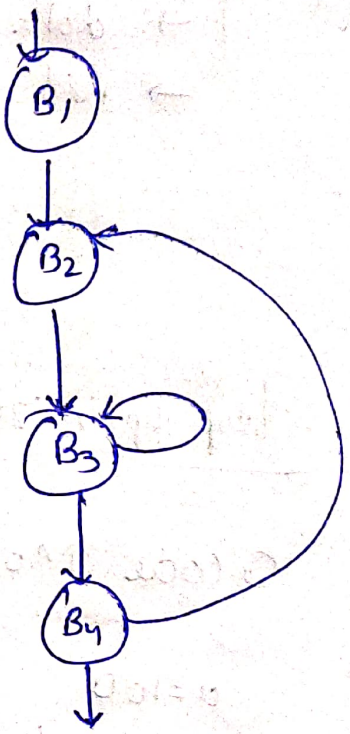
Control Flow graph of Intermediate code.

Step1: Find leaders of the block.

1.1 first → leader
1.2 address of any conditional/unconditional goto → leader
1.3 any next line after conditional/unconditional goto → leader.

Eg:

1) i = 1 — B_1
2) J = 1 — B_2
3) $t_1 = 5 * i$
4) $t_2 = t_1 + J$
5) $t_3 = 4 * t_2$ — B_3
6) $t_4 = t_3$
7) $a[t_4] = -1$
8) $J = J + 1$
9) if (J <= 5) goto (3)
10) $i = i + 1$ — B_4
11) if (i < 5) goto (2)



Step2: Draw CFG (Control Flowgraph)

Nodes = 4
edges = 5

* Basic Block ≈ to Control Flow graph.

both are used interchangeable.

→ Collection of intermediate code statement in a group.

without considering start, end

Nodes ⇒ start, end + Branch to basic block

Scope of optimization

local (within the block)

global. (between the blocks)

* Symbol Table → Different data structure for implementation of symbol

Eg ⇒
```
main() {
    char a[5];
    int n;
}
```

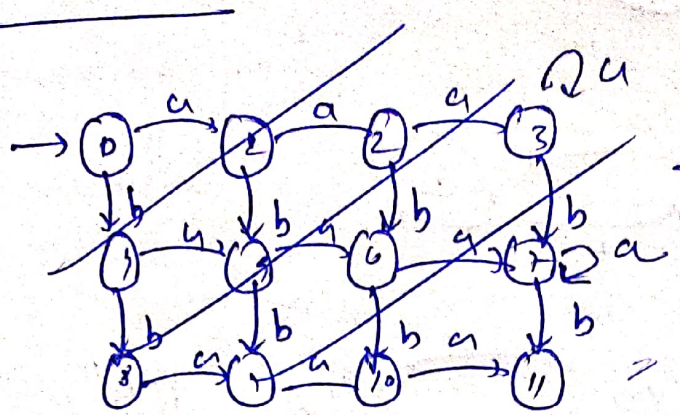
ordered array, unordered array, ordered linked list, unordered linked list

search time, hash tables → O(1)

O(log n) ← O(n)

Name	Type	Size	Dimension	1D	Lineofarr	Addr
a	int	4	0	3	(6-0)	1001x
n	char	5	1	2	- -	1005

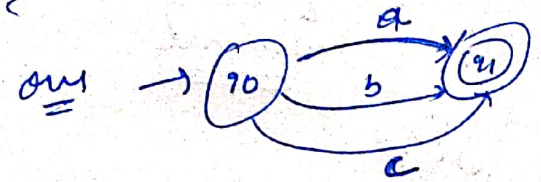
3a's 2b's

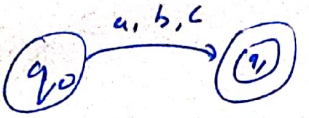


DFA TO Regex

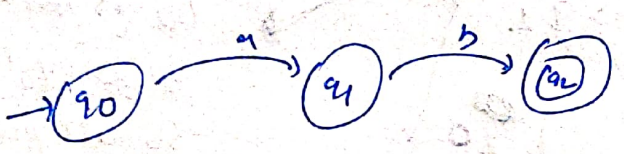
Finite automata → Regular Expression.

DFA → RE





$\Rightarrow (a+b+c)$



Original

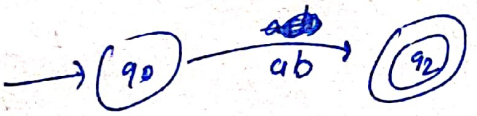
S1 ✗ (No incoming)

S2 ✗ (No outgoing)

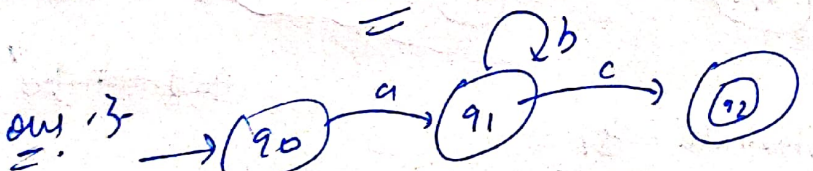
S3 ✓ (✓)

q_1 eliminate.

Steps:



ab

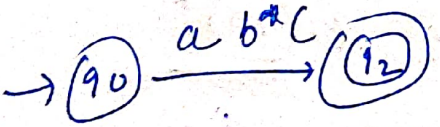


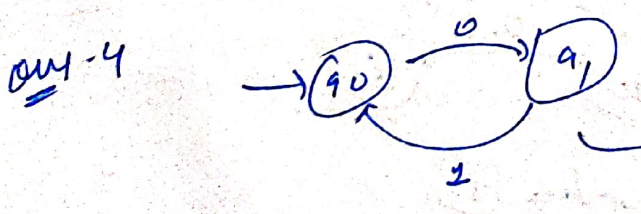
S_1 ✗

S_2 ✗

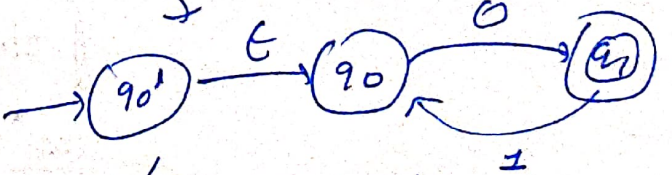
S_3

q_1

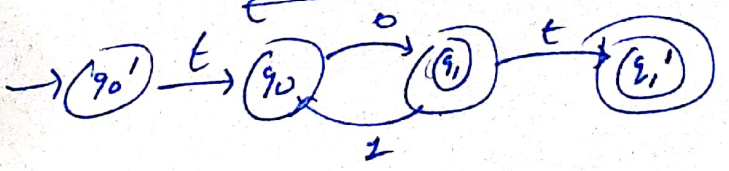




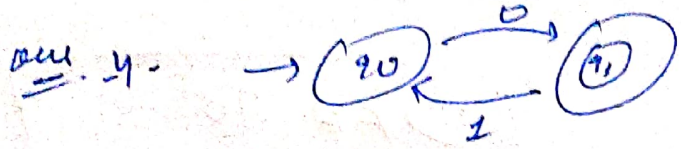
S_1 ✓ (inorm)



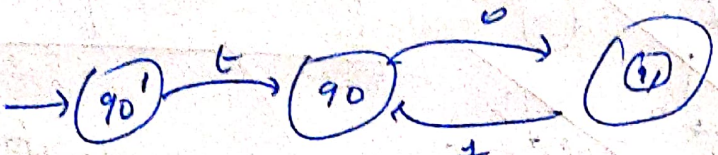
S_2 ✓ (inorm).



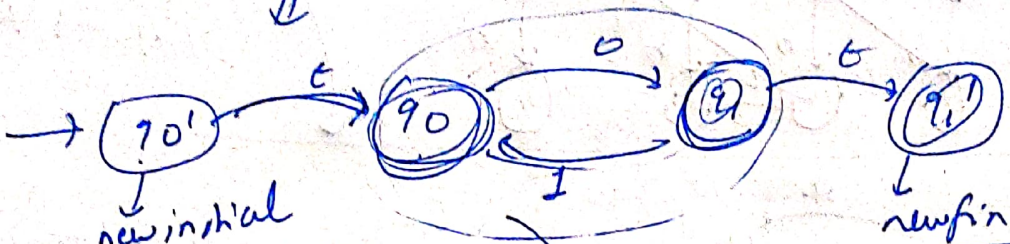
Sequence-1-



q1:



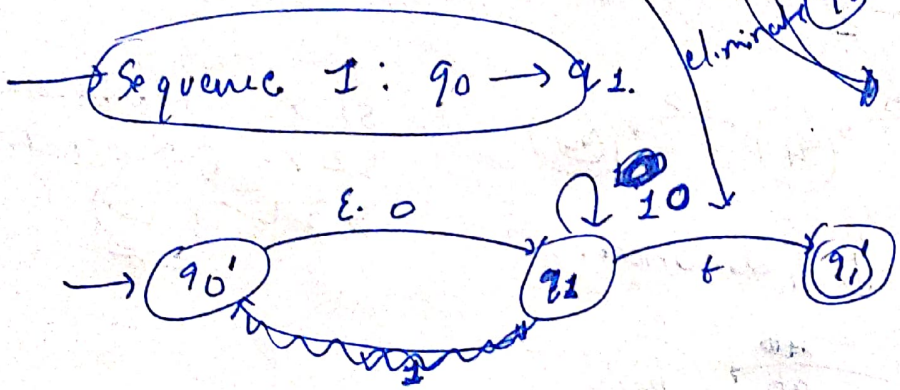
q2:



new initial

new final

q3:



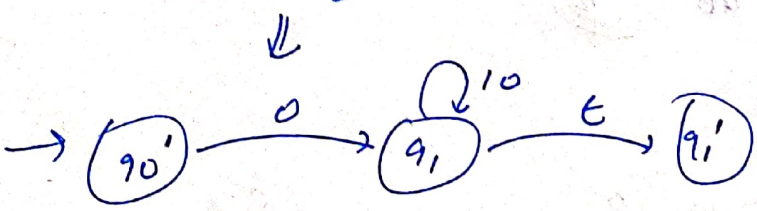
q0 is bridge blw q0' & q1
So all path blw them must be intact

q0' → q0'

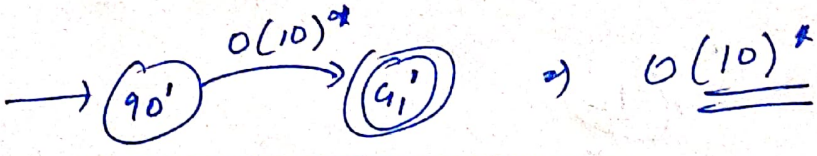
q0' → q1

q1 → q1

~~q1 → q0'~~

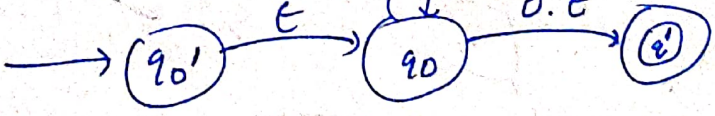


eliminate q1

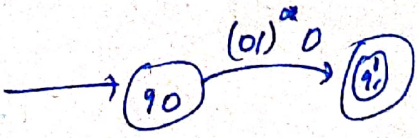


⇒ $0(10)^*$

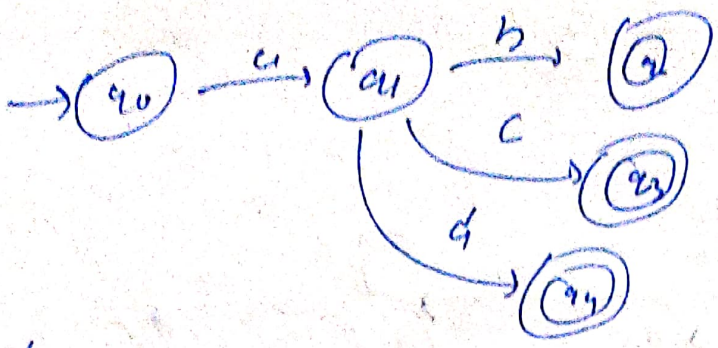
Sequence-2-



eliminate q0

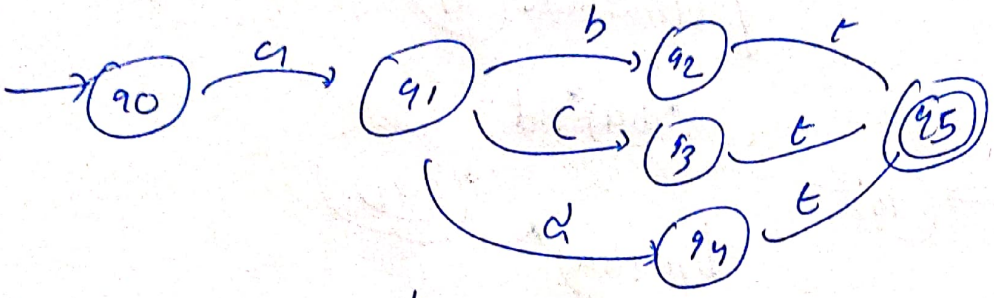


Q11.

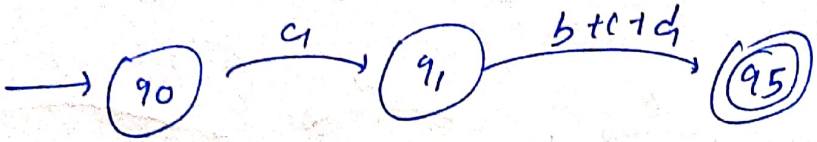


S1: X
S2: ∞ (followed if more than 1 final state))

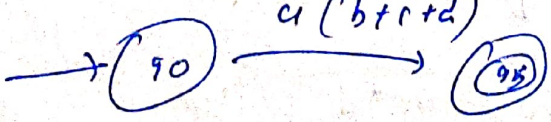
S2':



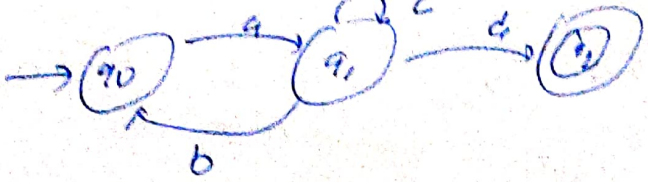
⇓ eliminate q_2, q_3, q_4 → b+c+d.



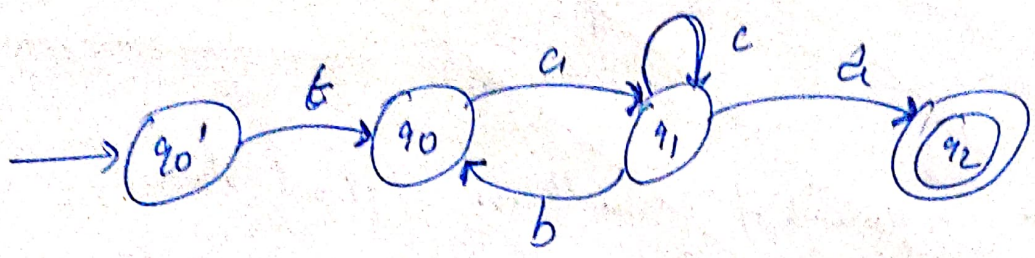
⇓ eliminate q_1



Q11



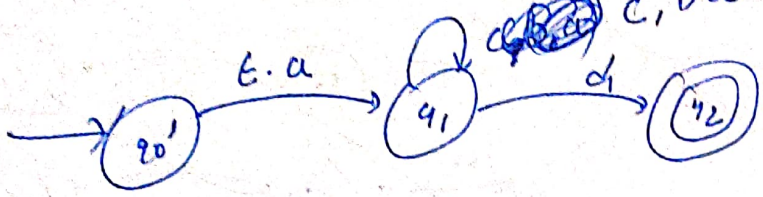
S1:



