

DATA STRUCTURE

(8)



Data Structure

References

1. The C- Programming Language
by Kernighan & Ritchie 2nd edition
2. Data structure in c by Tenenbaum
3. Google

Syllabus

1. Array
2. Stack
3. Queue
4. Linked List
5. Tree (BST , AVL , B-Tree , B⁺-Tree)
6. C-Concepts (Pointers)

D.N.T.
- 10 -

DELTIA	Page No.	
Date:		

Arrays

#

Declaration

int a ; Creating memory for the variable a
 a which will takes 2-bytes.

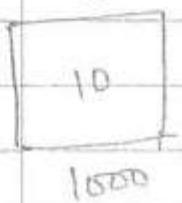


(Declaration is not for user it is for compiler)

#

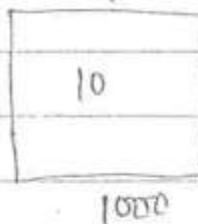
Initialization

a a=10;



#

int a = 10 ;

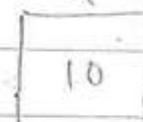


printf("%d", a)
 ↓

variable-name indicates value.

O/P : 10.

int a=10; / int a;
 a = 10;



// Normal variable
initialization

—15700

int a,b;



It is the compiler's
purpose.

not for us.

$$b = \& a$$

// Pointer Variable initialization

print("Hello, world")

Print { " % ", & b } \Rightarrow 10

* (1002)

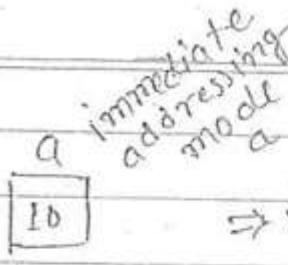
124

16

& → Reference operator

* \Rightarrow de-referencing operator.

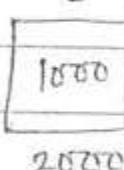
int a = 10



\Rightarrow Value. $a = 10$

int * b

$b = \& a$

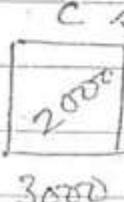


\Rightarrow Address.

$* b = 10$

int ** c

$c = \& b$

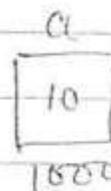


\Rightarrow (3:memory access)

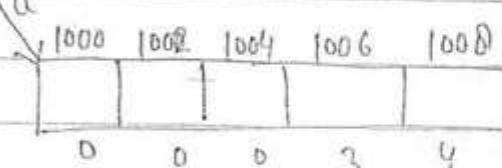
$*(*c) = 10$

Array

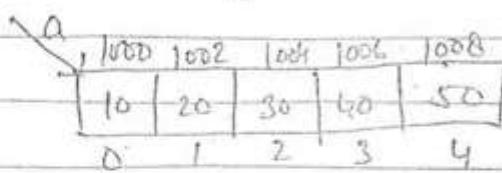
int a = 10 ;



int a[5] ;



$\text{int } a[5] = \{10, 20, 30, 40, 50\}$



here a → base address (0^{th} element's address)

* The main advantage of array is Random access
beacoz of Pointer Concepts. ***

$a[3] = 40$

$$*(a+3)$$



$$*(a+6)$$

$$*(1000+6)$$

$$*(1006)$$



40

$*(a+0) = 10$

$$a[0] = 10$$

for Compiler

#

$$*(a+4) = * (a+8)$$

$$\text{OR} \quad = *(1000+8)$$

$a[4]$

$$= *(1008)$$

for user

$$= 50$$

address)

#

$$a[i] = *(a+i) = *(i+4) = i[4]$$

in address

#

$$a[1] / *(a+1) = *(1000+2)$$

$$= *(1002)$$

$$= 20$$

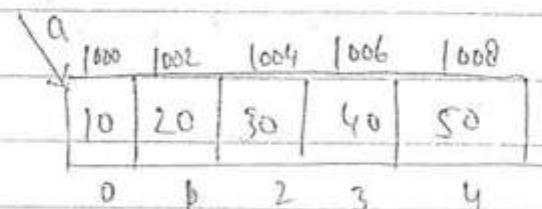
$$a[4] = *(a+4)$$

$$= *(4+4)$$

$$= 4[4].$$

One Dimensional Array.

int a[5] = {10, 20, 30, 40, 50}



$$\begin{aligned}
 \text{Loc}(a_{1,4}) &= 1000 + [(1-1)*4 + (4-1)] * 2 \\
 &= 1000 + [3] * 2 \\
 &= \underline{\underline{1006}}
 \end{aligned}$$

~~Ques.~~ Consider the following array declaration.

~~a[0:-10, 0:-10]~~

~~Base address = 1000~~

~~C = 10, RMD. them~~

~~Find the location of (a_{8,7}) .~~

$$\text{no of rows} = 10 - 0 + 1 = 11$$

$$\text{no of columns} = 10 - 0 + 1 = 11$$

~~Soln~~

$$\begin{aligned}
 \text{Loc}(a_{8,7}) &= 1000 + [(8-0)*11 + (7-0)] * 10 \\
 &= 1000 + [88 + 7] * 10 \\
 &= 1000 + 950 \\
 &= \underline{\underline{1950}}
 \end{aligned}$$

~~Consider the 007 array A [-5:-45, -5:-45]~~

~~BH = 0, 83 C = 10, RMD~~

~~Find location of (a_{0,0}) ?.~~

~~Soln~~

$$\begin{aligned}
 \text{no of rows} &= 5 + 5 + 1 = 11 \\
 \text{columns} &= 5 + 5 + 1 = 11
 \end{aligned}$$

$$\begin{aligned}
 \text{Loc}(a_{0,0}) &= 0 + [(0+5)*11 + (0+5)] * 10 \\
 &= 0 + [60] * 10 \\
 &= \underline{\underline{600}}
 \end{aligned}$$

Q2

$$\text{If } \text{Loc}(a_{3,5}) = ?.$$

$$\begin{aligned}
 \text{Loc}(a_{3,5}) &= 0 + [(3+5) \times 11 + (5+5)] \times 10 \\
 &= 0 + [22 \times 10] \times 10 \\
 &= \underline{\underline{320}}
 \end{aligned}$$

Column-Major-Order

int a[4][4]

D)] * 10

a ₁₁	a ₁₂	a ₁₃	a ₁₄
a ₂₁	a ₂₂	a ₂₃	a ₂₄
a ₃₁	a ₃₂	a ₃₃	a ₃₄
a ₄₁	a ₄₂	a ₄₃	a ₄₄

-5---45]

a

1000	02	04	06	08	10	12	14	16	18	20	22	24	26	28	30
a ₁₁	a ₁₂	a ₁₃	a ₁₄	a ₂₁	a ₂₂	a ₂₃	a ₂₄	a ₃₁	a ₃₂	a ₃₃	a ₃₄	a ₄₁	a ₄₂	a ₄₃	a ₄₄
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$$\begin{aligned}
 \text{Loc}(a_{3,4}) &= 1500 + [(4-1) \times 4 + (3-1)] \times 2 \\
 &= 1500 + 20 = \underline{\underline{1020}}
 \end{aligned}$$

~~Q9~~

$$\begin{aligned}
 \text{Loc}(q_{1,3}) &= 1000 + [(3-1) \times 4 + (4-1)] \times 2 \\
 &= 1000 + 22 \\
 &= \underline{\underline{1022}}
 \end{aligned}$$

~~Q10~~

$$\begin{aligned}
 \text{Loc}(q_{2,1}) &= 1000 + [(1-1) \times 4 + (2-1)] \times 2 \\
 &= 1000 + 2 \\
 &= \underline{\underline{1002}}
 \end{aligned}$$

* Consider the array A which is declared as follows:

$$a[\dots \dots +5, -5 \dots \dots +25]$$

$$B = 0, C = 5, \text{ CMOD}$$

Find location of $aB(q_{2,25})$?

~~Sol~~

$$\text{no of rows} = +5 + 5 + 1 = 11$$

$$\text{columns} = 25 + 5 + 1 = 31$$

$$\begin{aligned}
 \text{Loc}(q_{2,25}) &= 0 + [(25+9) \times 11 + (2+5)] \times 5 \\
 &= 0 + [339] \times 5 \\
 &= \underline{\underline{1695}}
 \end{aligned}$$

$$\begin{aligned}
 \text{Loc}(a_{-4,-1}) &= 0 + [(-1+5)*11 + (-4+5)] * 5 \\
 &= 0 + [44 + 1] * 5 \\
 &= 225
 \end{aligned}$$

Formula

Column Major Order

$[lb_1, \dots, ub_1, lb_2, \dots, ub_2]$

BA, C, m_r, n_c, CM_O then

find Loc(i, r) ?.

Where

BA = Base address

C = Size of element

m_r = no of rows.

n_c = no of columns.

CM_O = Column Major Order.

$$m_r = ub_1 - lb_1 + 1$$

$$n_c = ub_2 - lb_2 + 1$$

$$\boxed{\text{Loc}(a_{i,r}) = BA + [(r-lb_2) * m_r + (i-lb_1)] * n_c}$$

Row Major Order

A [lb₁, ..., ub₁, lb₂, ..., ub₂]

BA, C, RMO

$$m_r = ub_1 - lb_1 + 1$$

$$n_c = ub_2 - lb_2 + 1$$

$$\boxed{LOC(a_{ij}) = BA + [(i-lb_1) \times nc + (j-lb_2)] \times c}$$

LOWER TRIANGULAR MATRIX [LTM]

(Only Square Matrices)

int a[1--4, 1--4]

a ₁₁	a ₁₂ ⁰	a ₁₃ ⁰	a ₁₄ ⁰
a ₂₁	a ₂₂ ⁰	a ₂₃ ⁰	a ₂₄ ⁰
a ₃₁	a ₃₂ ⁰	a ₃₃ ⁰	a ₃₄ ⁰
a ₄₁	a ₄₂ ⁰	a ₄₃ ⁰	a ₄₄ ⁰

m₁₁ = 3 < 0

A is said to be LTM
iff

$$A[i,j] = 0 \text{ if } i < j$$

$$A[i,j] = \text{non zero if } i \geq j.$$

$[b_2]$] $\times C$

...