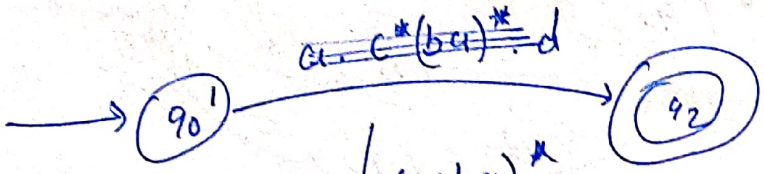
S2:

eliminate q_0

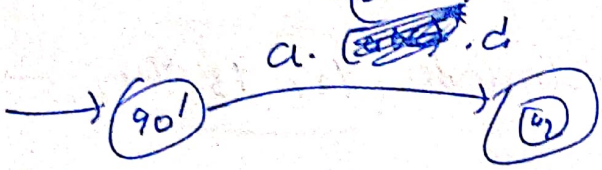
Sequence 1



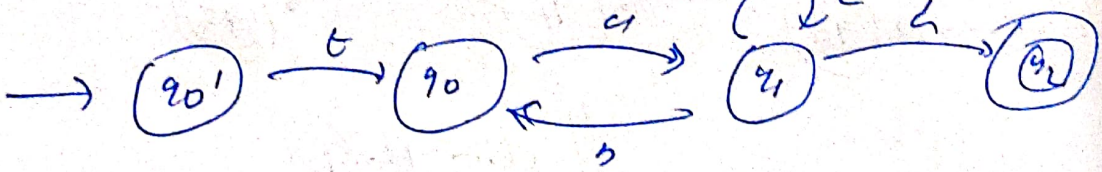
eliminate q_1



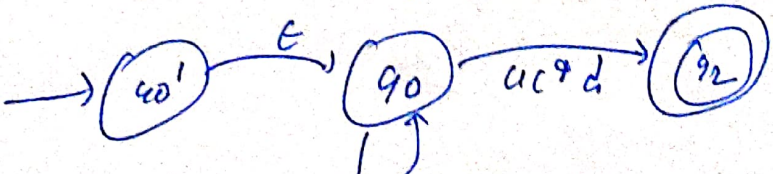
$(c+ba)^*$



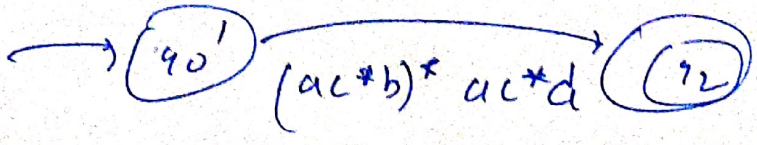
Sequence 2



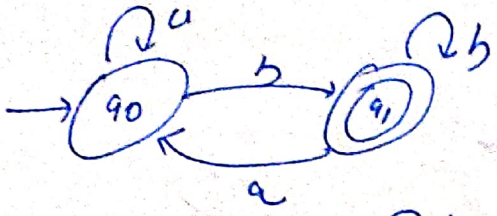
eliminate q_1



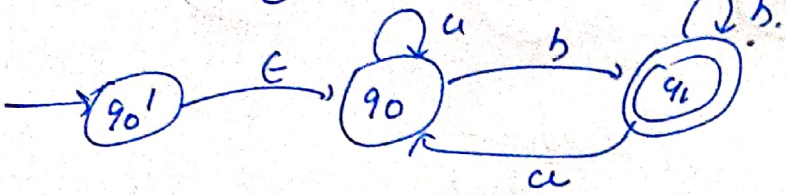
eliminate q_0



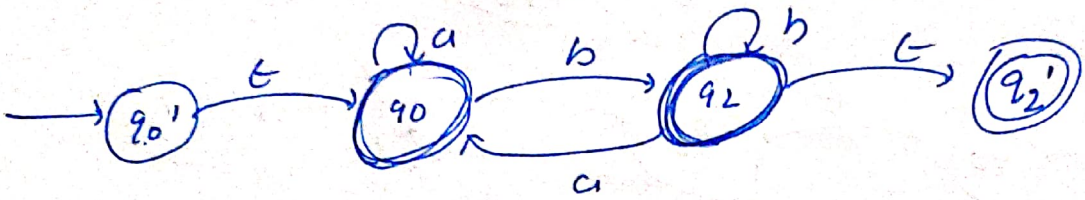
Qu4



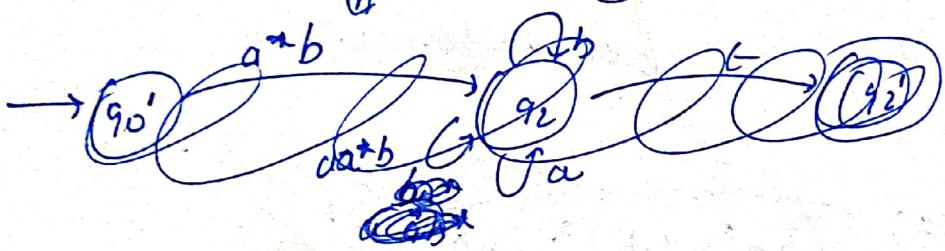
S1:



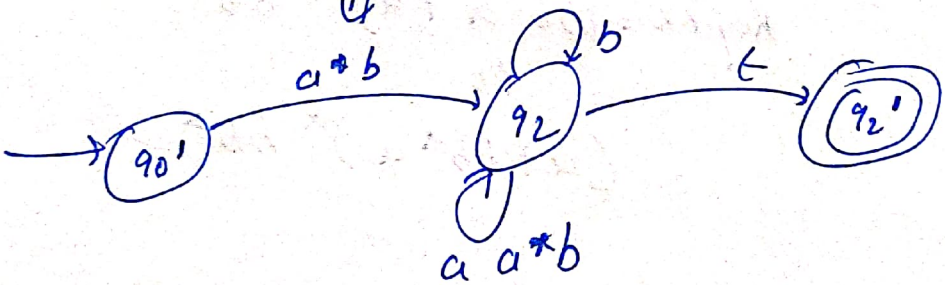
S2:



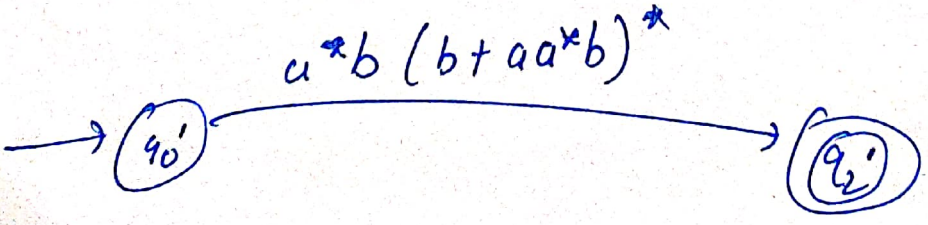
eliminate q_0



eliminate q_2

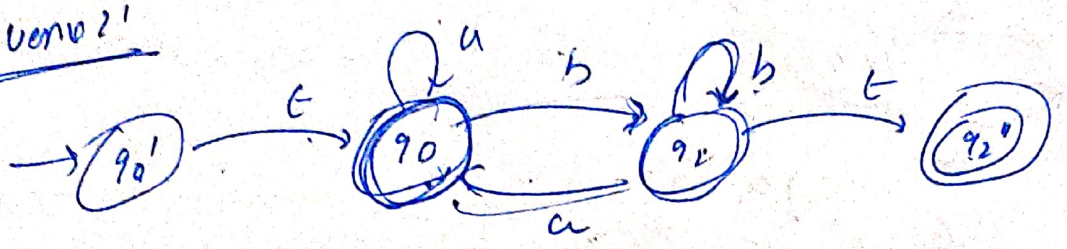


eliminate q_2.

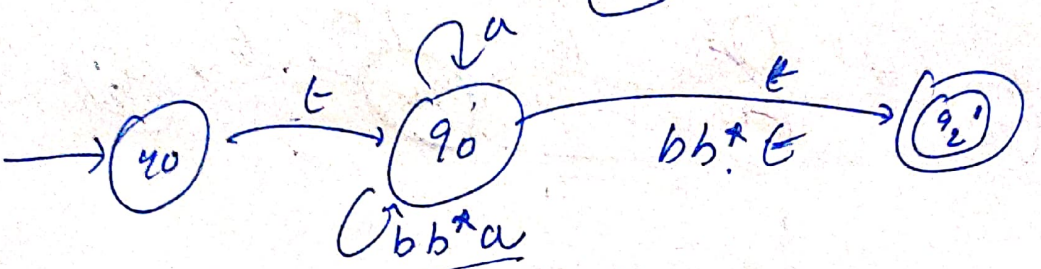


Regex $\Rightarrow a^*b(b^* + aa^*b)^*$

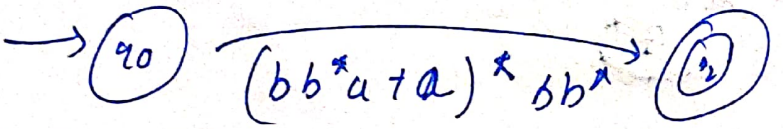
Sequence 2:



⇓ eliminate q_1



⇓ eliminate. a.



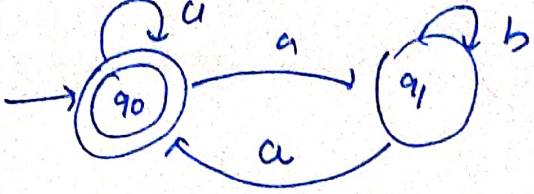
⇓

Regular expression $= (bb^*a + a)^* bb^*$

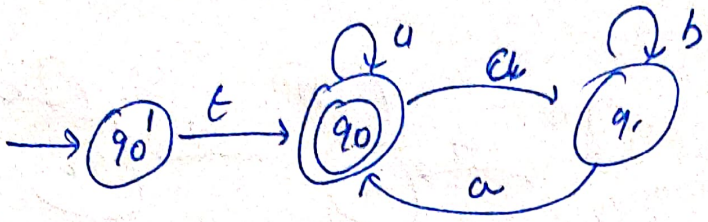
q_2

$(a + bb^*a)^* bb^*$

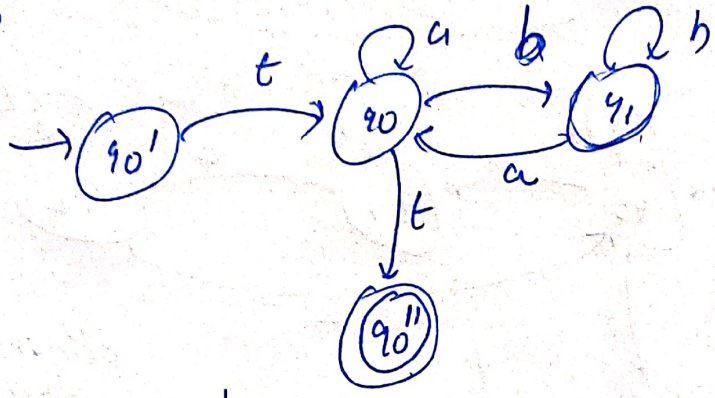
Qu =

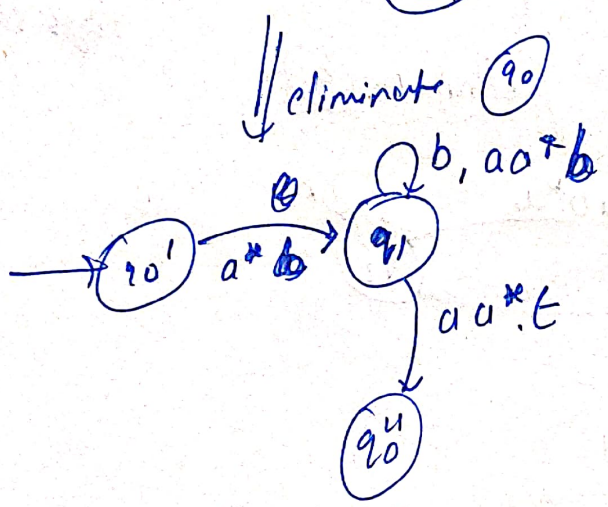


S1: →



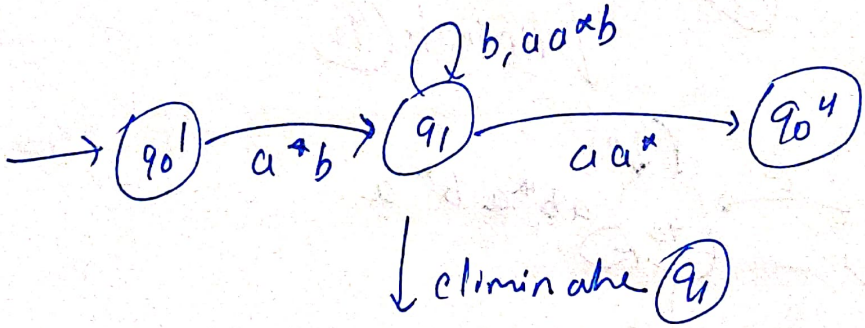
S2: →



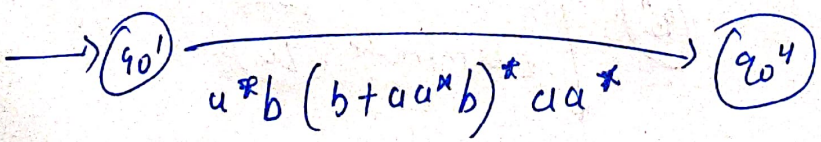


Important rule in Arden

$q_0' \rightarrow q_1$
$q_1 \rightarrow q_1$
$q_1 \rightarrow q_0''$

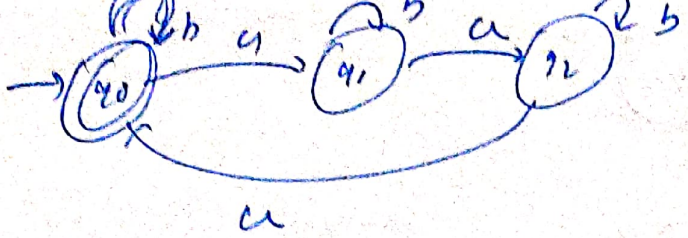


eliminate q_1

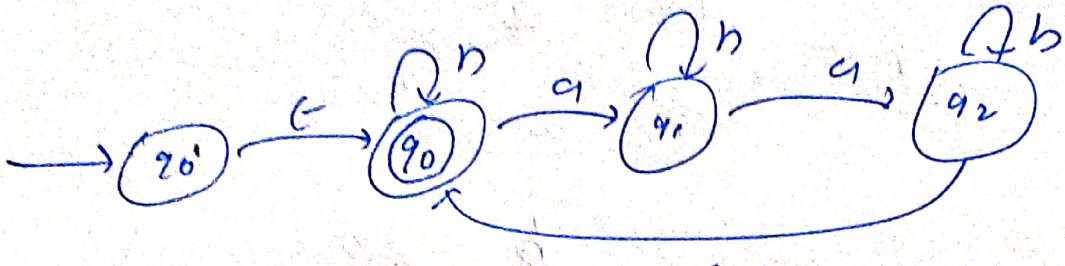


Regex: $a^*b\,(b+aa^*b)^*\,aa^*$

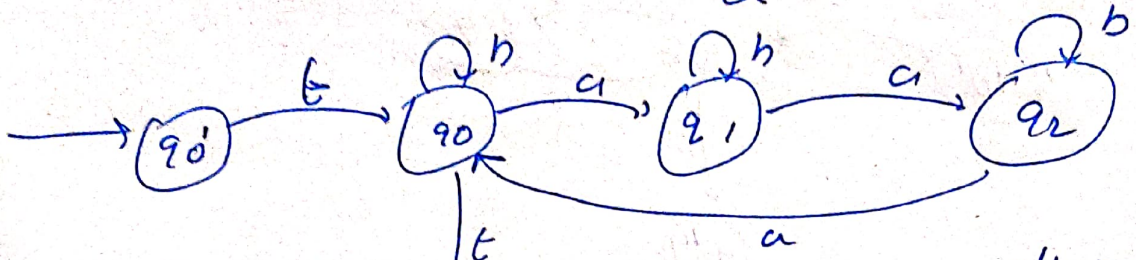
Qu =



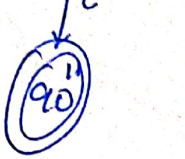
S1:



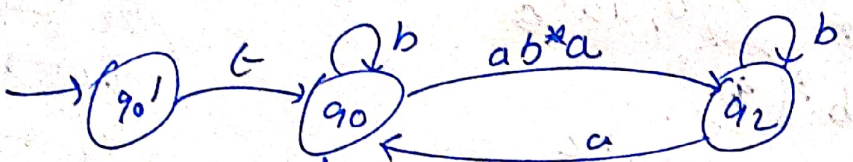
S2:

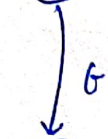


eliminate clear keys state



eliminate q_1.

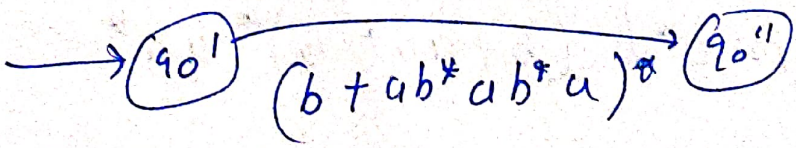




eliminate q_2

$q_0 \to q_0$

eliminate q_0.



Regular Expression to Finite Automata. (R.E to F.A)

Flowchart

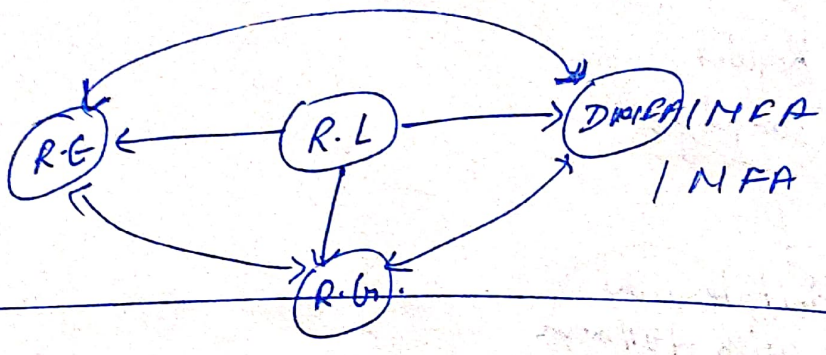
R.L → Central Concept

R.E

R.L → accept karne ke liye DFA or, NFA ^.

R.L → generate R.G

R.L → Represent R.E



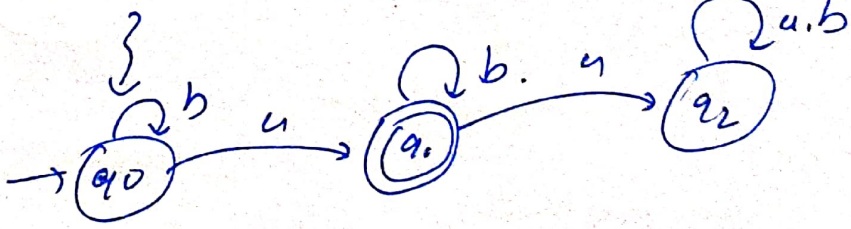
R.E to F.A

R.E finitely converted to N.F.A with Null move

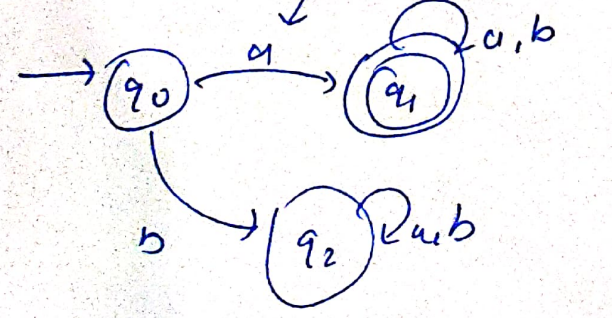
RE ⇒ R.L Language ⇒ Finite Automata → Bat Algo.

Qy b^*ab^* ⇝ R Language = exactly 1 a is.

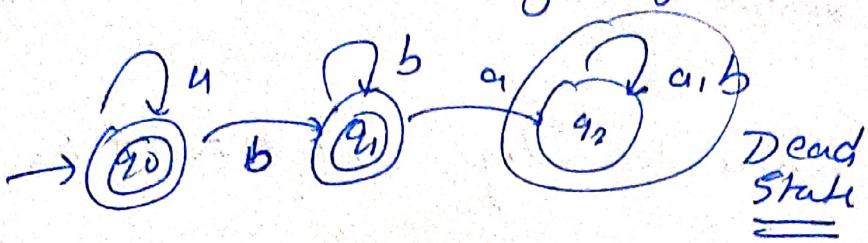
R.E



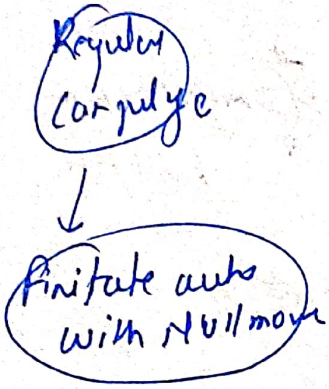
Qy. $a(a+b)^*$ ⇝ starting with a.



Ques 3- $a^* b^*$ → any No. of a's followed by any any No of b's.

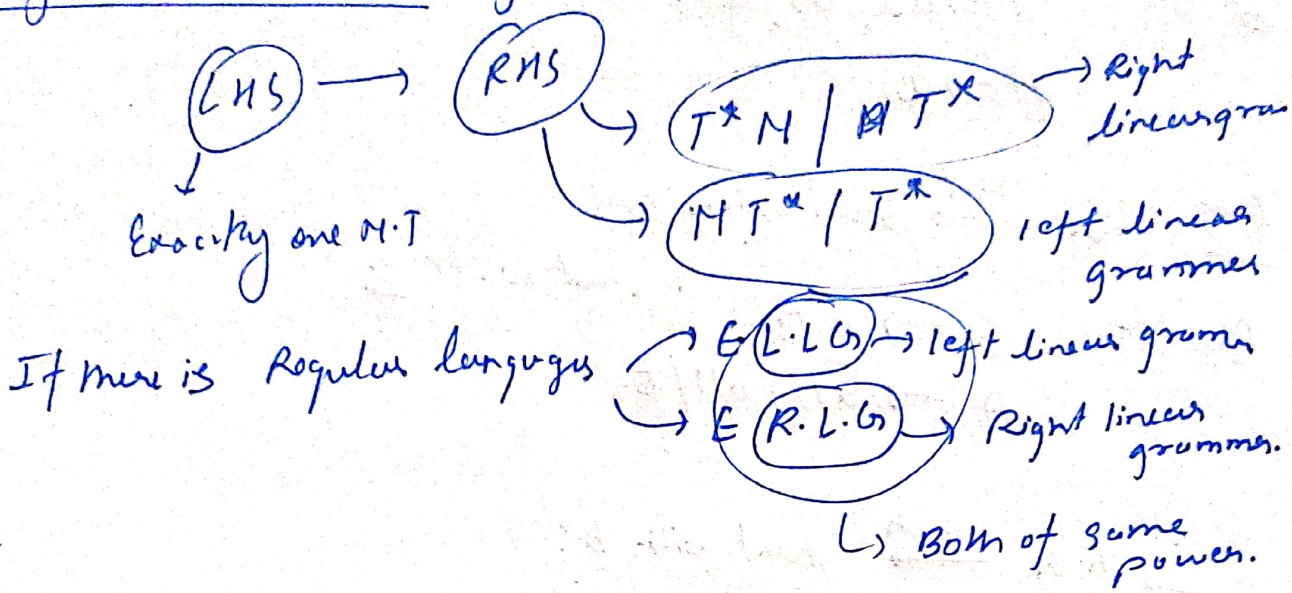


Ques If R_1 & R_2 are R.E so is $R_1 + R_2$



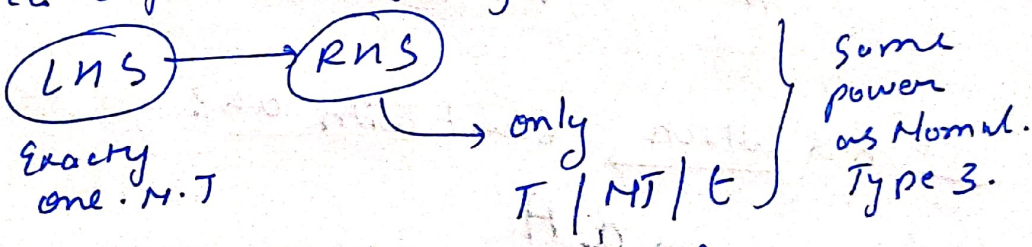
(3) Construction of Grammers.

(A) Regular Grammar. (Type-3).



For Construction purpose.

we use Restricted Definition of Type 3 (RG) grammar.



Ques.1- Construct R.G which genrate all string over a & b?

Ans 1. ∈, a, b, ab, ba - - - - .

$S \rightarrow aS / bS / \in$

Theorem: For any Regular language we have infinite Regular grammar generating it.

Ques-2- Construct R.G which genrate all string over a & b which ends with a?

Ans-2- a, aa, ba

$S \rightarrow aS / bS / a$

Que-3 - Construct R.G which ~~accepts~~ generate all string over a & b which ends with aa?

Ans-3. $S \to aS / bS / aA$

$A \to a$

Q-4 R.G must start with a.

Ans-4. $S \to aA$ → start with a

$A \to aA / bA / \epsilon$

Q-5 R.G which start with b?

Ans-5 $S \to bA$

$A \to bA / aA / \epsilon$

Q-6 R.G which start with aa?

Ans-6 $S \to aA$

$A \to aB$

$B \to aB / bB / \epsilon$

Q-7 R.G which start with a & end with a?

Ans-7 $S \to aA / a$

$A \to aA / bA / a$

Q-8 R.G which start a & end with b?

ab

a a b

Ans-8. $S \to aA$

$A \to aA / bA / b$

Q-9 R.G start & end with different letters?

Ans 9.

$S \to aA \mid bB$

$A \to aA \mid bB \mid b$

$B \to aB \mid bB \mid a$

Q-10 R.G start & end with same letter?

Ans-10 $L = \{a, b\}$?

$S \to aA \mid bB \mid a \mid b \mid \epsilon$

$A \to aA \mid bA \mid a$

$B \to aB \mid bB \mid b$

Q-11 $S \to aS \mid bS \mid a \mid b$

Ans-11 (1) All possible string over a & b which ends with a & b

(2) ——— start with a & b

(3) ——— where length is atleast one

(4) none

Q-12 R.G construct aa as substring?

Ans-12

$S \to aS \mid bS \mid aA$

$A \to aB$

$B \to aB \mid bB \mid \epsilon$

Q-13 R.G construct bb as substring?

Ans-13

$S \to aS \mid bS \mid bA$

$A \to bB$

$B \to aB \mid bB \mid \epsilon$

Q-14 Construct R.E $\xrightarrow{a}$ as substring?

Ans-14

$S \to aS \mid bS \mid aA$

$A \to aA \mid bA \mid \epsilon$

Equivalent

$S \to bS \mid aA$

$A \to aA \mid bA \mid \epsilon$

as language generated is same by both grammar that is Atleast one a.

Q-15 Construct R.E $\xrightarrow{no(a's)}$ divisible by 3?

Ans-15

$S \to aA \mid \epsilon \mid bS$

$A \to aB \mid bA$

~~$B \to aB \mid bB \mid \epsilon$~~ $B \to aS \mid bB$

Q-16 Construct R.E $\xrightarrow{no(a's)\ mod\ 3 = 1}$

Ans-16

$S \to aA \mid bS \mid \epsilon$

$A \to aB \mid bA \mid \epsilon$

$B \to aBS \mid bB$

$L = \{a, a^4 \ldots\}$ $no's(a's)\ mod\ 3 = 1.$

Q-17 Construct R.E $\xrightarrow{no(a's) mod 3 = 1 \& 2.}$

Ans-17

$S \to aA \mid bS$

$A \to aB \mid bA \mid \epsilon$

$B \to aS \mid bB \mid \epsilon$

Summary

$S \to \epsilon \rightsquigarrow a^0, a^3, a^6, a^9 \ldots$

$B \to \epsilon \rightsquigarrow a^1, a^4, a^7, a^{10} \ldots$

$C \to \epsilon \rightsquigarrow a^2, a^5, a^8, a^4 \ldots$

Q-18 n a's divisible by 5? $n(a's) \% 5 = 0$

Ans 18

$S \to aA / bS / \epsilon$

$A \to aB / bA$

$B \to aC / bB$

$C \to aD / bBC$

$D \to aS / bD$

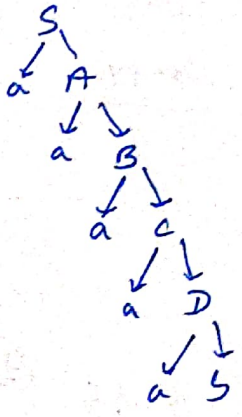
R5, Removed 0

R6, Removed 3

R2

R3

Removed 4.



Q-19 Construct R.G $\xrightarrow{\text{atleast}}$ 1a's?

Ans-19.

$S \to bS / aA$

$A \to aA / bA / \epsilon$

Q-20 Construct R.G atleast 2a's?

Ans-20

$S \to bS / aA$

$A \to aB / bA$

$B \to aB / bB / \epsilon$

$L = \{aa,$ aa, aaa, aa, a

Q-21 Construct R.G exactly 1a's?

Ans-21

$S \to bS / aA$

$A \to \epsilon / bA$

$L = \{a, ab, ba, bba\}$

Q-22 Construct R.G exactly 2a's?

Ans-22

$S \to bS / aA$

$A \to aB / bA$

$B \to \epsilon / bB$

$L = \{aa, \ldots\}$

Q.23 R.L almost 1's?

Ans.23 $S \to bS / aA / t$

$A \to t / bA$

Q.24 R.L almost 2's?

Ans.24 $S \to bS / aA / t$

$A \to aB / bA / t$

$B \to t / bB$

Q.25 ⑦ Category finite language.

Ans.25 $L = \{ a.ab, aab \}$.

$S \to a / aA / aB$

$A \to b$

$B \to aC$

$C \to b$

Theorem: Every finite language Regular.

Finite → Regular.

Why used Restricted definition of Type-3

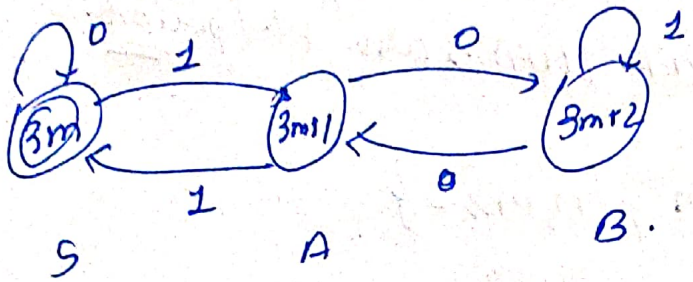
F.A ⇒ Restricted Definitions of Type-3.

easily interconversion able.

Equivalence of Regular grammar & finite automata.

Q) Construct RG which generate all those binary No whose decimal equivalent is multiple of 3?

Sol: Step1: Create FA:



S A B.

Remainder 0 Remainder 1 Remainder 2

Step2: Create R.E for F.A as same as S (Transition function)

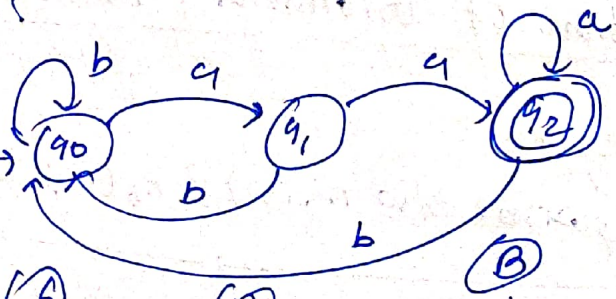
$S \rightarrow 0S / 1A / \epsilon$

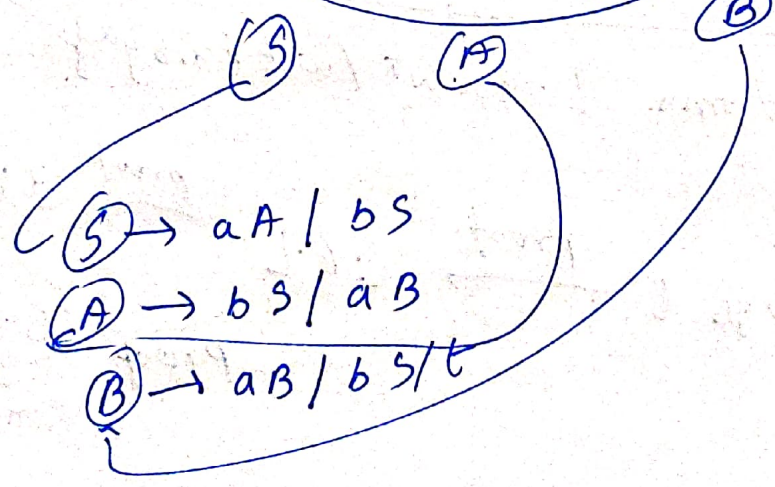
$A \rightarrow 1S / 0B$

$B \rightarrow 1B / 0A$

Ques aa as substring?

Sol: Step1: FA:





$S \rightarrow aA / bS$

$A \rightarrow bS / aB$

$B \rightarrow aB / bS / \epsilon$

aa
ab
aaa
aab

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

↓

Regular language is Not possible for ~~Context~~ production.

↓

(Finite automata also Not possible) DFA / NFA is also Not possible.

Ques $L = \{a^n b^m, n, m \geq 0\}$

Regular language ⟹ $S \to aS \mid bA \mid \epsilon$

$A \to bA \mid \epsilon$

Ques $L = \{a^n b^m \mid n < m\}$ } Ja par bhi hisab kitab Rahne wali but ayegi unka par Regular Grammar (T-3) ki not possible.

Ques $L = \{a^n b^m \mid n \neq m\}$ }

(B) Construction of Grammars. (Context free grammar)

→ Categories.

① Comparison b/w $n_a(w)$ & $n_b(w)$ & a's b's are ordered.

② ——————→ Not ordered.

③ Palindromes. $L = \{ww^T \mid w \in \{a,b\}^*\}$.

Language L ——proof——→ C.F.L ——proof——→ ∃ PDA (Right)

C.F.L ——proof——→ C.F.G

↓

Context Free Grammar.

Standard way
$S \to aSb / t$

Qu-1 $a^n b^n / n \geq 0$

Ans-1- $S \to aSb / t$

Qu-2- $a^n b^n / n \geq 1$ $L = \{ab, aabb \ldots\}$

Ans-2- $S \to aSb / ab$ → min length string language.

Qu-3 $a^n b^n / n \geq 2$

Ans-3 $L = \{a^2b^2, a^3b^3 \ldots\}$.

$S \to aSb / a^2b^2$ → min length string is know as stopper.

Qu-4 $a^n b^{n+1} / n \geq 0$

Ans-4 $L = \{b, ab^2, a^2b^3 \ldots\}$

$S \to b / aSb$

$a^n b^{n+1}$

$a^n b b^n$

$S \to aSb / b$

Qu-5 $a^{n+1} b^n / n \geq 0$

Ans-5 $L = \{a, a^2b, a^3b^2 \ldots\}$.

$S \to aSb / a$

rewrite as.

$a^{n+1} b^n$

$a^n . a b^n$

$S \to aSb / a$

Qu-6- $a^n b^m / n = m-1$ & $m \geq 1$

Ans-6 $L = \{b, ab^2, a^2b^3 \ldots\}$.

$S \to aSb / b$

method → Simply it $n = m-1$

$m = n+1$ → replace in question

$a^n b^{n+1} / n \geq 0$

Qus-7 $a^{n+1}\ b^{n+2} \mid n \geq 0$

Ans $L = \{ab^2, a^2b^3 \ldots\}$.

method 1:

$S \rightarrow aSb \mid ab^2$

or

method 2: $a^{n+1}\ b^{n+2} \mid n \geq 0$

assume, let $n+1 = m$ (from here.)

$\Rightarrow a^m\ b^{m+1} \mid m \geq 1$

$\Rightarrow a^m\ b\ b^m \mid m \geq 1.$

$L = \{a^1b^2, a^2b^3 \ldots\}$.

$S \rightarrow aSb \mid ab^2$

Qus-8 $a^n b^{2n} \mid n \geq 0$

$L = \{t, ab^2, a^2b^4, a^3b^6 \ldots\}$.

$S \rightarrow aSbb \mid t$

a bb

a a bb bb

a a a bb bb bb

Qus-9 $a^{2n} b^n \mid n \geq 0$

$L = \{t, a^2b, a^4b^2 \ldots\}$

$S \rightarrow aaSb \mid t$

aa b

aa aa b b

Qn-10 $a^{3n} b^{5n} / n \geq 0$.

Ans $S \rightarrow aaa S bbbbb / \epsilon$

$S \rightarrow a^3 S b^5 / \epsilon$

Qn-11 $\{a^n b^{2n+3} / n \geq 0\}$ $L = \{a^0 b^3, a b^5 \ldots\}$

Ans-11 $S \rightarrow a S bb / b^3$ → stopper. (min string).