LTM

ed]

a_{11}	0	0	0
a_{21}	a_{22}	0	0
a_{31}	a_{32}	a_{33}	0
a_{41}	a_{42}	a_{43}	a_{44}

Row Major Order

1000									
C1		C2		C3		C4		C5	
a_{11}	a_{21}	a_{12}	a_{22}	a_{31}	a_{23}	a_{32}	a_{41}	a_{33}	a_{42}
0	1	2	3	4	5	6	7	8	9

$$\begin{aligned}
 \text{LOC}(a_{4,3}) &= 1000 + [(4-1) + (3-1)] * 2 \\
 &\quad \text{(natural no.)} \\
 &= 1000 + [(4-1)(4-1+1) + 2] * 2 \\
 &= 1000 + 16 \\
 &= \underline{\underline{1016}}
 \end{aligned}$$

Column - Major Order

a_{11}	0	0	0
a_{21}	a_{22}	0	0
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

a_{11}	a_{21}	a_{31}	a_{41}	a_{12}	a_{22}	a_{32}	a_{42}	a_{13}	a_{23}	a_{33}	a_{43}	a_{14}	a_{24}	a_{34}	a_{44}
0	1	2	3	4	5	6	7	8	9						

$$\begin{aligned}
 LOC(a_{4,3}) &= 1000 + \left[\frac{4 \times 5}{2} - \frac{2 \times 3}{2} + (4-3) \right] \times 2 \\
 &= 1000 + [6] \times 2 \\
 &= 1016
 \end{aligned}$$

$$\begin{aligned}
 LOC(a_{44}) &= 1000 + \left[\frac{4 \times 5}{2} - \frac{(4-4+1)(4-4+1+1)}{2} \right. \\
 &\quad \left. + (4-4) \right] \times 2
 \end{aligned}$$

$$\begin{aligned}
 &= 1000 + 18 \\
 &= 1018
 \end{aligned}$$

$$\begin{aligned}
 LOC(a_{4,1}) &= 1000 + \left[\frac{4 \times 5}{2} - \frac{(4-1+1)(4-1+1)}{2} \right] \\
 &= 1000 + [10 - 10 + 3] \times 2 \\
 &\approx 1006
 \end{aligned}$$

~~11~~ $\pi [-25 \dots +25, -25 \dots +25]$

CMO ; LTM

$$BA = 0 \quad NR = 51$$

$$c = 1 \quad NL = 51$$

Find ?

$$\begin{aligned}
 LOC(a_{0,0}) &= 0 + \left[\frac{51 \times 52}{2} - \frac{(25-0+1)(25+1)}{2} + (0-0) \right] \\
 &= 1326 - 325 \\
 &= 991
 \end{aligned}$$

UPPER TRIANGULAR MATRIX

A matrix A is said to be upper triangular matrix
if:

$$A[i,j] = 0 \text{ if } i > j$$

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

Column Major Order

6	01	02	04	06	08	10	112	14	16	18
an	a11	a12	a14	a15	a13	a35	a16	a24	a34	a44
0	1	2	3	4	5	6	7	8	9	

प्रतिशेषी गणित

$$\begin{aligned}
 \text{Loc}(a_{3,4}) &= 1000 + \left[\frac{(4-1)(4-1)}{2} + (3-1) \right] \times 2 \\
 &= 1000 + 16 \\
 &= 1016
 \end{aligned}$$

$$\begin{aligned}
 \text{Loc}(a_{2,3}) &= 1000 + \left[\frac{(3-1)(3-1+1)}{2} + (2-1) \right] \times 2 \\
 &= 1000 + 8 \\
 &= \underline{\underline{1008}}
 \end{aligned}$$

formula

$$A[lb_1 - ub_1, lb_2 - ub_2]$$

BA, c, m^r, c_r, UTM, CMO, RMO

Find location of $a_{(i,j)} = ?$.

$$\text{Loc}(a_{i,j}) = BA + \left[\frac{(j-lb_2)(j-lb_2+1)}{2} + (i-lb_1) \right] \times c$$

Column Major order

$$m^r = ub_1 - lb_1 + 1$$

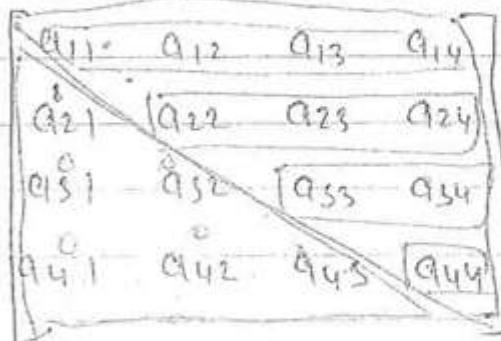
$$n^c = ub_2 - lb_2 + 1$$

$$\begin{aligned}
 \text{Loc}(a_{i,j}) &= BA + \left[\frac{m^r(m^r+1)}{2} \cdot \underbrace{(ub_1 - i + 1)(ub_1 - i + 1)}_{\downarrow} \right. \\
 &\quad \left. + (j-1)^2 \right] \times c
 \end{aligned}$$

Row-Major Order

UPPER TRIANGULAR MATRIX

Row Major Order



Row Major Order

Row						Col	Row	Col	Row	Col
0	1	2	3	4	5	6	7	8	9	10
a ₀₀	a ₀₁	a ₀₂	a ₀₃	a ₀₄	a ₀₅	a ₀₆	a ₀₇	a ₀₈	a ₀₉	a ₀₁₀
a ₁₀	a ₁₁	a ₁₂	a ₁₃	a ₁₄	a ₁₅	a ₁₆	a ₁₇	a ₁₈	a ₁₉	a ₁₁₀
a ₂₀	a ₂₁	a ₂₂	a ₂₃	a ₂₄	a ₂₅	a ₂₆	a ₂₇	a ₂₈	a ₂₉	a ₂₁₀
a ₃₀	a ₃₁	a ₃₂	a ₃₃	a ₃₄	a ₃₅	a ₃₆	a ₃₇	a ₃₈	a ₃₉	a ₃₁₀

$$LOC(a_{34}) = BA + 1000 + \left[\frac{4 \times 5}{2} - \frac{(4-3+1)(4-3+1+1)}{2} + (4-5) \right] \times 2$$

$$= 1000 + [10 - 3 + 1] \times 2$$

$$= 1000 + 16$$

$$= 1016$$

$$LOC(a_{1,4}) = 1000 + \left[\frac{4 \times 5}{2} - \frac{(4-1+1)(4-1+1+1)}{2} + (4-1) \right] \times 2$$

$$= 1000 + [10 - 10 + 3] \times 2$$

$$= 1006$$

IX

Q8 A one dimensional Array A contains $A[1 \dots 75]$. Each element is a string and take 3 bit. The Array store at location 1120. Then what is the location of $A(49)$.

- a) 1126
- b) 1164
- c) 1264
- d) 1169

Q9 Consider the following declaration of two dimensional array in C. char $a[100][100]$. Assume that array is storing memory address. What is location of $A[40][50]$.

- a) 4944
- b) 4050
- c) 5040
- d) 5050

$A[1 \dots 75]$

$$c = 3$$

$$\beta_A = 1120$$

$$LOC(A_{49}) = 1120 + (49-1) \times 3$$

$$= 1264$$

~~SDPL~~ $LOC(a_{40,50}) = 0 + [(40-0) \times 100 + (50-0)] \times 1$

$$= \underline{\underline{4050}}$$

Q3. Assume that A LTM $A[0 \dots (n-1), 0 \dots (n-1)]$ is stored in linear Array $B[0 \dots \frac{1}{2}n(n+1)-1]$ in row by row order.

for $n=100$; If $A[ac, bc]$ is stored in $B[i]$ where is $A[90, 90]$ is stored in $B[?]$

a) 4175

b) 0

c) 4165

d) 4166

Sol:

~~0 0 0 0 0 0 0 0 0 0~~
 $A[0 \dots (n-1), 0 \dots (n-1)]$

$B[0 \dots \frac{1}{2}n(n+1)-1]$

$\vdots [0 \dots \frac{1}{2}n(n+1)-1]$

$B[0 \dots 99]$

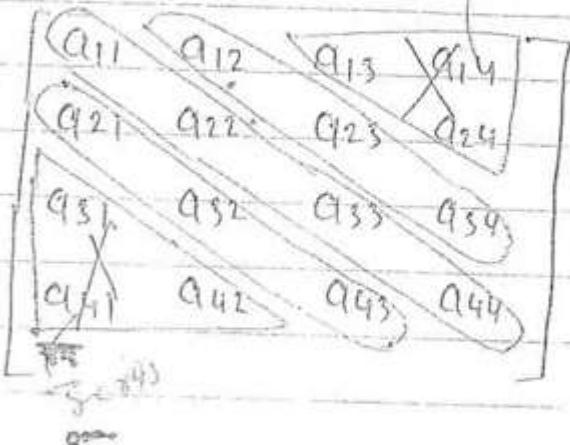
$C = 1, BA = 0 - QM0, CTM$

$$LOC(a_{90}, b_{90}) = c + \left[\frac{(q_0-0)(q_0-0H) + (00-0)}{2} \right] K_f$$

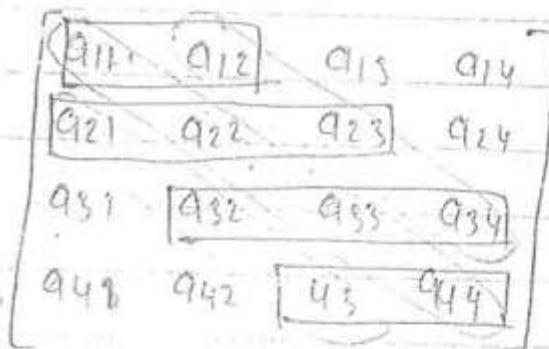
$$= 4175$$

Tridiagonal - Matrix

$$4 \times 4 = 16 \quad 3 \times 3 = 9 \\ 4+3+3=10 \quad 3n-2$$



Row-Major - Order



0	100	02	04	06	08	10	12	14	16	18
a11	a12	a21	a22	a31	a32	a33	a34	a41	a42	a44
0	1	2	3	4	5	6	7	8	9	