$a^n b^{2n+3}$ $a^n bbb b^{2n}$

Qn-12 $L = \{a^{2n+3} b^{3n+2} / n \geq 0\}$

Ans-12 a^{2n} aaa bb b^{3n} → S

$S \rightarrow aa S bbb / a^3 b^2$

$S \rightarrow aa S bbb / aaabb.$

Qn-13 $L = \{a^m, b^m / m = n\}$. $m, n \geq 0$

Ans-13 $L = \{\epsilon, ab, a^2 b^2 \ldots\}$.

$S \rightarrow a S b / \epsilon$ → stopper.

Qn-14 $a^m b^n / m > n$

Ans-14 $L = \{a, aa, aaa, a^4 \ldots$

$aab, aaab, aaaaab \ldots\}$.

$a \underline{a} b$

$S \rightarrow a Sb / a A$

$A \rightarrow aA / \epsilon$

aaa A bb

↓

aaa ?

Que-15 $L = \{a^m b^n \mid m < n\}$

Ans-B

$L = \{b, bb, bbb \ldots\}$

$abb, a^2b^3 \ldots\ldots$

a, b^3

aa bbb

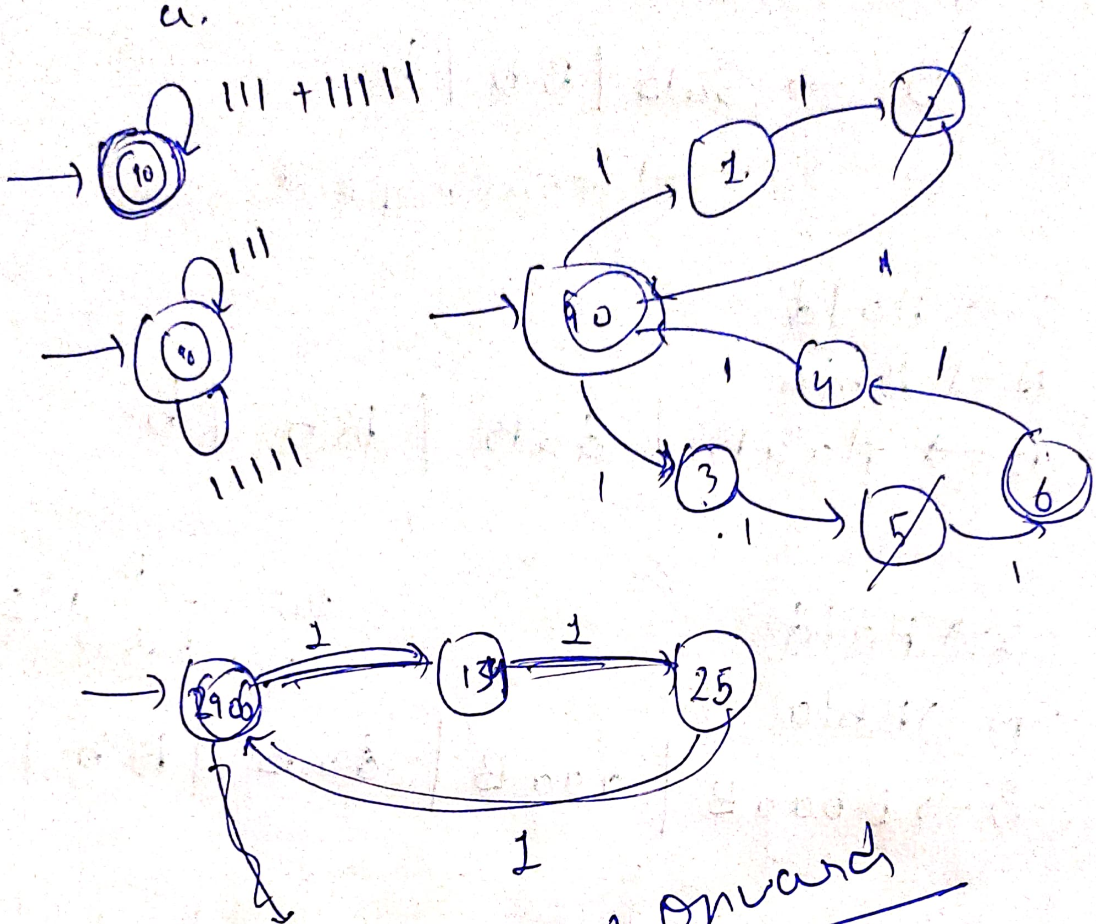
a (bb) b

$S \rightarrow aSb \mid bA$

$A \rightarrow bA \mid t$

Que (111 + 11111)*

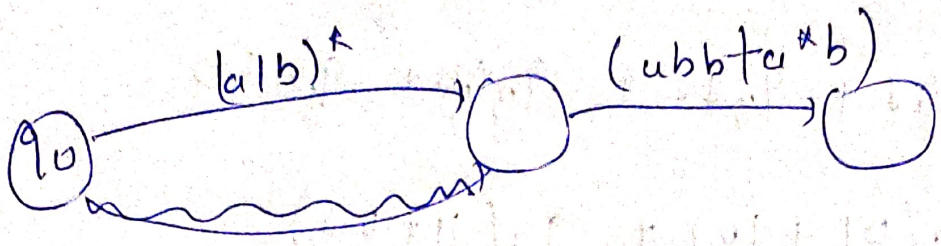
a.

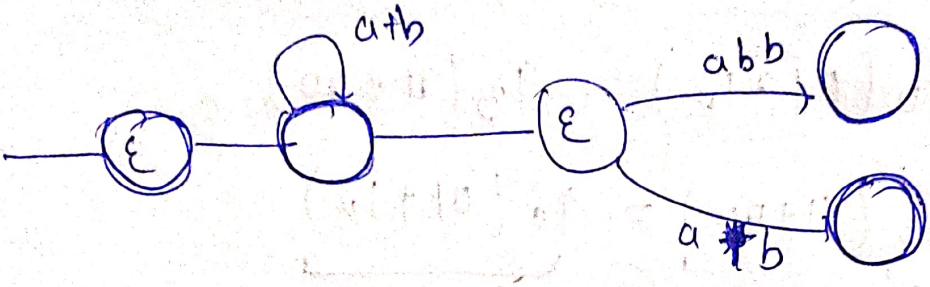


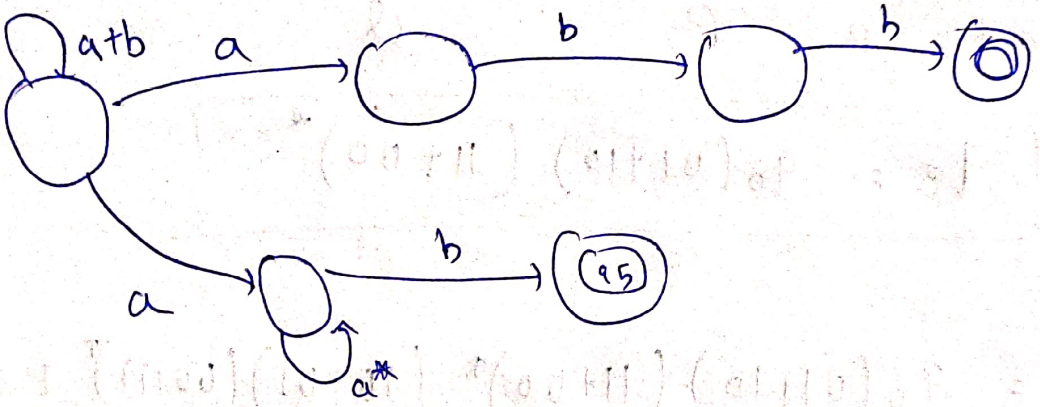
Rough work → Now onward

Practice Questions

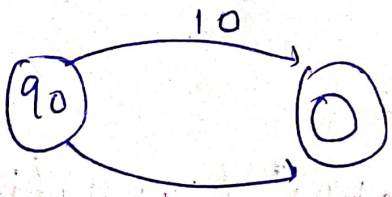
$(a|b)^* . (abb \mid a^*b)$



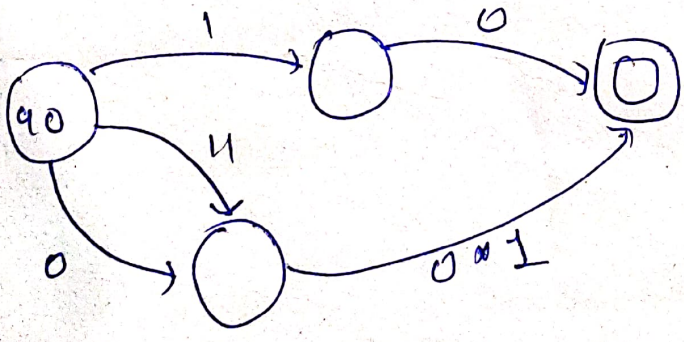


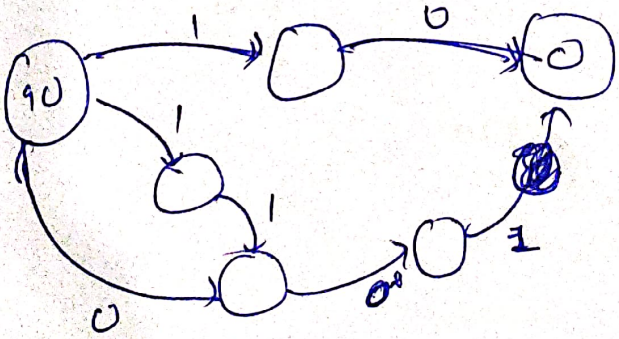


Q4



$(0+11)0^*1$





$q_3 = (q_0 0 + q_3 1) 1 + (q_3 0 + q_0 1) 0$

$q_3 = q_0 01 + q_3 11 + q_3 00 + q_0 10$

$q_3 = q_0(01 + 10) + q_3(11 + 00)$

$q_3 = q_0(01 + 10) + q_3(11 + 00)$

$\underbrace{q_3}_{R} = \underbrace{q_3}_{R}\underbrace{(11 + 00)}_{P} + \underbrace{q_0(01 + 10)}_{Q}$

$$\boxed{q_3 = q_0(01 + 10)(11 + 00)^*}$$

$\underbrace{q_0}_{R} = \underbrace{q_0}_{R}\underbrace{(01 + 10)(11 + 00)^*(10 + 01)(00 + 11)^*}_{P} + \underbrace{(00 + 11)^*}_{Q}$

$q_0 = (00 + 11)^*\left[(01 + 10)(11 + 00)^*(10 + 01)(00 + 11)^*\right]^*$

$q_0 = \epsilon + q_2 1 + q_1 0$ ——— (1)

$q_1 = q_0 0 + q_3 1$ ——— (2)

$q_2 = q_3 0 + q_0 1$ ——— (3)

$q_3 = q_1 1 + q_2 0$ ——— (4)

$q_0 = \epsilon + q_1 0 + q_2 1$ ——— (1).

$q_0 = \epsilon + (q_0 0 + q_3 1) 0 + q_2 1$

$q_0 = \epsilon + q_0 00 + q_3 10 + (q_3 0 + q_0 1) 1$

$q_0 = \epsilon + q_0 00 + q_3 10 + q_0 11 + q_3 01$

$q_0 = \epsilon + q_0 (00 + 11) + q_3 (10 + 01)$

$q_0 = q_0 (00 + 11) + q_3 (10 + 01) + \epsilon$

R R P Q

RP*.

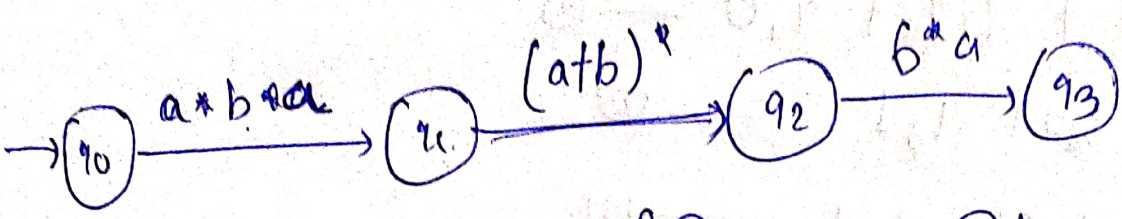
OP*

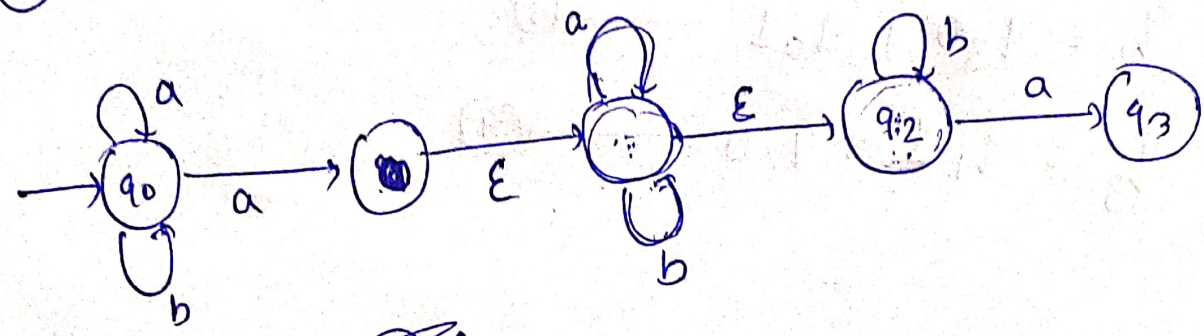
$q_0 = q_3 (10 + 01) + \epsilon \,(00 + 11)^*$

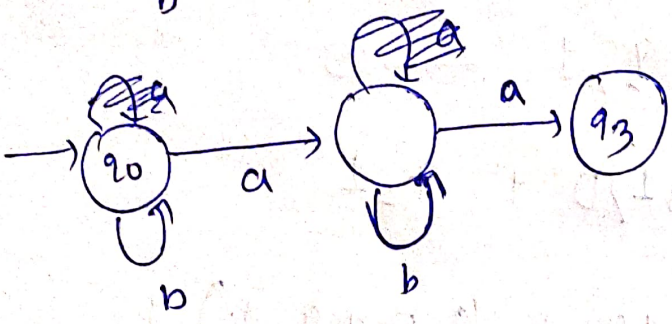
$$q_0 = q_3 (10 + 01)(00 + 11)^* + (00 + 11)^*$$

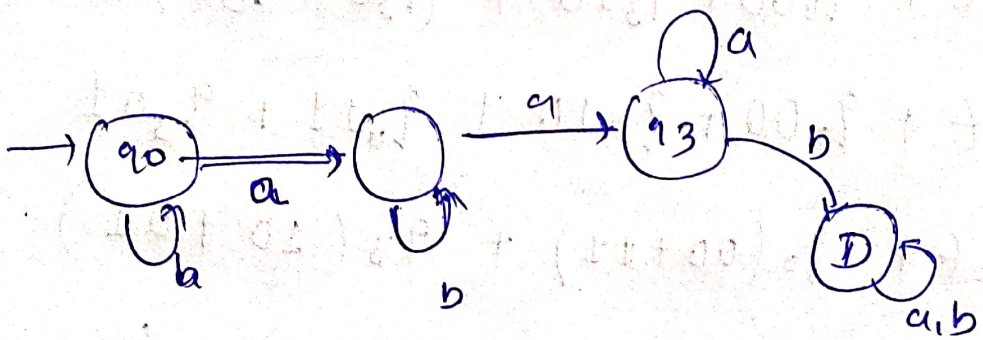
a.b → (f) —

Qu = ~~a* b*~~ a (a|b)* ~~b*~~ a

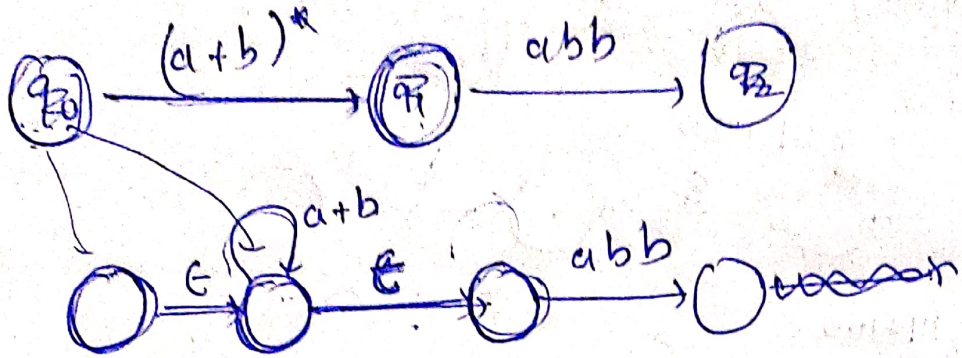








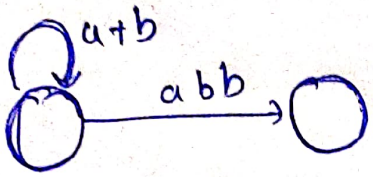
Qu $(a+b)^* abb$

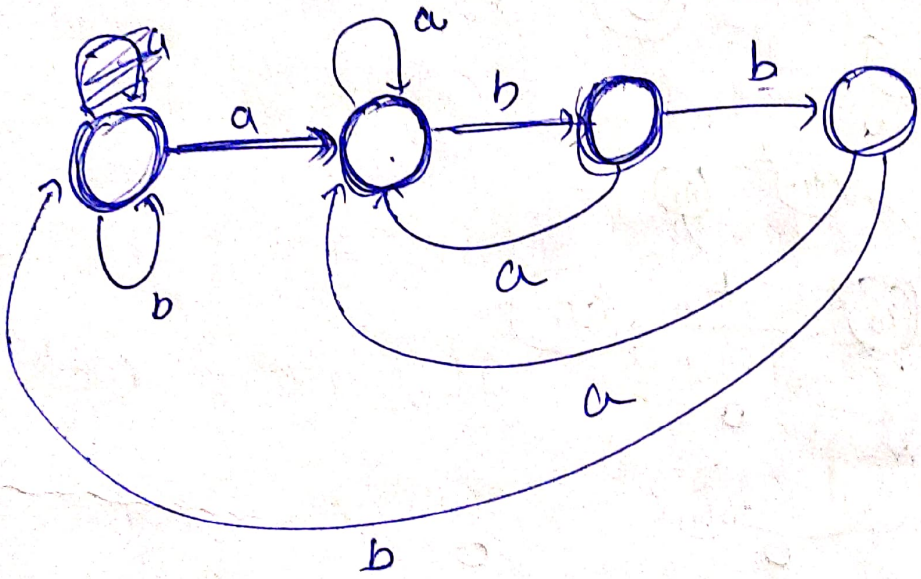


a a b b

a b

a a





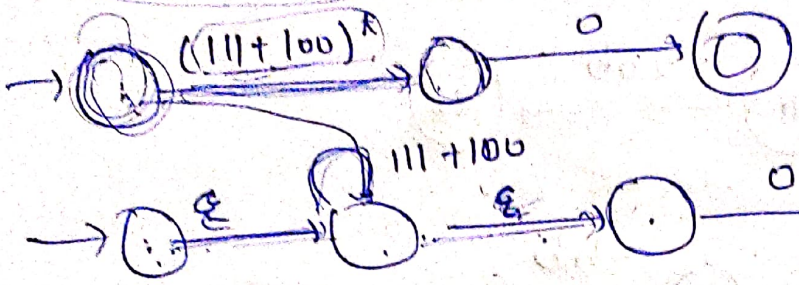
a b a b b

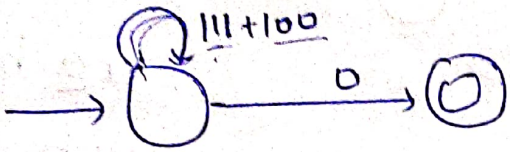
a b b a b b

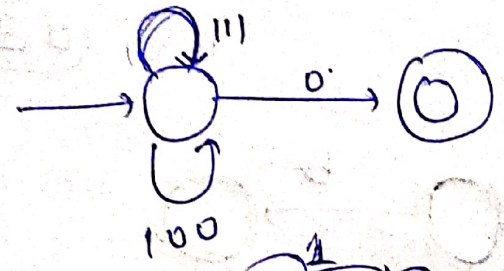
a b b b a b b

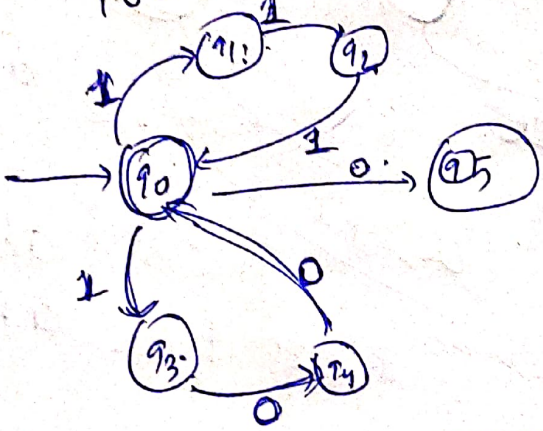
Q

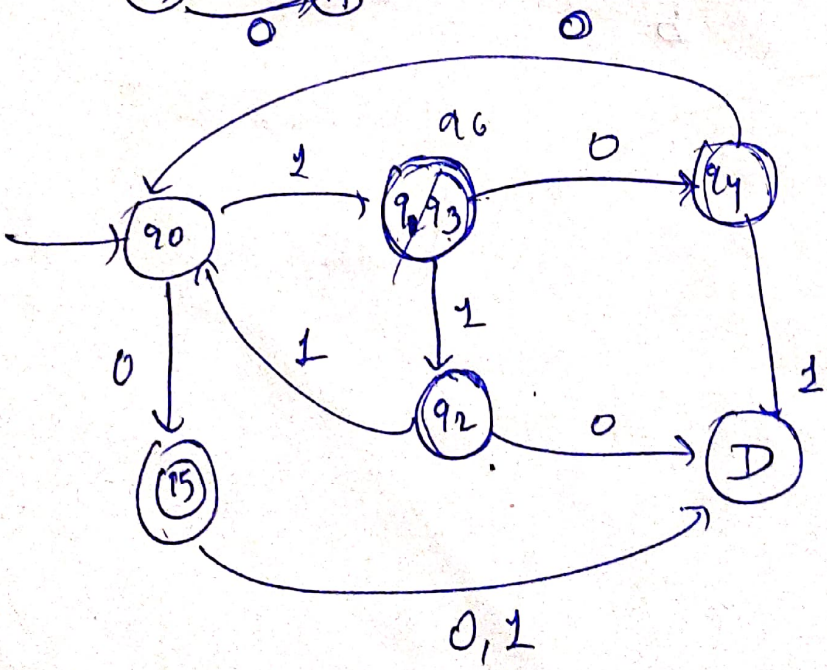
Qu = $((111 + 100))^* 0$





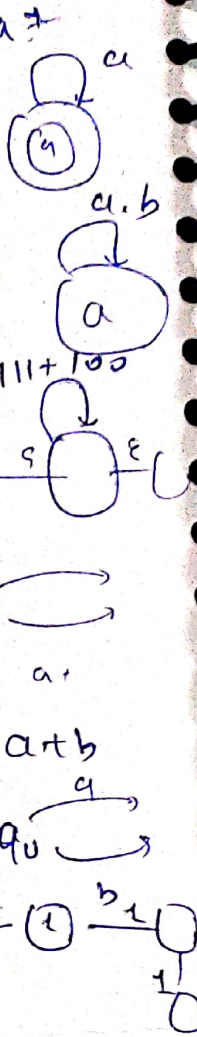


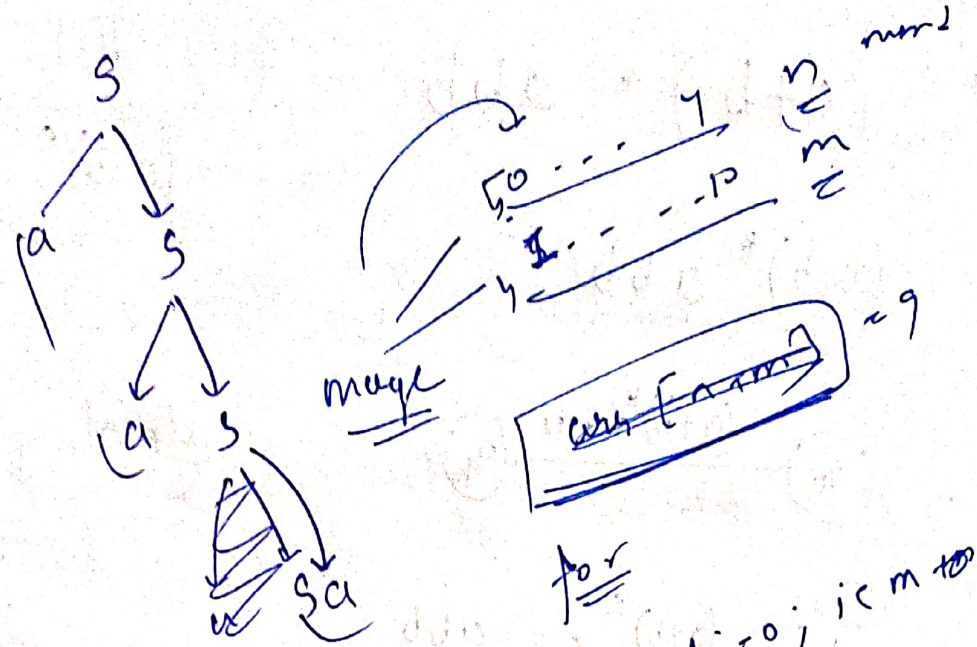




a.b

(a+b)*

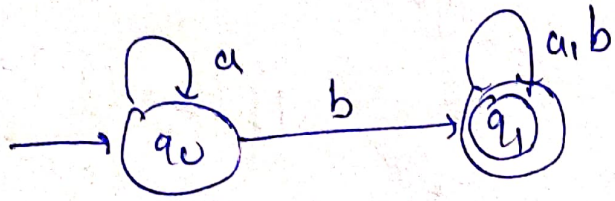




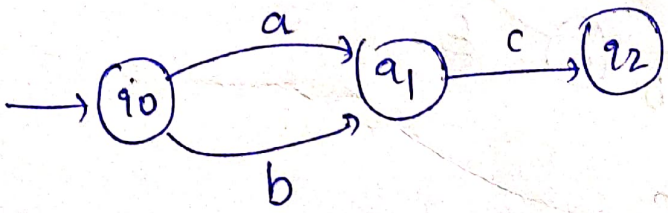
for

```
for (int i=0; i<m to; i++)
    arr[i] = num1[i];
}
for (int j=0; i<
```

$a^*b\,(a+b)^*$



Qu = $(a+b)\cdot c$.

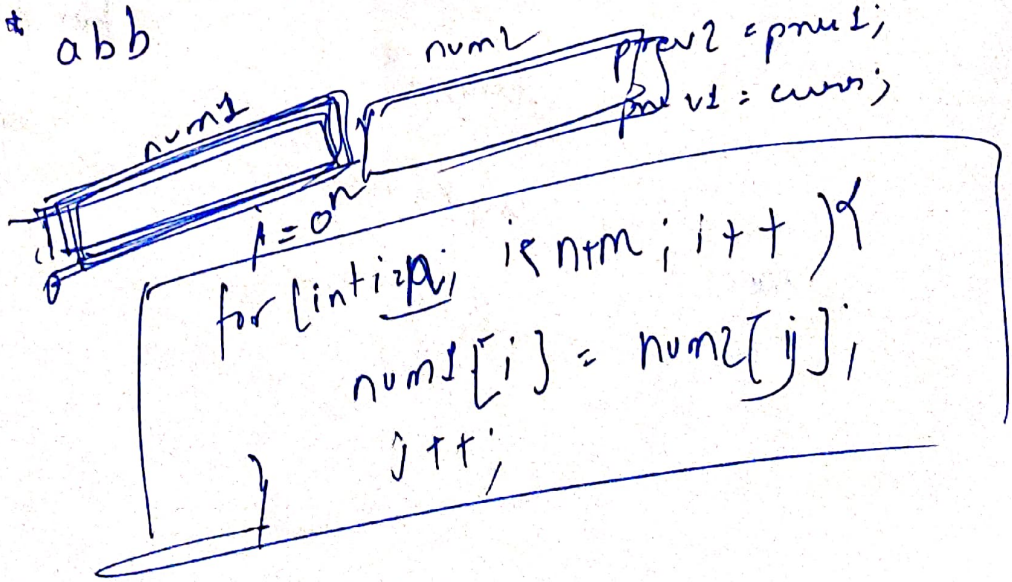


num1[10];

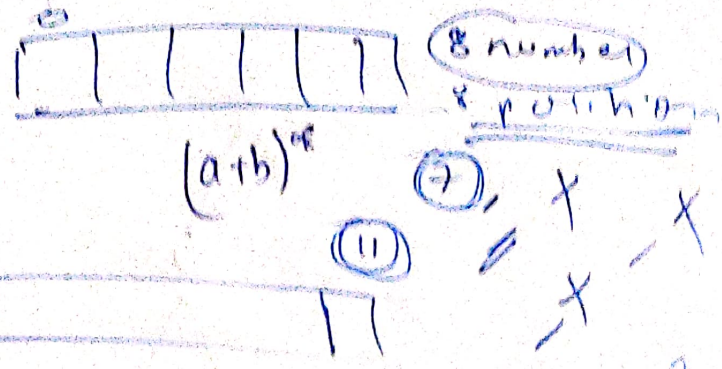
0 1 1
1 2
prev1 prev1

curr = prev2 + prev1

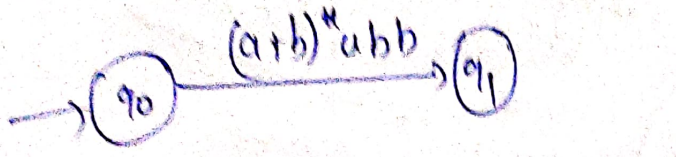
Qu = $(a+b)^*\,abb$

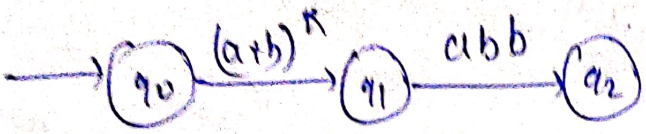


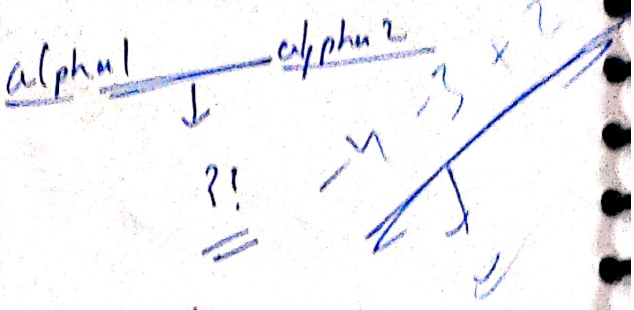
Qu $(a+b)^* abb$



$(a+b)^* abb$







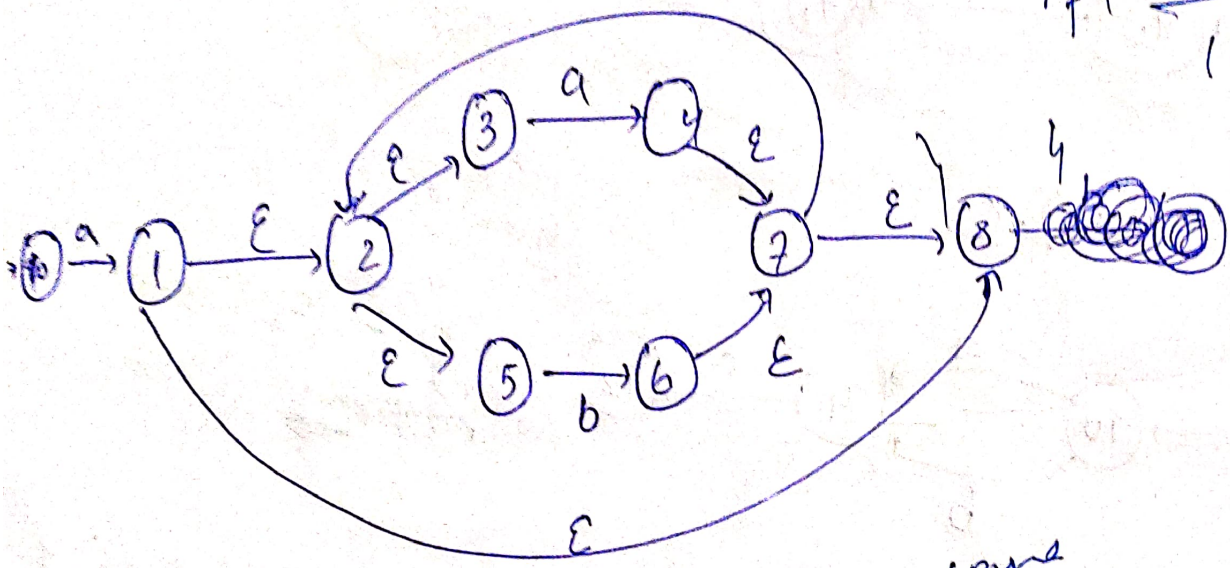
count

for (int = 0; i < arr.size(); i++) {

if (arr[i] == 'y')

count += arr[i]

Qu $a(a+b)^*$



length: 9

alphu = 2

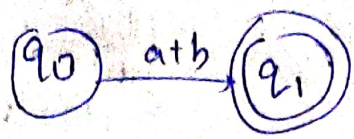
alphu = 6

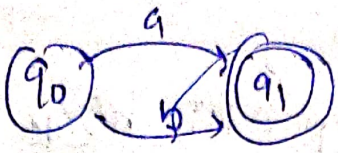
count

for (— — —

counter;

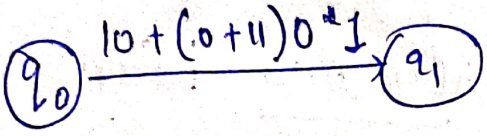
a+b

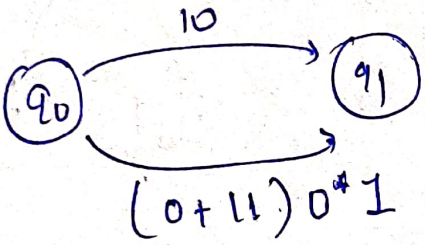


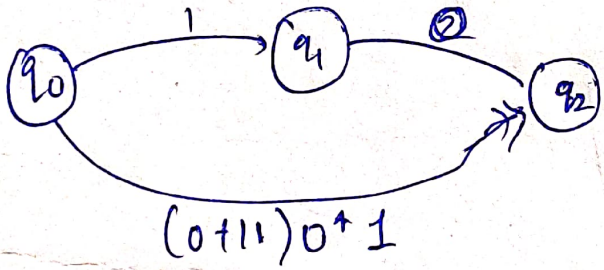


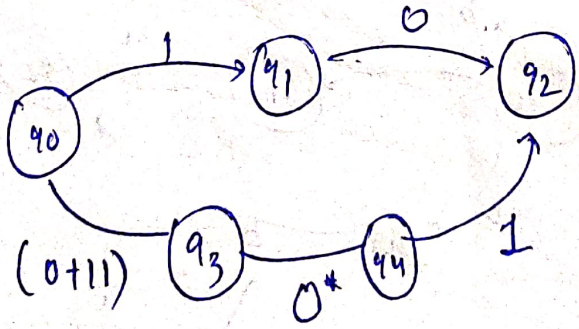
a.b

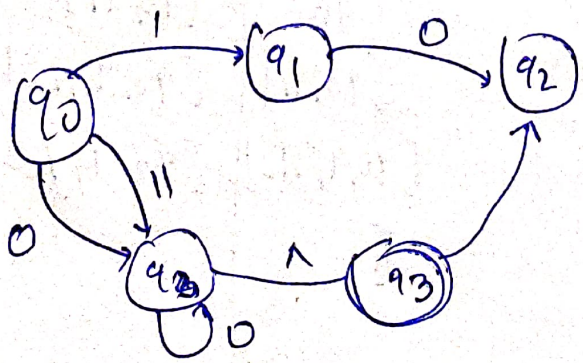
Eg $10 + (0+11)0^*1$

(1) 