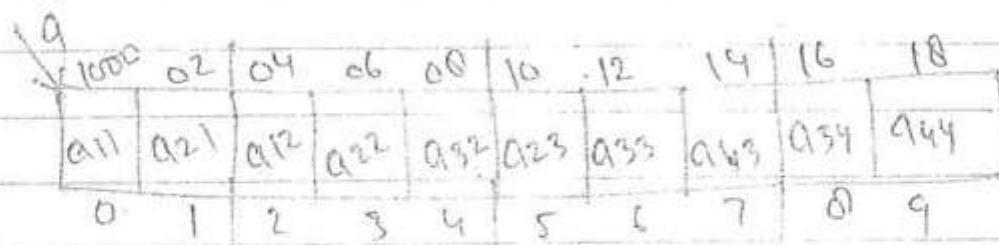
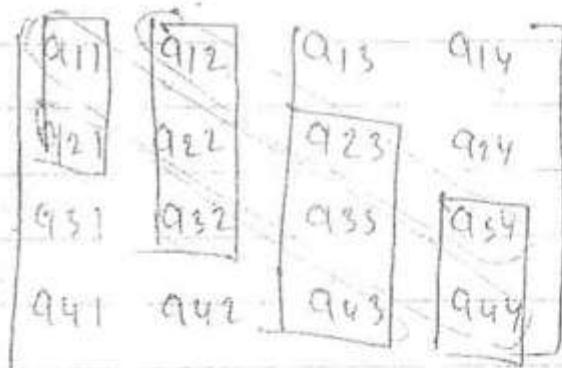
$$\text{Loc.}(a_{4,3}) = 1000 + [3(4-1)-1 + 3-(4-1)] \cdot 2 \\ = 1000 + [16] \\ = 1016$$

$$\begin{aligned}
 \text{LOC}(a_{3,2}) &= 1000 + [3(3-1)-1 + 2(3-1)] \times 2 \\
 &= 1000 + [5+0] \times 2 \\
 &= \underline{\underline{1010}}
 \end{aligned}$$

formula

$$\text{LOC}(a_{i,j}) = BA + [3(i-1) - 1 + (j - (i-1))] \times C$$

Column Major Order



$$\begin{aligned}
 \text{LOC}(a_{3,4}) &= 1000 + [3(4-1)-1 + 3 - (4-1)] \times 2 \\
 &= 1000 + 16 \\
 &= \underline{\underline{1016}}
 \end{aligned}$$

Formula

$$\text{LOC}(a_{i,j}) = BA + [3(j-1)b_2 - 1 + i - (j-1)] + C$$

Diagonal by Diagonal

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

Lower				Middle				Upper			
a_{11}	a_{12}	a_{13}	a_{14}	a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}
0	1	2	3	4	5	6	7	8	9	10	11

$$\text{LOC}(a_{i,j})$$

$$\text{if } (i == jH)$$

$$\text{LOC}(a_{i,j}) = BA + [i-2] \times C \Rightarrow \text{lower}$$

$\text{LOC}(a_{i,j})$

↳

if ($i == j$)

$$\boxed{\text{LOC}(a_{i,j}) = BA + [N-1 + (i-1)] \times C} \quad \text{-Middle}$$

$\text{LOC}(a_{i,j})$

↳

if ($i == j-1$)

$$\boxed{\text{LOC}(a_{i,j}) = BA + [N-1 + N + (i-1)] \times C} \quad \text{-Lower}$$

↳
Upper

Z - Matrix

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

First Row, Last Row and middle.

1st Row				Last Row					Middle	
a ₁₁	a ₁₂	a ₁₃	a ₁₄	a ₄₁	a ₄₂	a ₄₃	a ₄₄	a ₂₃	a ₃₂	
0	1	2	3	4	5	6	7	8	9	

Middle

$LOC(a_{i,j})$

\downarrow
if ($i = lb_1$)

$$LOC(a_{i,j}) = BA + (j - lb_2) * c \rightarrow 1st Row$$

Lower

$LOC(a_{i,j})$

\downarrow

if ($i = ub_1$)

$$LOC(a_{i,j}) = BA + [N + (j - ub_1)] * c \rightarrow Last Row$$

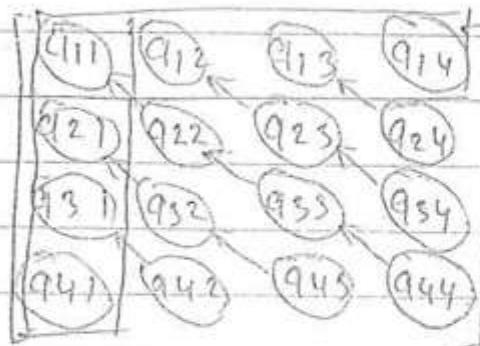
$LOC(a_{i,j})$

\downarrow

if ($i \neq lb_1 \text{ & } i \neq ub_1 \text{ & } i+j = N+1$)

$$LOC(a_{i,j}) = BA + [N + N + (i-2)] * c \rightarrow Middle$$

Toepiltz - Matrix



Matrix A is said to be Toeplitz-matrix if.

$$A[i, j] = A[i-1, j-1] \text{ for all } i > 1 \text{ and } j > 1$$



10	20	30	40
50	10	20	30
60	50	10	20
70	60	50	10

$$4 \times 4 \Rightarrow n^2$$

$$4+4-1=7 \Rightarrow 2N-1$$

\Rightarrow Store 1st row && remaining elements of 1st column.

1 st row				1 st column (Remaining)			
01	02	03	04	05	10	11	12
a ₁₁	a ₁₂	a ₁₃	a ₁₄	a ₂₁	a ₃₁	a ₄₁	
0	1	2	3	4	5	6	
a ₂₂ a ₃₂ a ₄₂	a ₂₃ a ₃₃ a ₄₃	a ₂₄ a ₃₄ a ₄₄		a ₃₂ a ₄₃	a ₄₂		

$LOC(a_{i,j})$

if

$i \leq j$

$$LOC(a_{i,j}) = BA + (j-i) * c \quad \text{1st Row}$$

$LOC(a_{i,j})$

if

$i > j$

$$LOC(a_{i,j}) = BA + N + i(j+1) * c \quad \begin{matrix} \text{1st column} \\ (\text{Rowing}) \end{matrix}$$

	x									
	a_0^0	a_0^1	a_0^2	a_0^3	a_0^4	a_0^5	a_0^6	a_0^7	a_0^8	a_0^9
	10	20	30	40	50					
	0	1	2	3	4					

$$LOC(a[3]) = BA + (3-0)*2$$

$$= 1000 + 6$$

$$= 1006$$

$$= 46$$

1st column.

10	20	30	40	50
3	4	5	6	7

Offset
(Should be calculated)

$$\begin{aligned}
 & \star(\text{Loc}(a(6))) = BA + (6-3) \times 2 \\
 & = 1000 + 6 \\
 & = \star(1006) \\
 & = 40
 \end{aligned}$$

Note :- Why Array always start from 0, why not 1.

If Array is starting from 0 no need offset value. If array is starting other than 0 we have find offset value.

int a[5] = {10, 20, 30, 40, 50};

a	1000	02	03	04	05
	10	20	30	40	50
	0	1	2	3	4

$$\begin{array}{ccc}
 \star(0+1) & = & a[1] \\
 \Downarrow & & \Downarrow \\
 \star(1000+2) & & \star(0+1) \\
 \Downarrow & & \Downarrow
 \end{array}$$

(we calculated)

$$*(1002) \quad *(1+a)$$

↓ ↓
20 1[4]

at+ X, [a = a+1]

↓ ↓
L-value R-value

↓
L-value can't be changed.

in D, why

need

g. other

int a[5] = {10, 20, 30, 40, 50}; ✓

int a[] = {10, 20, 30, 40, 50}; ✓
 ↙ optional

✓ int a[3] = {10, 20, 30, 40, 50}

10	20	30	40	50
0	1	2	3	4

a[0]

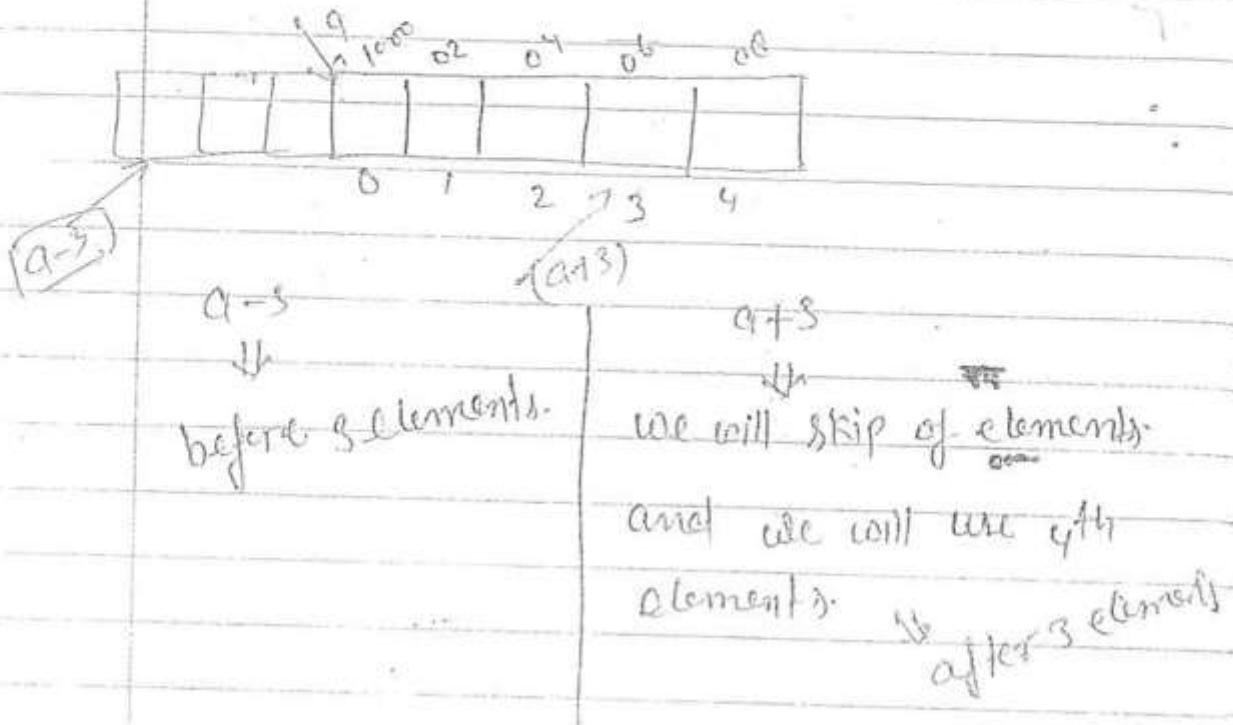
a[1]

a[2]

a[3]

a[3] = garbage, bcz total element you store.

$\text{int } a[5] = \{10, 20, 30, 40, 50\};$



$\checkmark \quad \text{int } a[3, 2] = \{10, 20, 30, 40, 50, 60\}.$

$\text{int } a[7][2] = \{10, 20, 30, 40, 50, 60\}.$

OPTIONAL
1 6 *middle*
2 3
3 2
6 1

Size
10

$\text{int } a[2][3][2] = \{ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120 \}$

$\text{int } a[] [] [] = \{ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120 \}$

optional ↓ Mandatory
 ↓ Mandatory

members.

44 In multidimensional array (m) we have to specify (m-1) value.

$\text{int } a[4][5][6];$

↓

4 arrays and each array size is (5×6) .

$$nr = Ub_2 - Lb_2 + 1$$

$a[0] \Rightarrow 5 \times 6$

$$nc = Ub_3 - Lb_3 + 1$$

$a[1] \Rightarrow 5 \times 6$

$$ng = Ub_4 - Lb_4 + 1$$

$a[2] \Rightarrow 5 \times 6$

Size of array = $nr \times nc$

$a[3] \Rightarrow 5 \times 6$

$\text{int } a[4][5][6]$

↓

$a[lb_1 - Ub_1, lb_2 - Ub_2, \dots, lb_5 - Ub_5]$

Row major order

$$\text{LOC}(a_{i,j,k}) = BA + [(1 - lb)(nr \times nc) + (j - lb_2) \times nc + (k - lb_3)] \times c$$

Column Major Order

$$\text{Loc}(a_{i,s,k}) = \beta A + [(j-1)b_1](nr \times nc) + (k-1)b_3) \times nr \\ + (s-1)b_2]$$

Q What does the following C program.

```
# include < stdio.h >
```

```
Void f( int &p, int &q )
```

```
{ p=q  
*p=2;
```

```
int i=0, j=1 →
```

```
int main()
```

```
{ -f(&i, &j)
```

```
printf("%d %d", i, j);
```

```
return 0;
```

```
}
```

a) 2, 2

b) 2, 1

c) 0, 1

d) 0, 2

Static Variable

Main()

```
{
① static int var = 5;
② printf ("%d", var--);
③ if (var)
④ main();
}
```

$x = 4$

Post Increment $x = 4$

$$y = x++ + x++$$

Printf(y, y);

↓ ↓

6 6

$x = 486$

Pre Increment $x = 5$

$$y = ++x + ++x$$

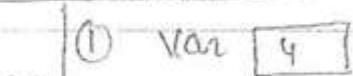
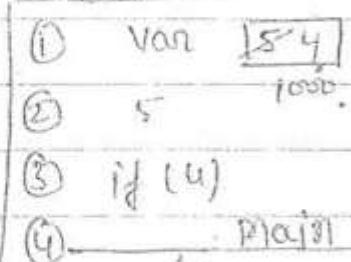
Printf(x, y);

↑ ↑

6 6

\Rightarrow If Hetic is not (without Static)

main



、^o(3) 18(4)

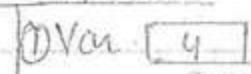
દેવ પટેલ

Stack is overfull

Statische Verbalbe

only one time
memory copies
and also initialization

→ Memory created only
time of compilation



⑤ 14(4)

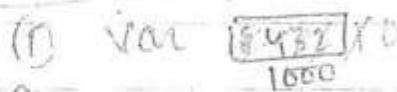
9

1

~~0/0 → 5555 - -~~

⇒ If Statie is available (with Medic)

Phantom



(5) $i_d(u)$

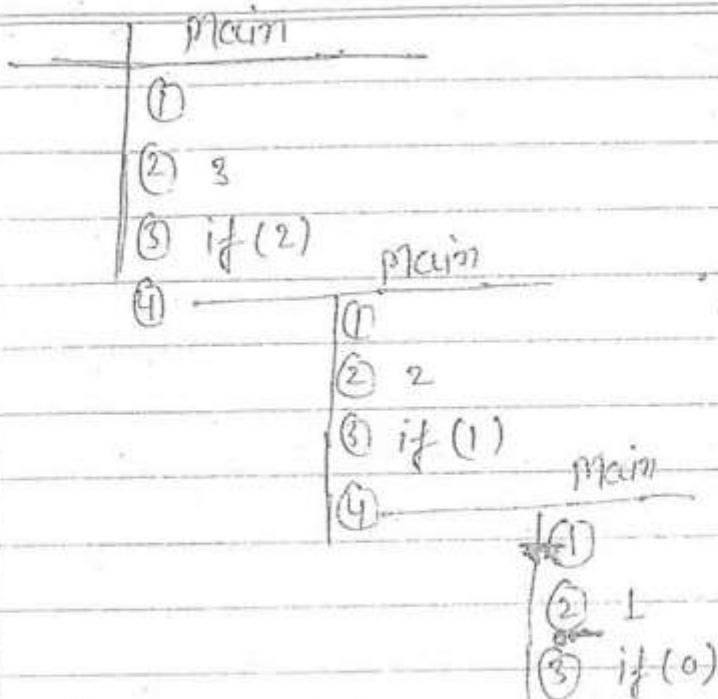
(4) plain

10

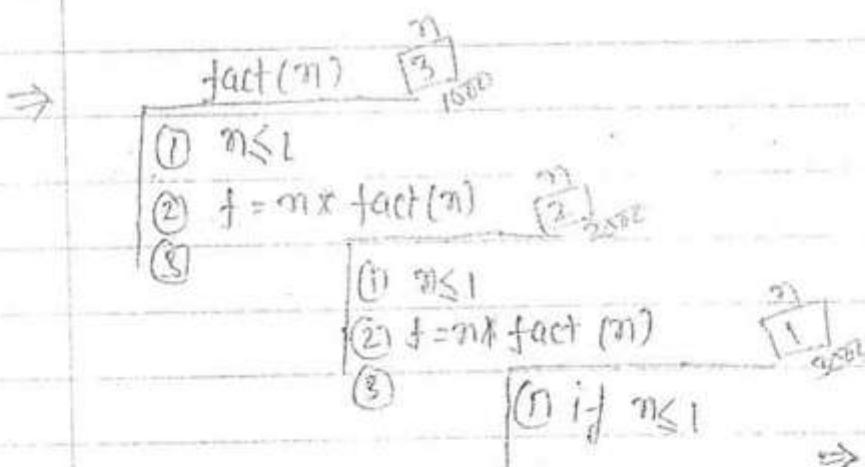
② 4

(3) if (3)

6



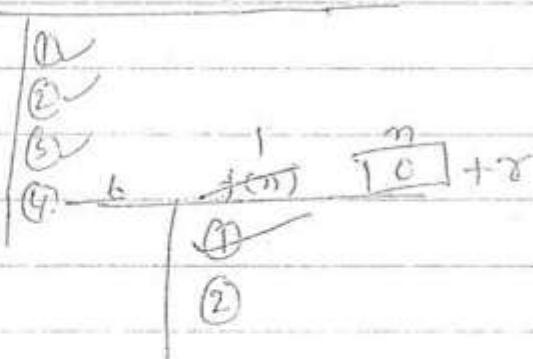
O/P \Rightarrow 5, 4, 3, 2, 1.



Note

- (1) for static variable memory will create only one becoz memory created at compile time.
- (2) for static variable initialization will be done only once.

$\boxed{6} \quad n$
 $\boxed{\text{fun}} \quad \boxed{1} + \tau$



Ques. Consider the following C program.