(2) 

(3) 

(4) 

(5) 

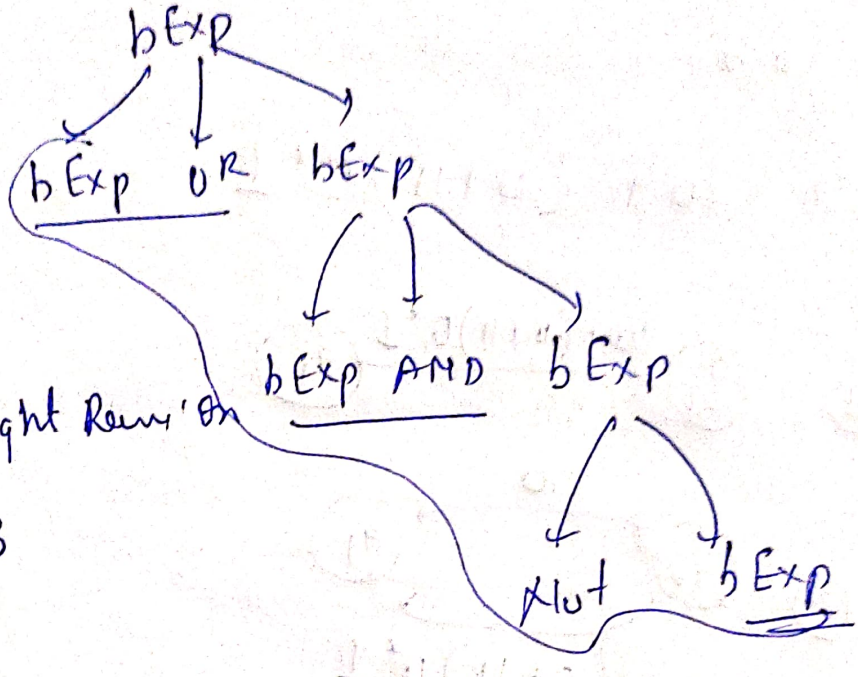
Qu =

OR
AND
Not

L → R

bExp OR bExp

bExp OR bExp AND (Not bExp)



Qu Left Recursion to Right Recursion

① $A \rightarrow A\alpha \mid B$

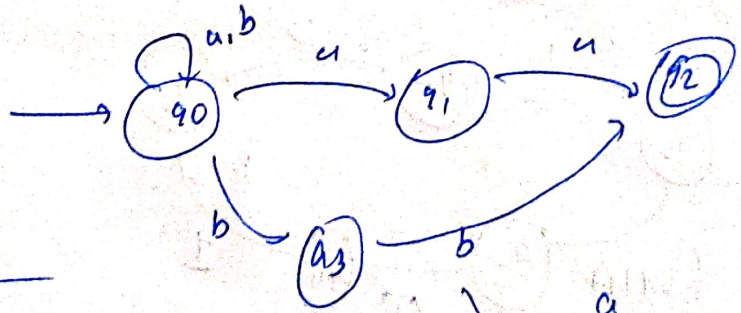
Qu $P \rightarrow P + Q \mid Q$

Qu

$P \rightarrow Q P'$

$P' \rightarrow + Q P' \mid \epsilon$

Qu.



	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	-
(q_2)	-	-
q_3	-	$\{q_2\}$

	a	b
q_0	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0]$
$[q_0, q_1, q_2]$	$[q_0, q_1]$	$[q_0]$

Ex: $E \to E + T \mid T$ ⟹ Highest to Lowest

$T = T * F \mid F$

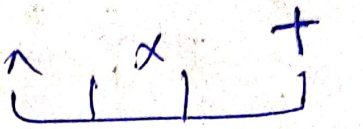
$F = G_1 \wedge F \mid F$

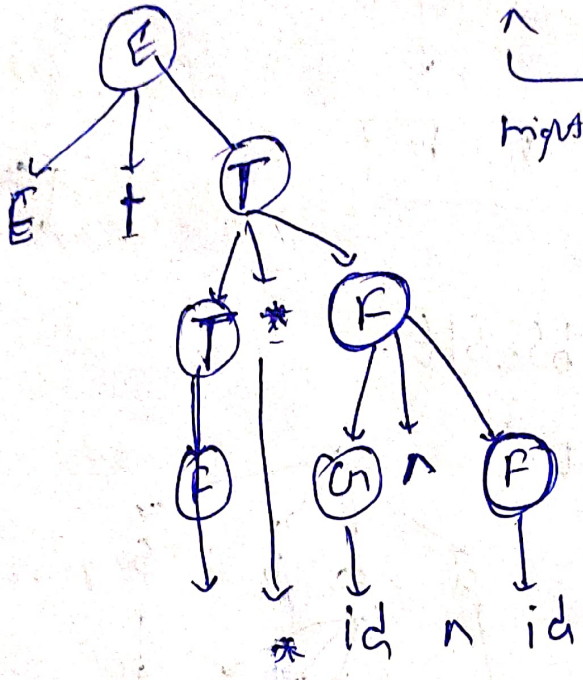
$G_1 \Rightarrow id$ ⟹ highest

left association

left

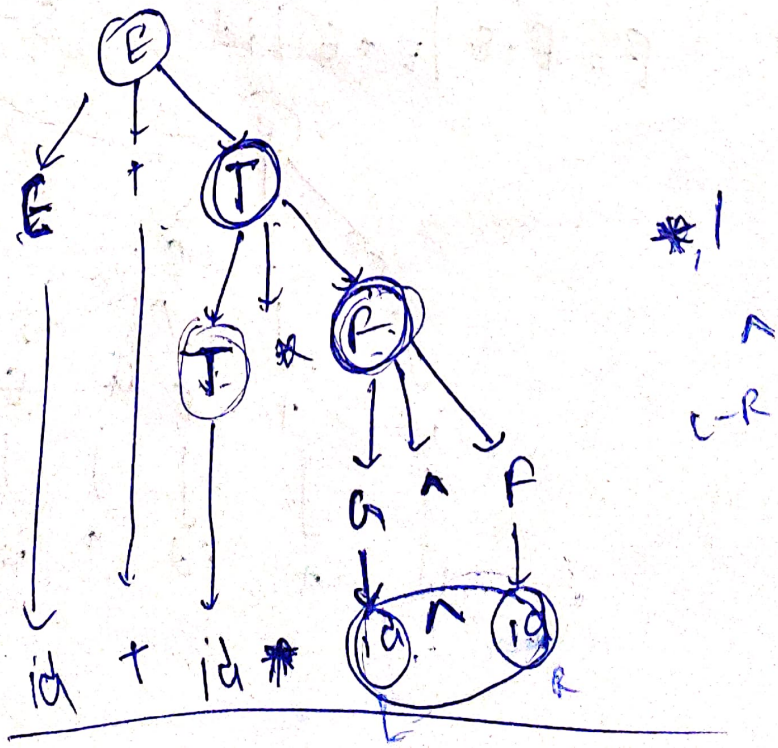
Right associative





bExp → bExp OR bExp

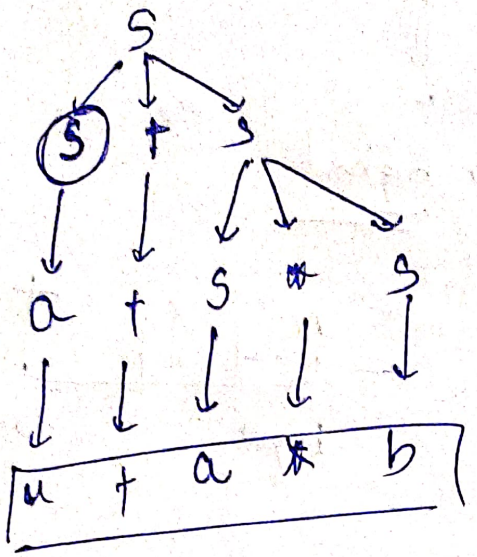
bExp AND (bExp OR bExp)



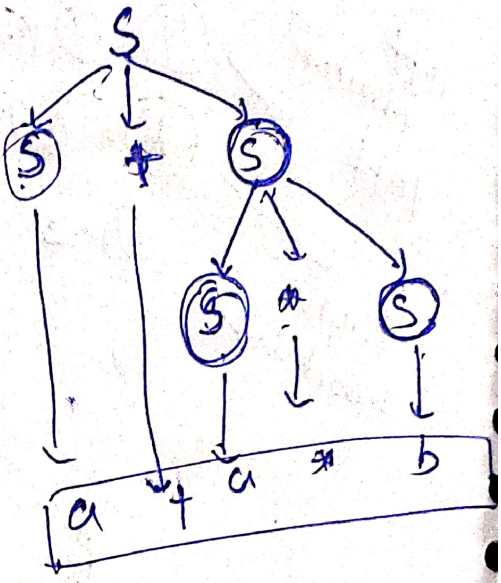
$S \rightarrow S+S \mid S*S \mid a \mid b.$