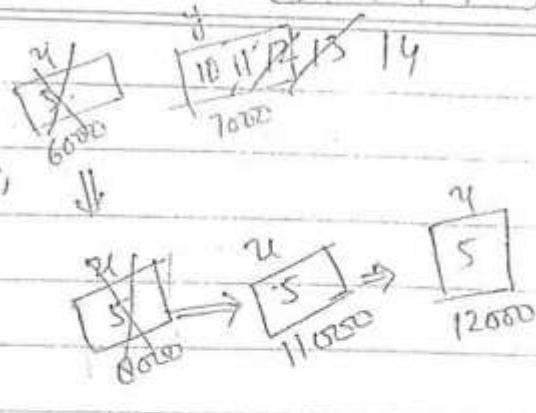
```
# include <stdio.h>
int fung( int u );
int fung( int u );
main()
{
    int u=5, y=10, count;
    for( count=1; count<=2; ++count )
    {
        y += fung(u) + fung(x);
        printf("%d", y);
    }
}
```

```
fung( int x )  $\boxed{u}$  int y;
{
    y = fung(x);
    return(y);
}
```

Recursive Program Only Stack

DETA	Page No. _____
DATE	Date: _____

fun(int u)
 {
 static int y = 10;
 y = y + 1;
 return (u + y)
 }
 o/p =



- a) 42,74 b) 80,43 c) 43,80 d) 45,43

~~sd~~

~~# Under the following C program.~~

int incr(int i)
 {

 static int count = 0;

 count = count + i;

 return (count);

}

Main()

{ int i, j;

for (i=0, i<=4 ; i++)

{ j = incr(i)

} The value of j at the end of above program

- a) 10 b) 4 c) 6 d) 7



Main()

{

int i=0, j=1, k=-1, d=2, m;

$$m = \left(\frac{d^{i+j} + j! + 2 \& x^{k+1}}{d^k + k!} \right) \mod 10,$$

Printf ("%d%d%d%d%d", i, j, k, l, m);

i=1 j=3 k=1 l=2 m=1

#

Void main()

{

int i=0;

for(i=0; i<5; i++)

switch (i)

case 0:

close 0; i+=5;

case 1:

close 1; i+=2;

case 2:

close 2; i+=5;

default: i+=4;

break;

Printf ("%d%d%d", i);

O/P = 16, 31

Note:- In switch statement after every case then should be a break.

- * After default stmt. no need of break.
- * We can give case no as int const or char.
- * In switch stmt we can keep default stmt. whenever we want. But it will execute when no one matching.

~~Main()~~

2 PM

```

Main()
{
    char a[] = "Good"; → [G O o d] [0 1 2 3 4]
    char *s = "Good"; → s
    a[2] = 'o'; → a[2] = 'o' [200]
    s[2] = 'o'; → s[2] = 'o' [000]
    printf("%s.%s", a, s);
}
    Good Good
  
```

~~Set~~

✓ strlen("Good") = 4

✓ sizeof("Good") = 5

✓ int d(s)

✓ strlen(a) =

✓ sizeof(s) = 10

✓ printf("%s", 100) → Good

✓ printf("%s", a) → Good

✓ printf("%s", 200) → Good

✓ printf("%s", q++) → cod;

~~printf("%s", 100)~~

~~printf("%s", a)~~

~~printf("%s", 200)~~

~~printf("%s", q++)~~

⇒ `printf ("%c", a);`

\downarrow

G

⇒ `printf ("%c", *(s+3));`

\downarrow

d

⇒ `printf ("%c", *(a+3));`

\downarrow

⇒ `printf ("%c", a);`

⇒ `printf ("%d", *q);`

ASCII value

of G.

⇒ `printf ("%c", *(a+2));`

\downarrow

$97[2]$

⇒ `printf ("%c", *(s+2));`

\downarrow

$97[2]$

Procedure small()

begin

Var u: real.

$$u = 0.125$$

Show();

end;

begin

$$u = 0.25$$

Show();

small()

end.

show

main

$$u = 0.125$$

static: 0.25, 0.25

Dynamic: 0.25,

static

Var u, g: integer

Procedure P(var n: integer)

begin

$$u = (n \div 4) / (n - 5)$$

end

Procedure Q()

begin

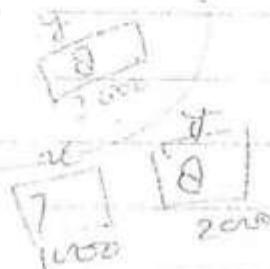
Var m, n: integers

$$m = 5, n = 4$$

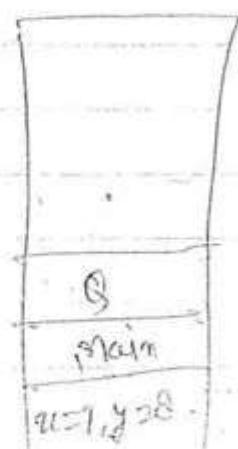
P(y)

writeln(m)

end.



static



```

begin
u:= y:= 8;
    ←
    Q.L
    while (u);
end.

```

Static : 3, 6.

Dynamic: 6, 7.

Var a,b: integer;

Procedure P()

```
begin
```

a:=5, b:=10,

```
end
```

Procedure Q()

```
begin
```

Var {a,b: integer} a,

b;

```
end.
```

```
begin
```

a:= b:=

;

Write a,b;

```
end;
```

Static: 5, 10

Dynamic: 1, 2

Q Consider the following c Program.

Var r: integer.

Procedure Two()

begin

while(r)

end.

Procedure one()

begin

Var r: integer.

r=5;

Two;

end

begin

r=2

Two;

One;

Two;

end;

Output: 2,2,1,

Dynamically allocated

Parameter Passing Techniques

- ① Call by value.
- ② Call by reference.
- ③ Call by copy-store (r) copy in-copy out
 - (r) value return
 - (r) value send
- ④ Call by Name.
- ⑤ Call by need.
- ⑥ Call by Text.

Call by Value (Call by Copy)

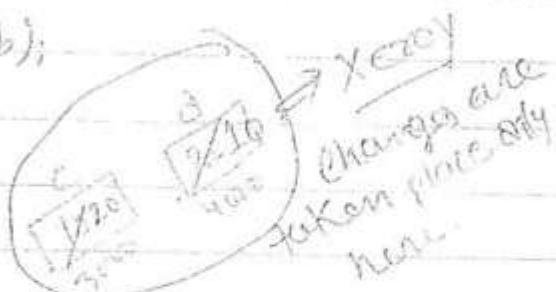
Main()

```
int a=10, b=20;
printf("%d %d", a, b);
swap(a, b);
```

```
printf("%d %d", a, b);
```

```
swap(int c, int d)
```

```
int t;
t=c;
c=d;
d=t;
```



Initial - 10, 20 (Without swap)
Final - 20, 10 (swap)

Print array(a, i, j)

{

if (i == j)

{ printf("%d", a[i]);

return;

}

else

{

Print array(a, i+1, j);

printf("%d", a[i]);

}

}

i j

i s t

i j

i j

i j

i j

i j

i j

i j

50, 40, 39, 20, 10

Print array (a, i, j)

```

    {
        if (i == j)      1 5
                        2 5
                        3 5
                        4 5
                        5 5
    {
        printf ("%d", a[i])
        return ;
    }
    else
    {
        printf ("%d", a[i]);
        printarray (a, i+1, j)
    }
}

```

Q Write a recursive C program to print the given array of n elements in reverse order.

Sol

[10 | 20 | 30 | 40 | 50]

[10 | 20 | 30 | 40 | 50]

[10 | 20 | 30] 40 , 50

[10 | 20 | 30] , 40 , 50

[10] 20 | 30 | 40 | 50

10 , 20 , 30 , 40 , 50

Application of Stack

- (i) Recursion
- (ii) Infix to postfix conversion.
- (iii) Prefix to Postfix
- (iv) Fibonacci series.
- (v) Tower of Hanoi.

Recursion

Q Write a recursive C-Program to print the given array of n-elements ($n=10$).

Sq^m

10	20	30	40	50
1	2	3	4	5

10	20	30	40	50
2	3	4	5	

10, 20, 30	40	50
3	4	5

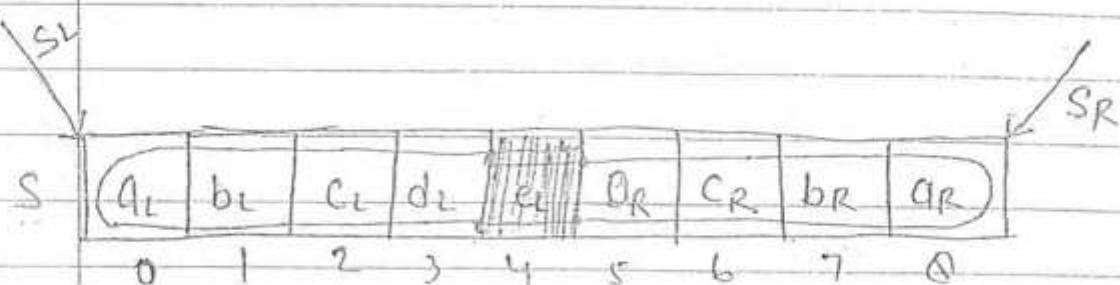
10, 20, 30, 40	50
4	5

10, 20, 30, 40, 50

10, 20, 30, 40, 50

Efficient way of implementing multiple stack.

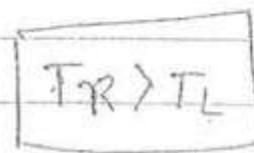
$$M=9$$



$$TL = -X, 0, 1, X, 4$$

$$TL++$$

$$S[TL] = u$$



$$TR = p[X]$$

$$TR--$$

$$S[TR] =$$

* In order to implement multiple stack efficiently ~~why~~ we are taking two stack in which are growing in opposite order what will be the full condition.

a) $TL + TR = m \times TL > RR \times$

b) $TL = TR = \frac{m}{2} \times$ (overwriting not possible)

c) $TR = TL - 1 \times$ (becoz TL is greater)

d) $TL = TR + 1 \checkmark$

$\text{POP}(S, M, N, T_i)$

```

    {
        int y;
        if ( $T_i = i \left( \frac{M}{N} \right) - 1$ )
            {
                printf ("Stack is underflow");
                exit (1);
            }
        else
            {
                y = S[Ti];
                Ti--;
                return (y);
            }
    }
  
```

Note →

Above two procedure (Push & Pop)
 Implementing stack in single Array but
 it is not the efficient way (implementing)-
 becoz 1 stack is full and remaining all
 other stack empty, it is giving error
 message saying that stack is overflow.

Push (s, M, N, Ti, u)

```

    {
        if ( Ti == ( i + 1 ) (  $\frac{M}{N}$  ) - 1 )
            {
                printf( "Stack is overflow" );
                exit( 1 );
            }
        else
            {
                Ti++;
                s[ Ti ] = u;
            }
    }

```

This code true for all the stack other than last stack. For the last stack push code assume as single stack push code.

Pop (s, M, N, To)

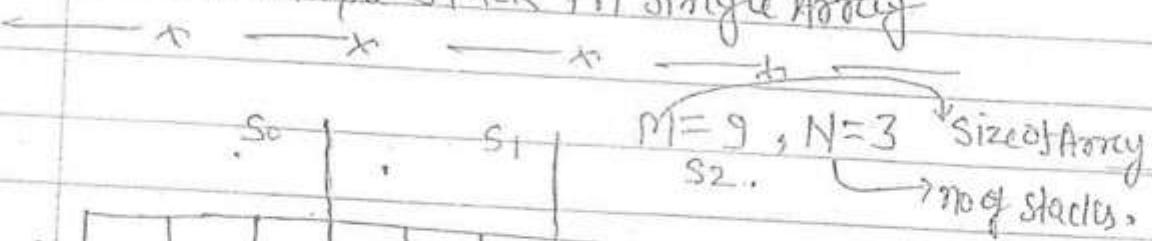
```

    {
        if ( Ti == 1 (  $\frac{m}{n}$  ) - 1 )
            else
                {
                    y = s[ To ];
                    To--;
                    return ( y );
                }
    }

```

~~Final~~ Implementing

Multiple Stack in Single Array



S	0	1	2	3	4	5	6	7	8	Size of every stack
$T_0 = -1$				$T_1 = 2$			$T_2 = 5$			$\frac{M}{N} = \frac{9}{3}$
										$\frac{3}{3}$

Initial top of the stack.

$$0^{\text{th}} = T_0 = 0\left(\frac{9}{3}\right) - 1 = -1$$

$$1^{\text{st}} = T_1 = 1\left(\frac{9}{3}\right) - 1 = 2$$

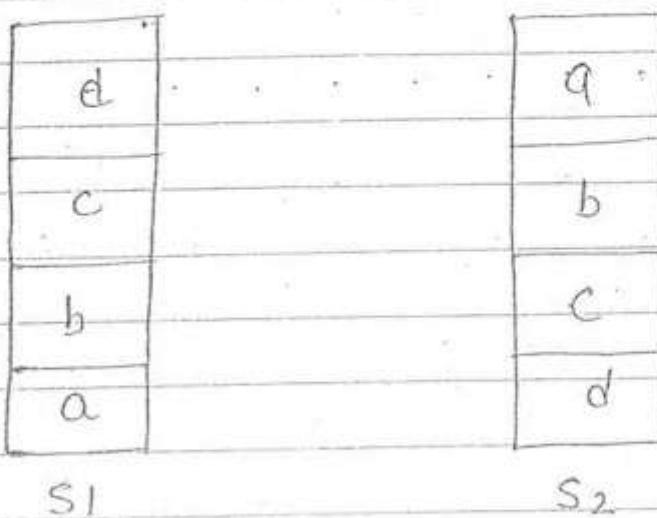
$$2^{\text{nd}} = T_2 = 2\left(\frac{9}{3}\right) - 1 = 5$$

$$i^{\text{th}} = T_i = i\left(\frac{M}{N}\right) - 1 =$$

Push (S, M, N, T_0, u)

$T_0 = (0+1)\left(\frac{M}{N}\right) - 1$
 T_0++ ,
 $S[T_0] = u$

Implementing Queue with Stack.



Queue:

Insertion

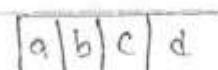
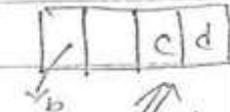
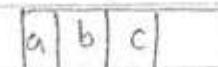
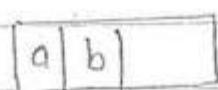
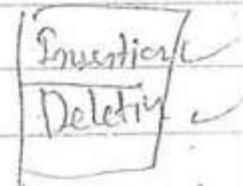
Push all the element
into stacks,

Deletion

(i) Pop all the element
from s_1 and push
into s_2

(ii) Pop elements from s_2

Queue



$\rightarrow a$



$\rightarrow a$

* ~~main~~) ~~s~~ by yes

{
 char a[2] = "Good";

char *s = "Bad";

a[2] = 'o'; // 54410 → 21410

s[2] = 'o';

printf("%s %s", a, s); // after

"Bad"
 300 301 302
 good

Note

- (i) Contents of the array can be change but contents of the string const. can't be change.

a[2] ✓

s[2] [result will be undefined]

✗

(ii)

- Base address of the array can't be change but s can be change.

a = 3002 ✗

s = 3000 ✓

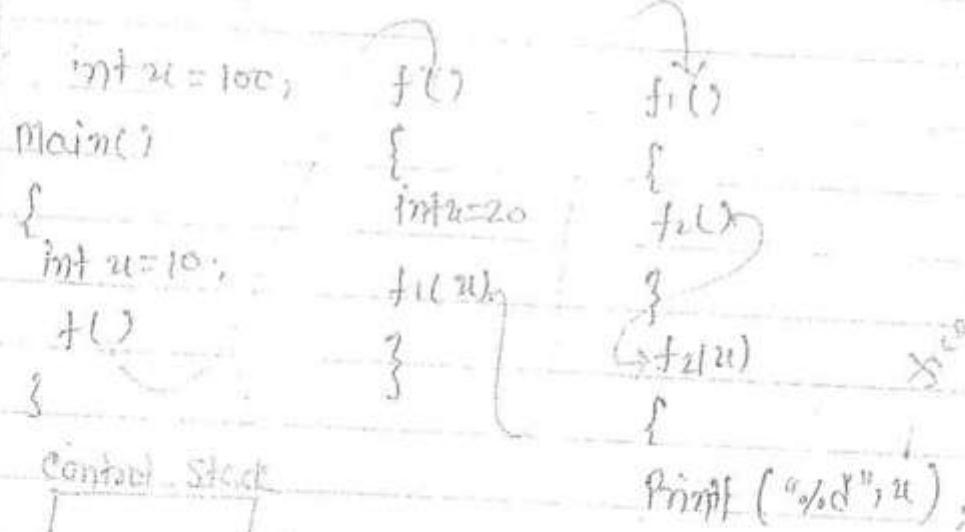
Slope of a variable (free variable)

- ① Static scoping. (Compile time) - Global variable.
- ② Dynamic scoping. (running time)

e.g.

```
main()
{
    int u=10;
    printf("%d", u)
}
```

O/P \Rightarrow 10



Control Stack

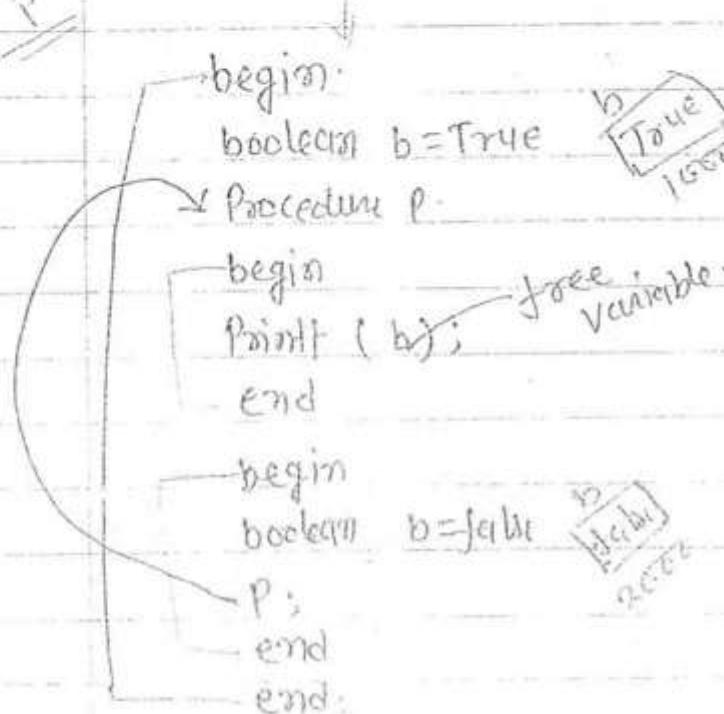


older language no one can't call the main but modern language can call the main like C.

DEJA	Page No. _____
	Date: _____

Dynamic Scoping

most recently called function. C, language follow the dynamic scoping



Static : true

Dynamic : false

Consider the following Program.

Var u:real

procedure show();

begin
print(u);

end;

u

u

$$1^{\text{st}} = 2 \times 0(u+y) + y.$$

$$2^{\text{nd}} = 2 \times 1(u+y) + y.$$

$$3^{\text{rd}} = 2 \times 2(u+y) + y.$$

$$4^{\text{th}} = 2 \times 3(u+y) + y.$$

$$5^{\text{th}} = 2 \times 4(u+y) + y$$

)

1

1

$$n = 2 \times (n-1)(u+y) + y.$$

$$\text{total} = ny + 2(u+y)[0+1+\dots+(n-1)]$$

$$= ny + 2(u+y) \frac{(n-1)(n-1+1)}{2}$$

$$= ny + 2(u+y) n(n-1)$$

$$= n[y + (u+y)(n-1)]$$

$$\text{Avg} = \frac{n[y + (u+y)(n-1)]}{n}$$

$$= y + nu + ny - u - y.$$

$$= [n(u+y) - u]$$

if $n=2$ we take only
e and b.