$a+a*b$

LDT

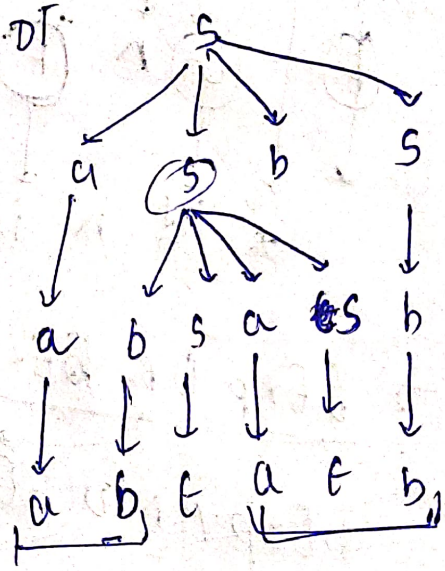


RDT

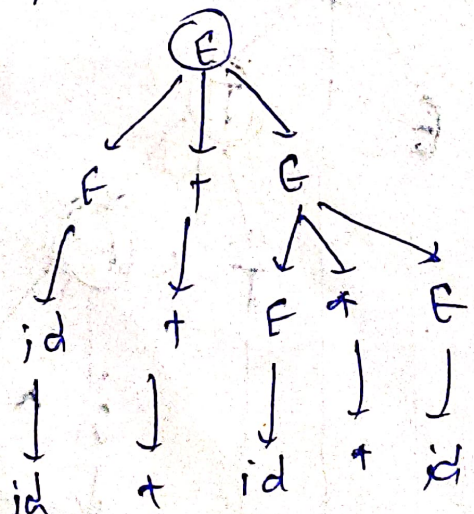


Qu: $S \rightarrow aSbS \mid bSaS \mid \epsilon$

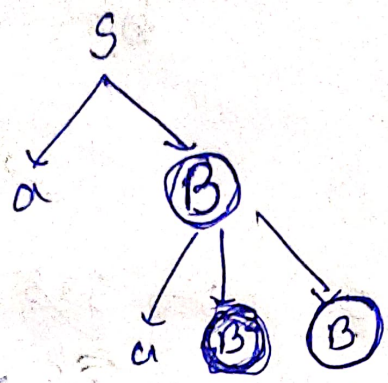
$w = abab.$ LDT

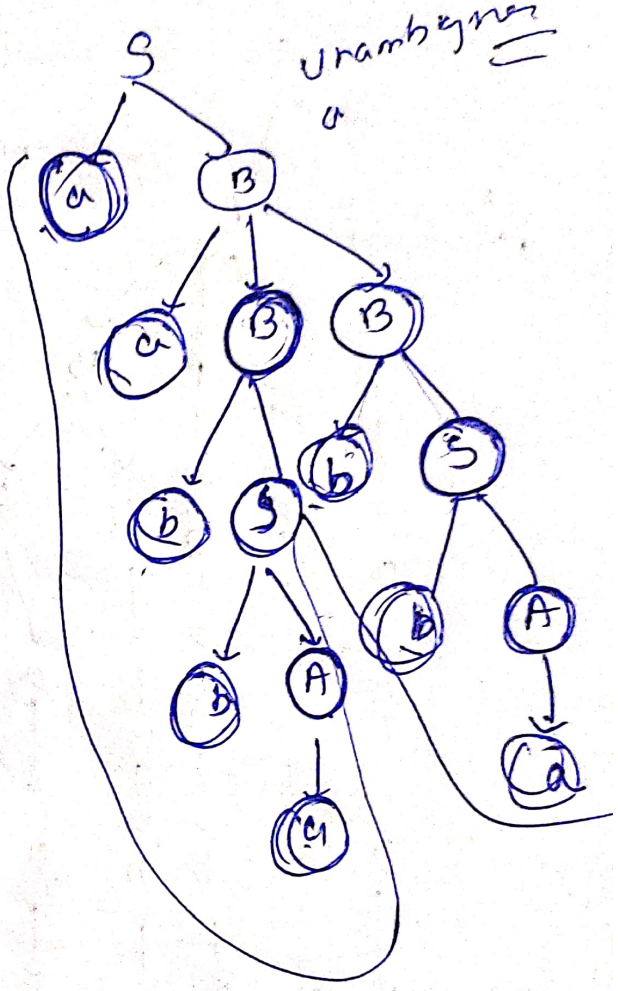


Qu: $E = E+E \mid E*E \mid id.$ $\quad S \rightarrow id+id+id.$



Right most

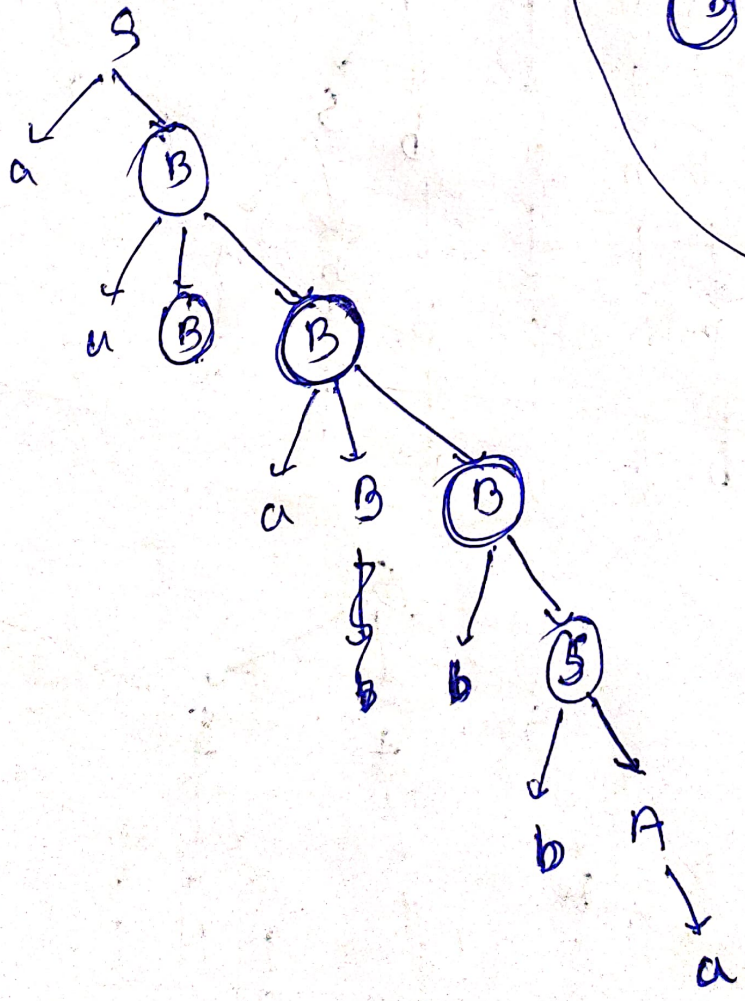




Unambiguous

a

aabbabba



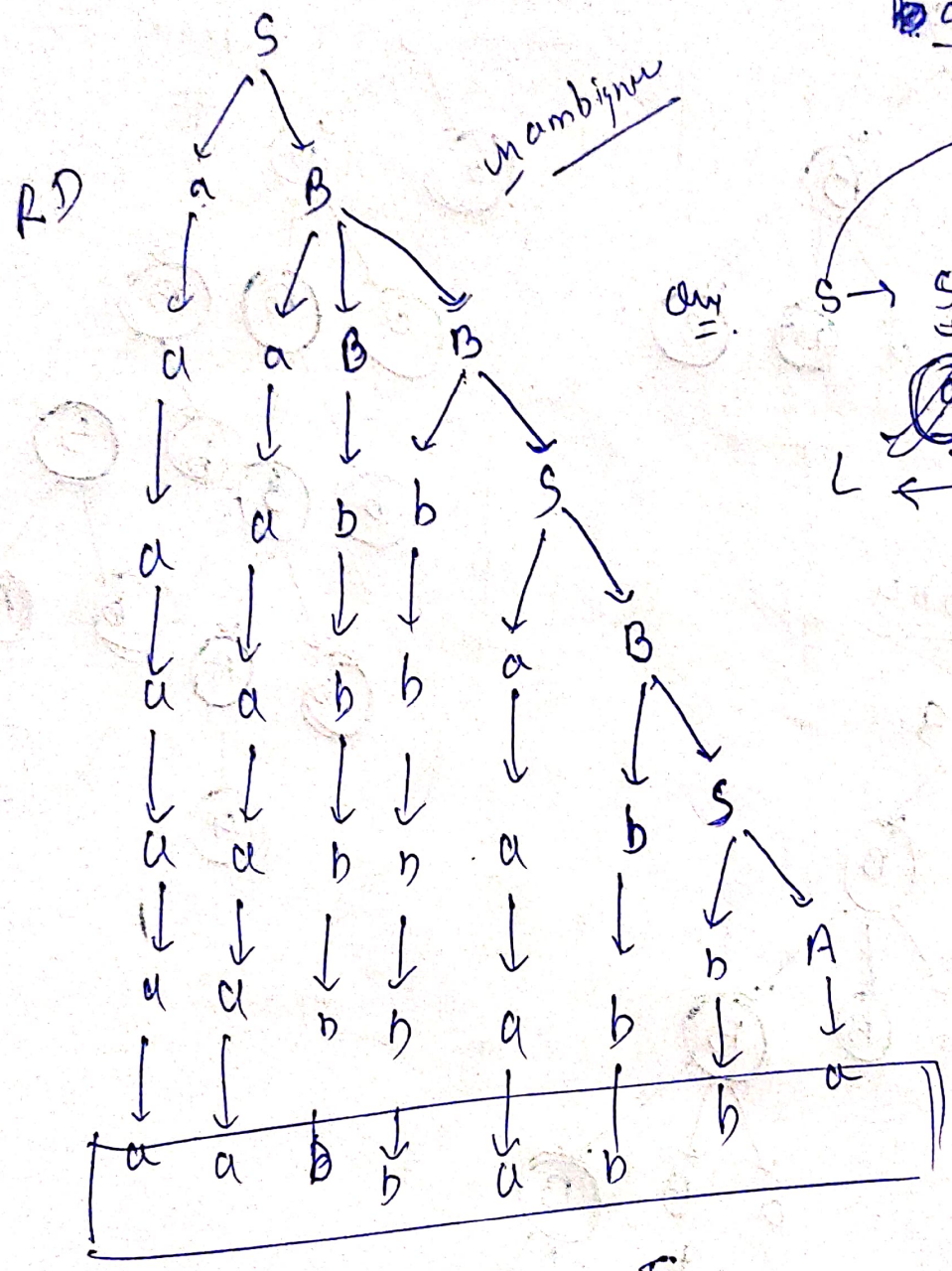
Qu2

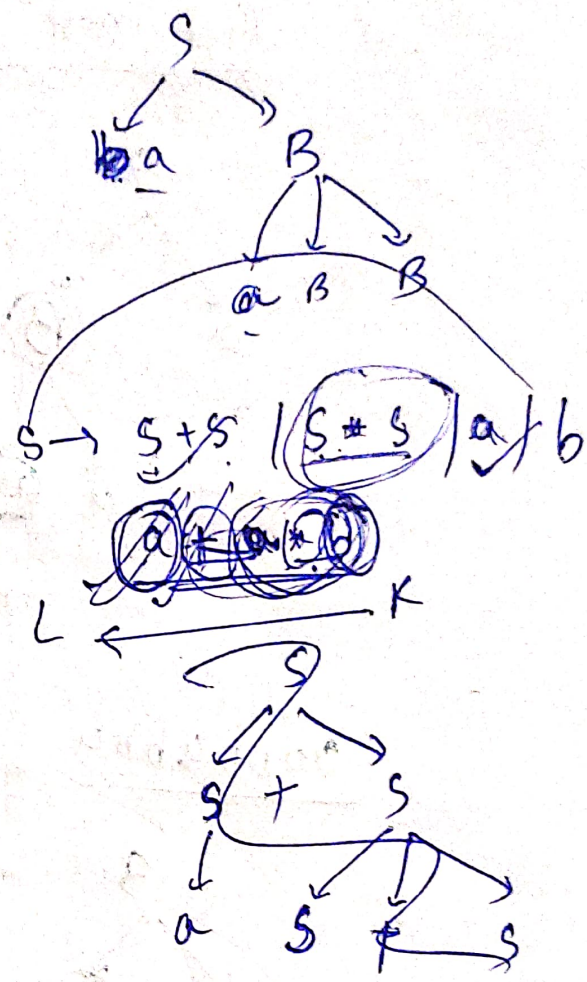
$S \rightarrow aB \mid bA$

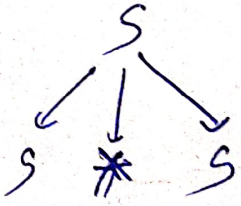
$A \rightarrow a \mid aS \mid bAA$

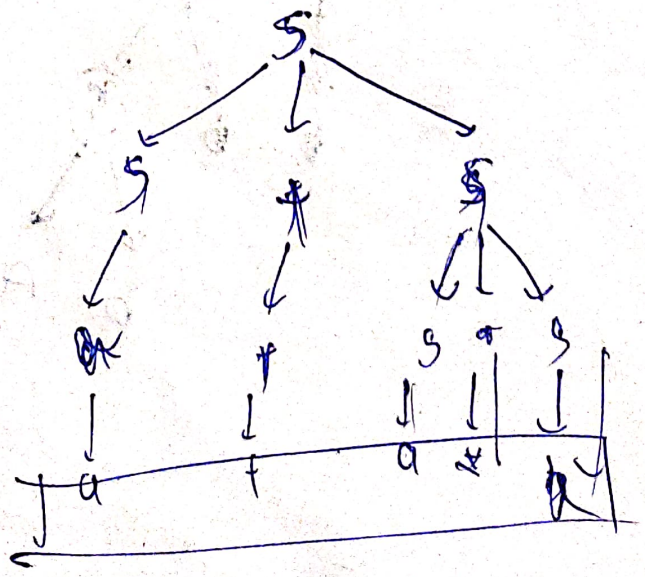
$B \rightarrow b \mid bS \mid aBB$.

aabbabba









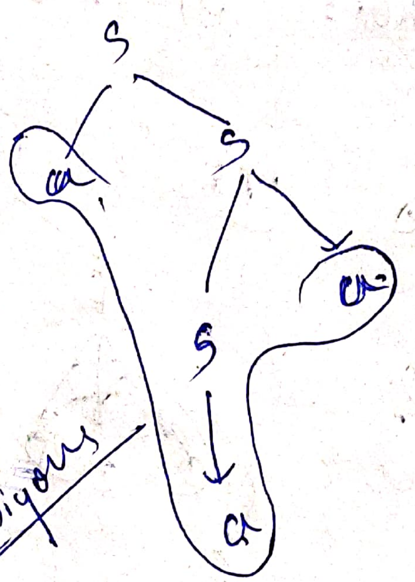
1. $a(a+b)^*$ $\underline{a}$
2. $(a+b)^* a$ $a\,\square$ $\square\, a\, \square$ $a\,\square$ a
3. $a(a+b)^* a$ $\neq \underline{a}$
4. ~~$a(a+b)^* + (a+b)^* a$~~

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

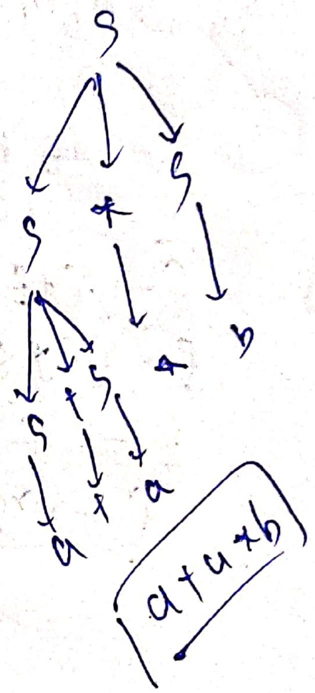
Qu. $S \to aS \mid Sa \mid a$ $a+a+b$ R L

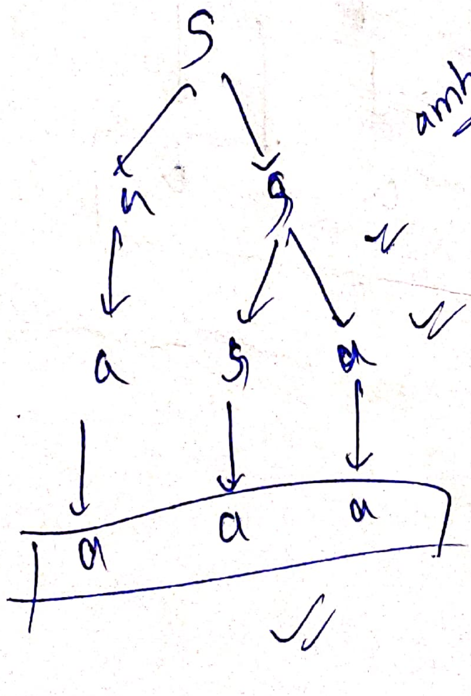
$w = aaa$

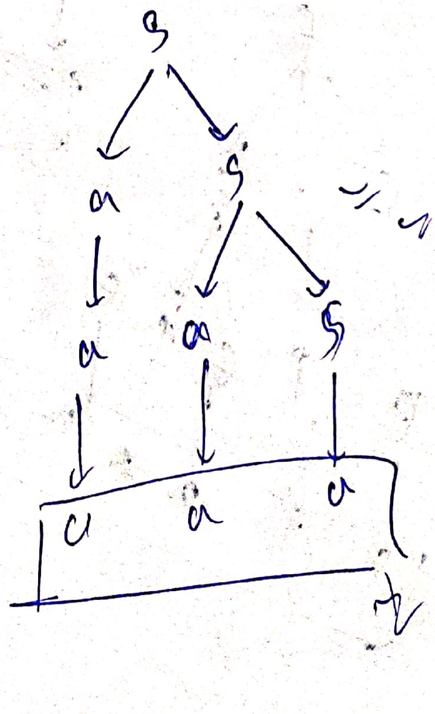
$S \to aS$
$S \to Sa$
$S \to a$



ambiguous







Q. $0^{2n+3} b^n$, $n \geq 0$

R.G. ⇒ $S \rightarrow aaSb \mid aaa$

$a^{2n+3} b^n$, $n \geq 0$ 1

R.G. $S \rightarrow aaSb \mid a^5 b$

Q. $a^m b^n$, $m \geq n$, $n \geq 0$.

R.G. : $S_1 \rightarrow aS_1b \mid \lambda$

Q. $E \rightarrow E+E$

$E \rightarrow E*E$

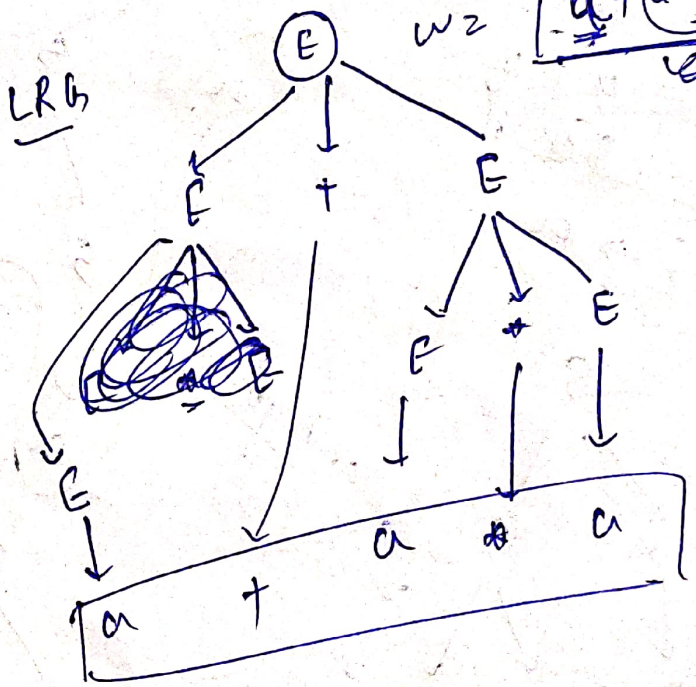
$E \rightarrow E$

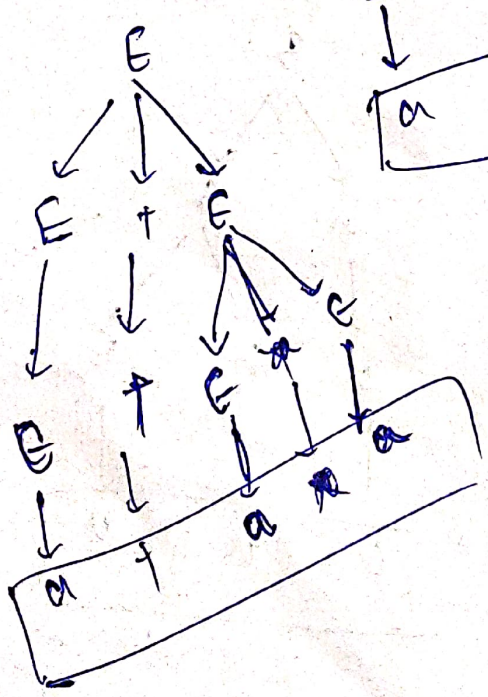
$E \rightarrow a$

LRD

R'