$$\frac{3}{+ 19}{=} 22$$

$\Rightarrow 11$

then $n=2$

$$u=5$$

$$\begin{array}{r} y=3 \\ \hline \end{array}$$

000

~~xx~~

$$u=2$$

$$y=4$$

$$n=5 \quad e=4$$

$$d=4, 2, 4, 2, 4$$

$$c=4, 2, \underline{4}, 2, 4, \underline{1}, 2, \underline{4}, \underline{3}, 4$$

$$b=\underline{4}, \underline{1}, \underline{4}, \underline{2}, \underline{4}, \underline{2}, \underline{4}, \underline{1}, \underline{4}, \underline{2},$$

$$\underline{4}, \underline{2}, \underline{4},$$

2	e	2	4
4		4	
2	d	2	4
4		4	
c		2	4
b		2	
9		4	
		2	

$$9 = \underline{4}, \underline{2}, \underline{4}, \underline{1}, \underline{4}, \underline{2}, \underline{4}, \underline{2}, \underline{4}$$

$$\underline{2}, \underline{4}, \underline{1}, \underline{4}, \underline{2}, \underline{4}, \underline{1}, \underline{4}, \underline{2}, \underline{4}$$

$$e \Rightarrow 4$$

$$d \Rightarrow 16$$

$$c \Rightarrow 28$$

$$b = 40$$

$$a = 52$$

$$\underline{140}$$

$$5 = 140$$

$$1 \Rightarrow \underline{140} = 28$$

$$5(6) - 2$$

$$= 28$$

~~XX~~ Let S be stack of size n , starting with empty stack. Suppose we insert $\frac{1}{2}n^2$ natural numbers in sequence and then we perform n -pop operations. Assume that push and pop operation takes x seconds each and y seconds. Elapse betw the end of one such stack operation and the start ~~of~~ of the next operation. For $m \geq 1$ define the stack life of m as the time elapsed from the end of $\text{push}(m)$ to the start of the pop operation that removes m from S . The avg stack life of an element of the stack is:

- (a) $n(x+y)$ (b) $3y + 2x$
 (c) ~~$n(x+y)-x$~~ (d) $y - 2x$

~~SM~~

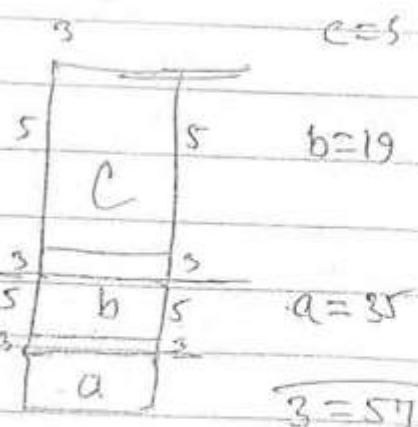
Push or Pop = $\int 2x$

Push - Push = $\int y$

Push - Pop = $\int y$

Pop - Pop = $\int y$

$$\begin{aligned} n &= 3 \\ x &= 5 \\ y &= 3 \end{aligned}$$



`int pop(s , N , Top)`

```

    {
        int y;
        if (Top == -1)
            {
                printf ("Stack is underflow");
                exit (1);
            }
        else
            {
                y = s[Top];
                Top--;
                return (y);
            }
    }
  
```

$\Rightarrow \Theta(1)$

Observation (push)

- (i) element x to be passed.
- (ii) Increment top.
- (iii) Insert the element.
- (iv) no need to return y .

Observation (pop)

- No need to Pass x
- Delete
- Decrement
- need of returning element y

DETAI
Page No. _____
Date _____

Name of Stack Size of Stack Element to be inserted.

Void Push (S, N, Top, u)

```

    {
        if (Top == N-1) → Isfull();
        {
            Print ("Stack is overflow");
            exit (1) → abnormal termination
        }
    }
    else
    {
        Top++; → O(1) constant time
        S[Top] = u; } S[Top] = u ✓
        { } S[Top+1] = u X
    }
}

```

P.C.P (Top)

```

    {
        if (Top == -1)
            empty.
        else
        {
            y = S[Top]
            Top--;
            return (y)
        }
    }
}

```

S	N=6
5	.
4	.
3	d
2	c
1	b
0	a

(i)
(ii)
(iii)
(iv)

STACK

Definition :- One side open, another side closed
 :- Last in first out [LIFO].
 :- Top is a variable which contains position of the newly inserted element.

Operations :- ADT of stack (what opn can perform)

- (i) push (stack, integer) = stack.
- (ii) pop (stack) = integer.
- (iii) isempty (stack) = Boolean;
- (iv) isfull (stack) = Boolean.

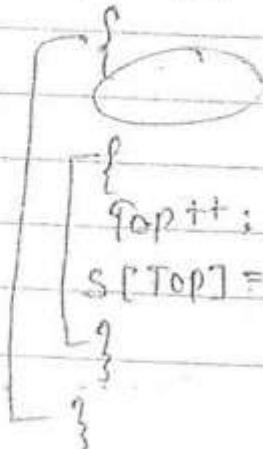
Push()

S N=5

When stack is empty

Top = -1 (initially)

Push(a)



4	c
3	d
2	e
1	b
0	a

Top = N-1 .

Q) int main()

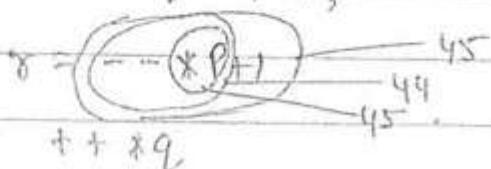
int m = 44;

int * p = & m;

int & r = m; ~~(r address is same as m address)~~

int n = (*p)++; means r is alias to m.

int * q = p - 1;



printf("m=%d, n=%d, r=%d", m, n, r);

{ }

m	p	n	r
44	1000	44	998
1000	2000	3000	4000

O/P 2.

a) 44, 46, 45

b) 45, 44, 45

c) 46, 44, 46

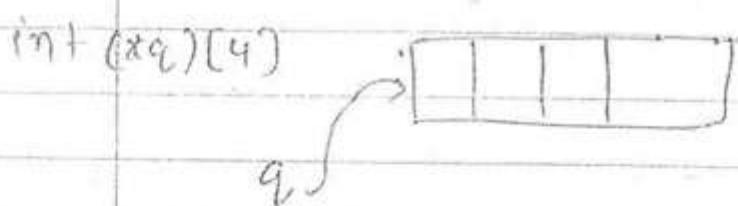
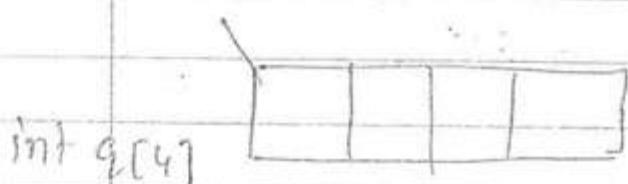
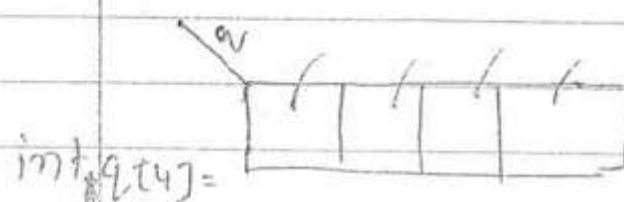
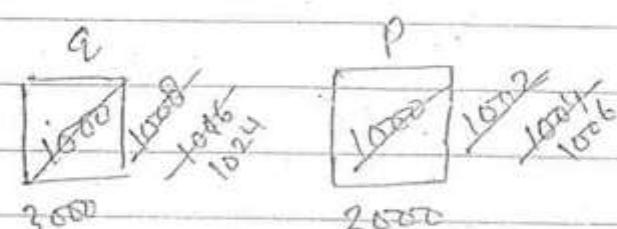
d) 46, 45, 46

11

2, 9, 0, 6}

Solⁿ

1600	02	04	06	08	10	12	14	16	18	20	22
5	7	5	9	4	6	3	1	2	9	0	6
0	1	2	3	4	5	6	7	8	9	10	11



O/P	a+i	p	q
i=0	1000	1000	1000
i=1	1008	1002	1008
i=2	1016	1004	1016
i=3	1024	1006	1024

main()

{ int a[3][3] = { 5, 7, 5, 9, 4, 6, 3, 1, 2, 9, 0, 6 };

int *p;

int (xq) [4];

p = q;

q = a;

for (i=0; i<3; i++)

{

printf ("%d %d %d %d", a[i], p[q])

p++;

q++;

}

}

Soln

	00	01	02	04
00	5	7	5	9
02	4	6	3	1
04	2	9	0	6

a[0] = 1000

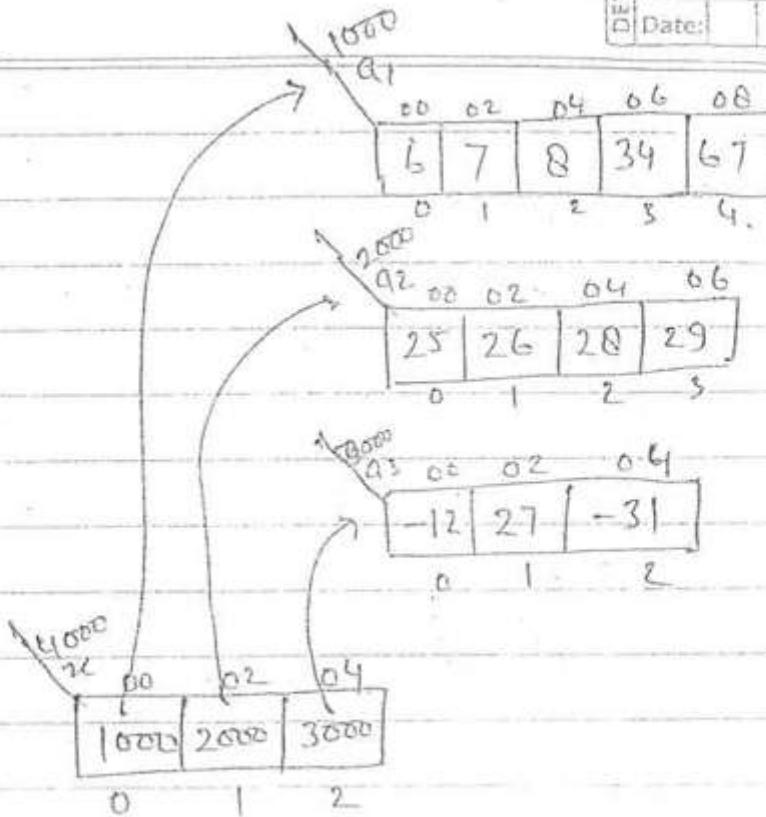
a[1] = 1008

a[2] = 1016.