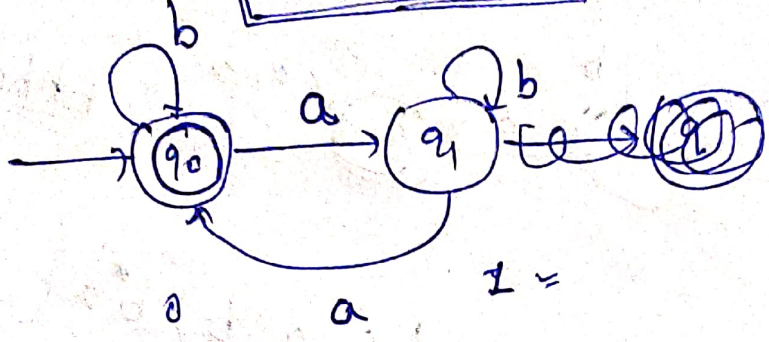
$w = a+a*a$



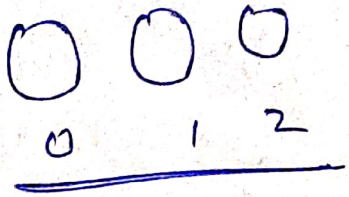


Even No. of a's $L = \{ \epsilon, a^2 \ldots \}$

$a \% 2 == 0$

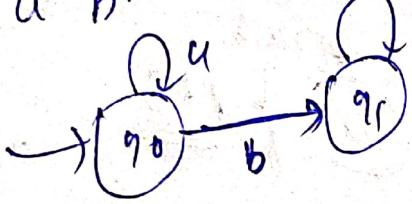


$a \% 3$



0 a 1 =

$a^* b^*$



$a^* b b^*$

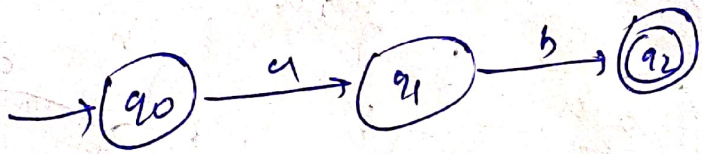
Qy $L = \{ a^n b^n \mid n \geq 1 \}$

ab
$a^2 b^2$

$L = \{ ab, a^2 b^2, a^3 b^3 \ldots \}$

ab ab

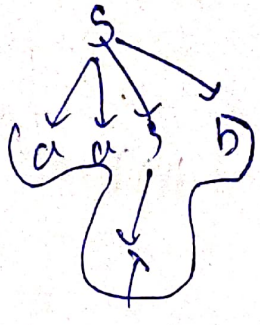
aabb



Qy. $L = a^n b^n, n \geq 0$ $\quad S \rightarrow aSb \mid \epsilon . ab$

$L = a^n b^{n+2}, n \geq 0$ $\quad S \rightarrow aSb \mid bb$

Qy $a^{2n} b^n, n \geq 0$ $\quad S \rightarrow aaSb \mid \lambda$

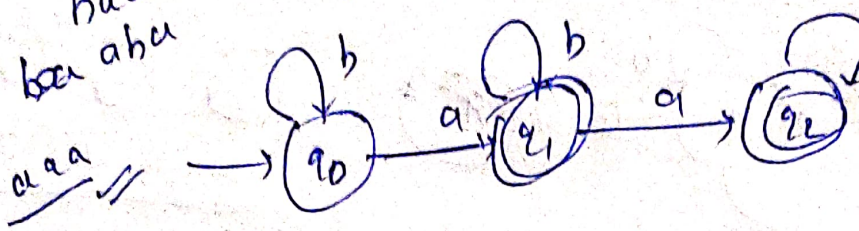


aab

At least two a's

$L = \{aa, aba, baa, aaa, \ldots\}$

(n+1) states



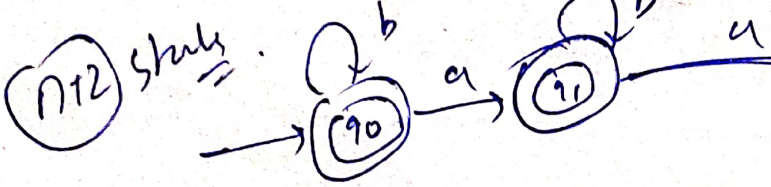
baa
aba
aaa
a
bbbaba

almost one a

$L = \{\epsilon, a, b, b^*a\}$

aa

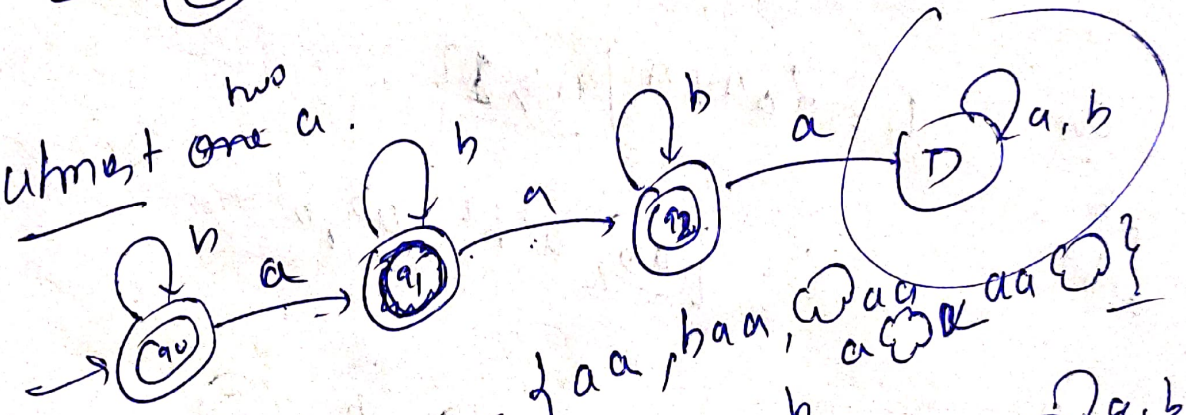
(n+2) states



PRS

ab

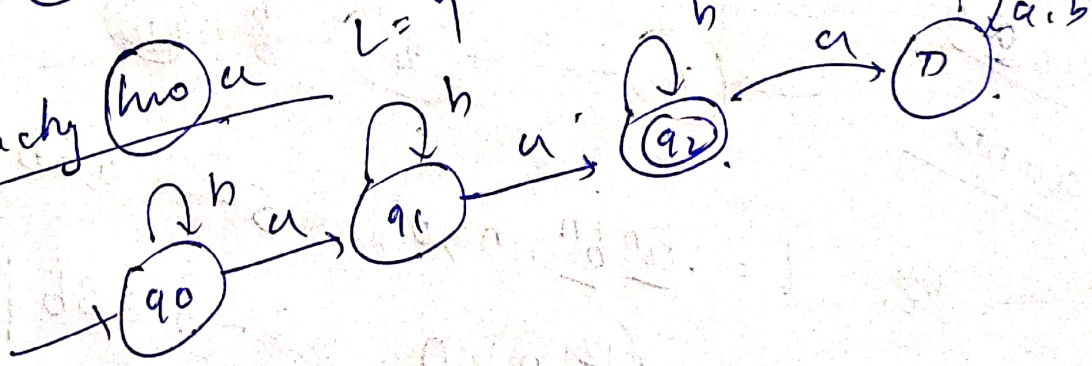
almost two a's



exactly two a

$L = \{aa, baa, aba, aa\}$

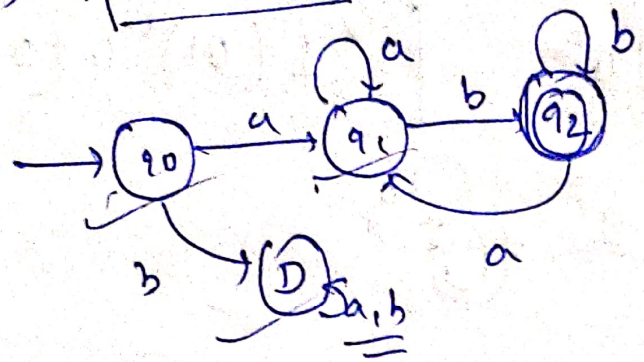
(n+2)



n letters	a's b's
Atleast (n+1)	(n+1)
Atmost } n+2	} n+2
Exactly }	

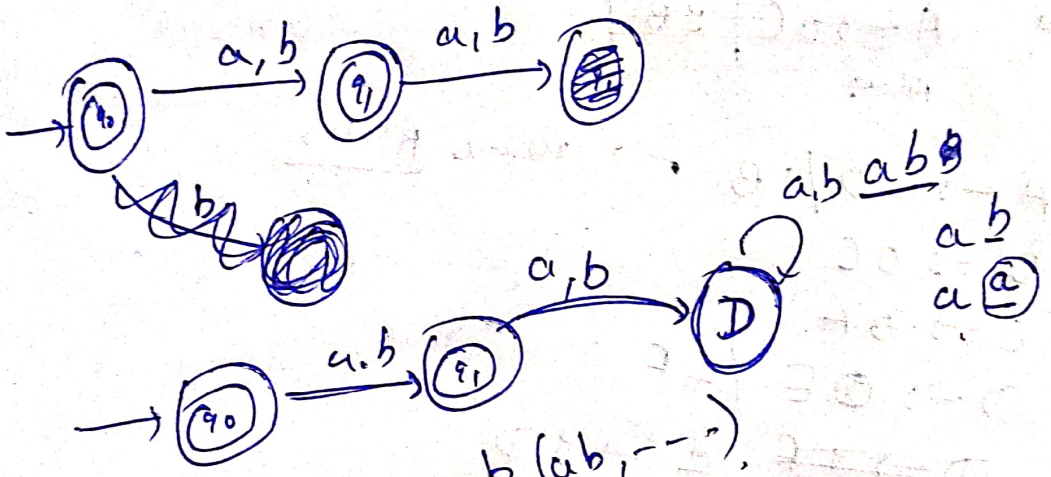
start a end b

t, ba, ab



Atmost (1)

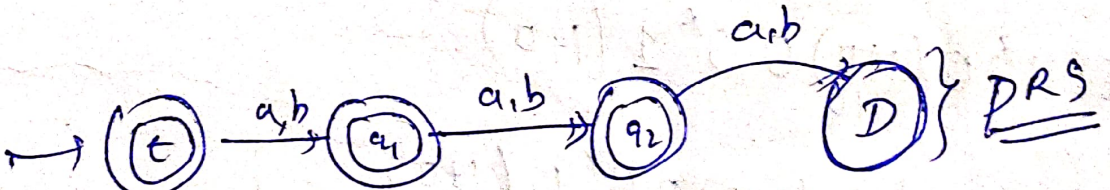
$L = \{t, a, b\}$.



t, aa, b, (ab, --)

(2+2) = 4 (n+2)

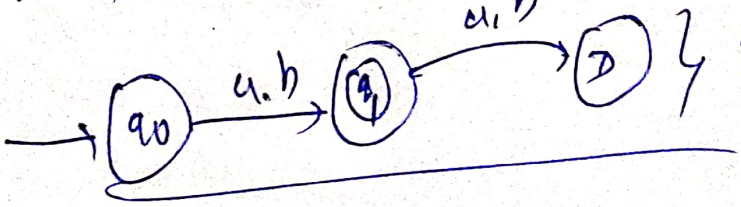
atmost (2)



(1) $L = \{a, b\}$ 1

abbb

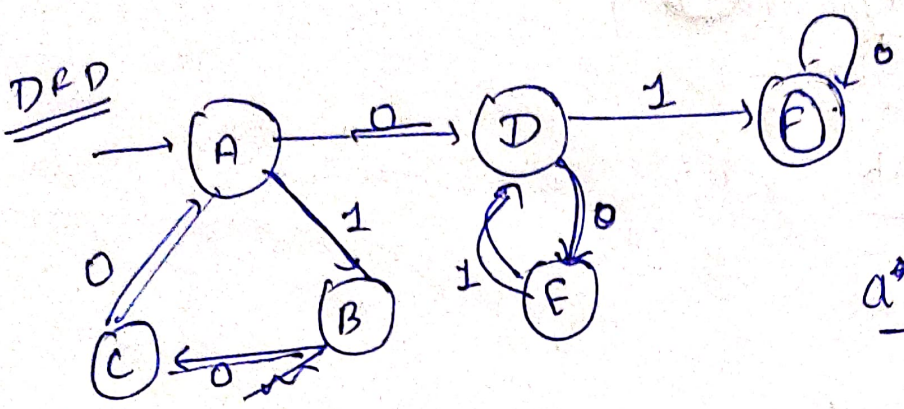
n+2



Qn = $0^*1(10)^*0(001)^*$ R.E: $0^*1(0+1)^*0($

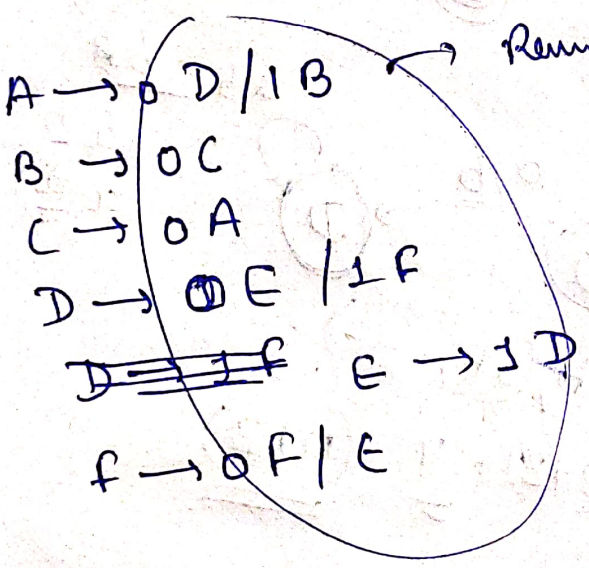
Reverse $(100)^*0(01)^*10^*$ $\boxed{0^*1(01)^*0(00)^*}$

DFD



$a^*(bc)^*$

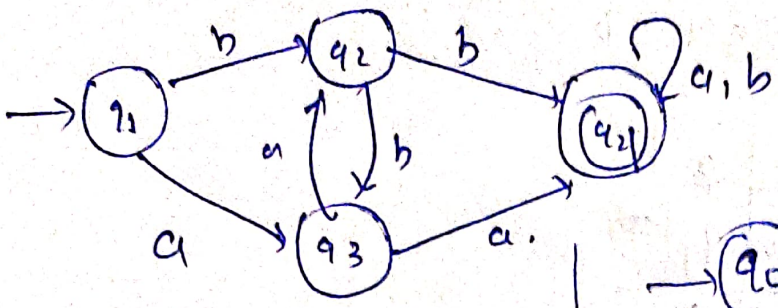
Grammar ~~A → 0C | 1B~~

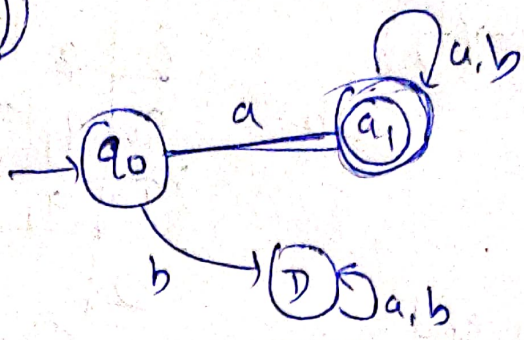


Reverse R.H.S

R.E: $1^*0(1+0)00^*1(1+0)$

R.E = $1^*0(1+0)00^*1(1+0)$





~~$q_1 \longrightarrow bq_2$~~

① $A_1 \longrightarrow bA_2$

② $A_1 \longrightarrow aA_3$

③ $A_2 \longrightarrow bA_3$

④ $A_2 \longrightarrow bA_2 \mid b$

⑤ $A_3 \longrightarrow aA_2 \mid a$

⑥ $A_3 \longrightarrow aA_2$

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_1$

S $\quad$ S_1

$q_0 \longrightarrow aq_1$

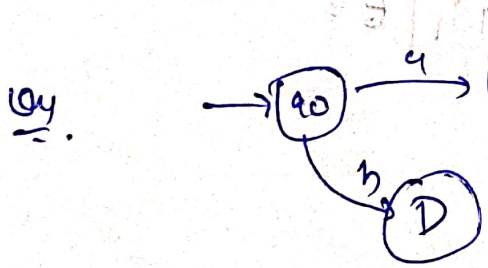
$q_1 \longrightarrow aq_1 \mid a$

$q_1 \longrightarrow bq_1 \mid b$

001 0 01
001 1 X

$S \longrightarrow aS_1$

$S_1 \longrightarrow aS_1 \mid bS_1 \mid a \mid b$

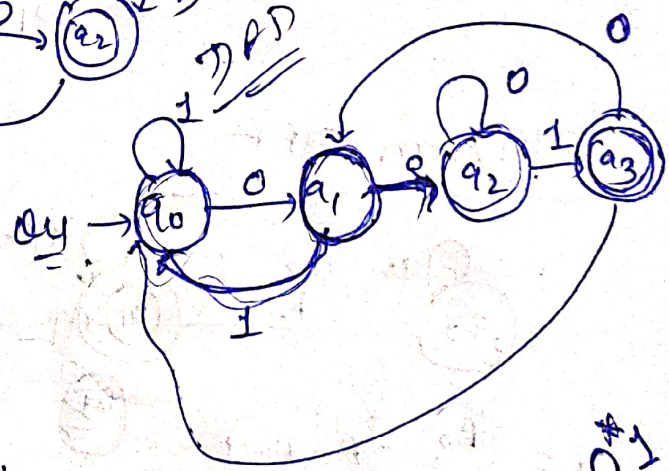


$q_0 \longrightarrow aq_1$

$q_1 \longrightarrow aq_1 \mid bq_2 \mid b$

$q_2 \longrightarrow aq_1 \mid bq_2 \mid a \mid b$

0 1
00 1
000



$q_0 \longrightarrow 1q_0 \mid 0q_1$

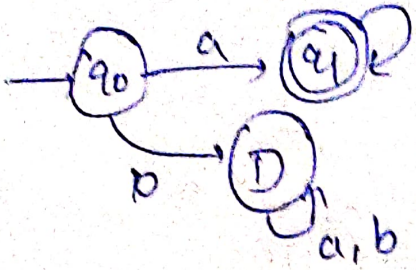
~~$q_0 \longrightarrow 0q_1$~~

$q_1 \longrightarrow 0q_2 \mid 1q_0$

$q_2 \longrightarrow 0q_2 \mid 1q_3 \mid \epsilon$

$q_3 \longrightarrow 0q_1 \mid 1q_0$

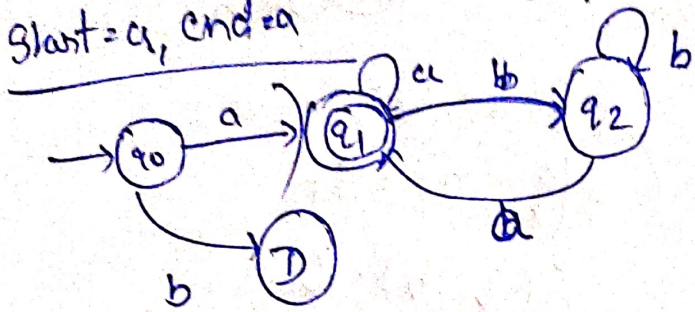
R.E $= 1^*0(1+0)00^*1(1+0)$



$L = \{a, ab \cdots\}$.

δ	a	b
q_0	q_1	D
q_1	q_1	q_1
D		

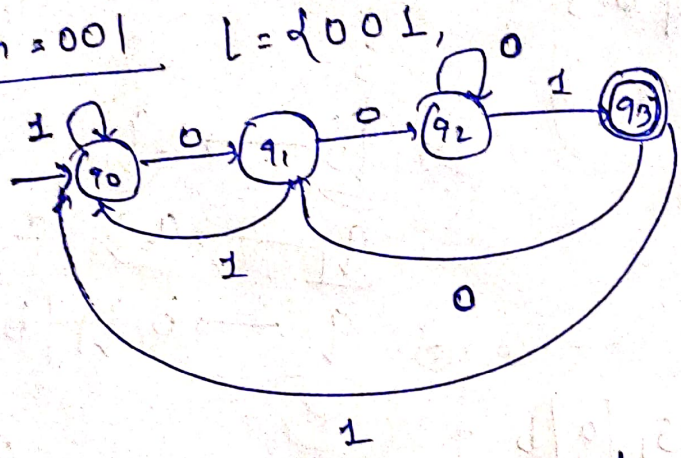
Start = a, end = a



ab a

ab b

end with = 001 $L = \{001, \ldots\}$



0 0

0 1 001

00 0 1

00 1

00 0

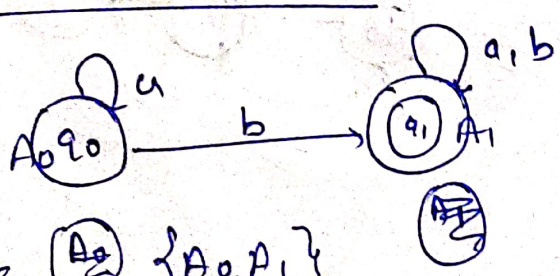
00 1 0

001 1 001

End $S = S_1 001$

$S_1 = 0S_1 \mid 1S_1 \mid \epsilon$

RG from DFA



$G = (V, T, P, S)$

$V = \{A_0, A_1\}$

$T = \{a, b\}$.

$S = A_0$

$\delta(A_0, a) = A_0$

$\delta(A_0, b) = A_1$

$\delta(A_1, a) = A_1$

$\delta(A_1, b) = A_1$

$A_0 \longrightarrow aA_0$

$A_0 \longrightarrow bA_1 / b$

$A_1 \longrightarrow aA_1 / a$

$A_1 \longrightarrow bA_1 / b$

$A_0 \longrightarrow aA_0$

$A_0 \longrightarrow bA_1 / b$

$A_1 \longrightarrow aA_1 / a / bA_1 / b$.