i = 1

i = 2

i = 3



$$k(n(9+0)+2) = 9(6)(2)$$

$$\Rightarrow a[0][2] = \frac{(a+0)+2}{4880} \\ \frac{1020+4}{1024},$$

$\Rightarrow \pi(a[2])$

$$k(a+1)$$

X (480 + 4)

X-3020

-12

DELTA	Page No.	—
Date	—	—

$\text{int } a_1[] = \{ 6, 7, 8, 34, 67 \}$

$\text{int } a_2[] = \{ 25, 26, 28, 29 \}$

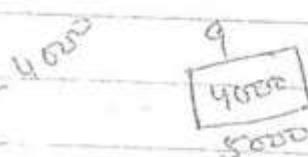
$\text{int } a_3[] = \{ -12, 27, -51, 9 \}$

$\text{int } *x[4] = \{ a_1, a_2, a_3 \}$

main()

```
{
    fun(x);
}
```

fun(int *a)



```
{
    printf("%d", a[0][2]); => 8
}
```

```
printf("%d", *a[2]); => -12
```

```
printf("%d", *++a[0]); => 7
```

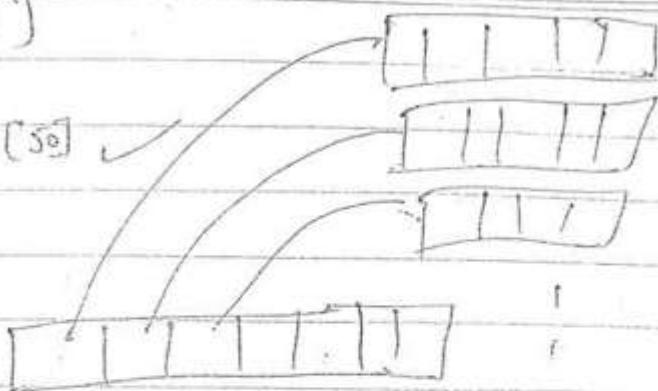
```
printf("%d", *(*(a + 0))); => 25
```

```
}
```

~~int * p[10]~~

int p[10][50]

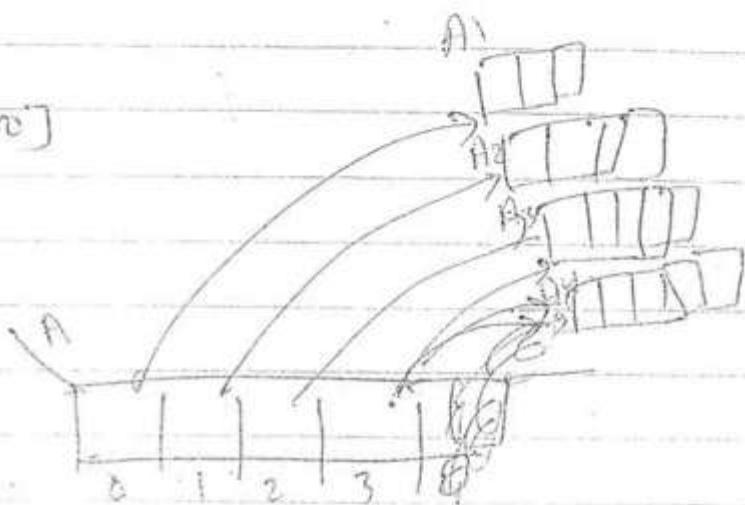
Setⁿ



A [0] [100]

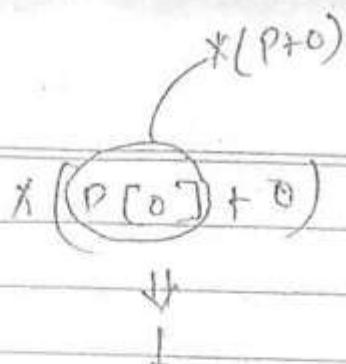
44

* A [0]



int A1[3], A2[4], A3[5], A4[6],

int * A[4] = {A1, A2, A3, A4}.



$$*(p+i) = p[i]$$

Solution

i	J	$*(*(p+i)+j)$
0	0	1
1	2	
2	3	
1	0	4
1	1	5
2	6	
2	0	7
1	1	8
2	9	

In this Problem

p is 1-D Array

but it give
behavior like
2-D Array
becoz

it give
array.

Which of the following is true for the following C declaration.

int A[10][10], & B[10]

- a) $B[5] = \& \text{address}$ ✓
- b) $B[5][5] = 10$ ✓
- c) $A[5] = \& \text{address}$ ✗
- d) $A[5][6] = 10$ ✓

n	ptr
$n+1 \Rightarrow 1006$	$\text{ptr}+1 \Rightarrow 1002$

~~#P~~

main()

{

int $a[3][3] = \{1, 2, 3, 4, 5, 6, 7, 8, 9\};$

~~ptr~~

int * $p[] = \{a, a+1, a+2\};$

for ($i=0, j \leq 2, j++$)

{

O/P $\Rightarrow 1, 2, 3, 4, 5,$

$6, 7, 8, 9$

for ($j=0, j \leq 2, j++$)

{

$p[i][j]$

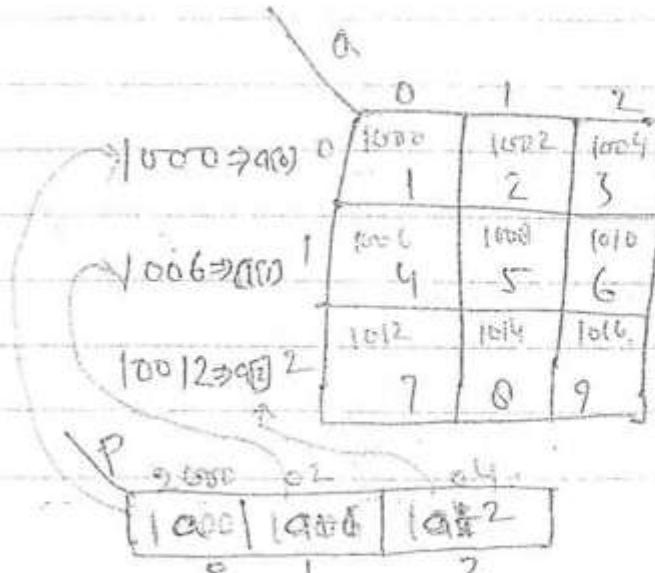
printf("%d", *(*(p+i)+j));

}

}

}

}



~~main~~

{

int n[3][3] = {2, 4, 3, 6, 0, 5, 3, 5, 1};

int *ptr;

ptr = n;

printf("%d", n[2]); $\Rightarrow 1012$

printf("%d", ptr[2]); $\Rightarrow 3$

printf("%d", *(ptr + 2)); $\Rightarrow 3$

Sol^m

		n[0]			n[1]			n[2]		
		0	1	2						
1000	0	2	4	3						$\Rightarrow n[0] \Rightarrow 1000$
1006	1	6	0	5						$\Rightarrow n[1] \Rightarrow 1006$
1012	2	3	5	1						$\Rightarrow n[2] \Rightarrow 1012$

ptr

1000

* (ptr + 2)

1010 + 2

* (1004)

5

$$\frac{a[4]+1}{1016} \quad | \quad \frac{k(a[1]+1)+1}{1016}$$

↓
1020

(1)

$$a[2]+1$$

$$\frac{k(1024+4)}{1020} \Rightarrow 1020$$

$\frac{k(k(x(a+0)+1)+1)}{1020}$

$x(a+0) = *a$

$*x a = a[0][0]$

$x(x(a+0)+0)$

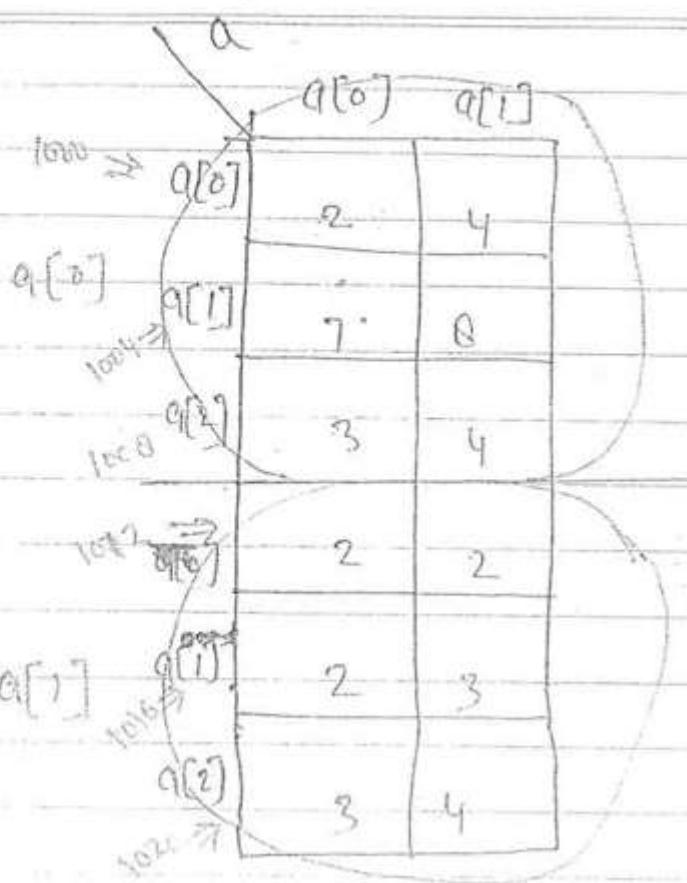
$x(a[0]+0)$

Ans

In 3-D Array first * means array second * means rows and 3rd * (last*) means column is selected.

3-D Array

DELTA	Page No.	_____
	Date:	_____



$$a = 1000$$

$$a+1 = 1012$$

$a[1][1] = 1012$ means it is selected.

$$(a+1)+1 \\ \Rightarrow 1024$$

$$a[1][1]+1 \\ \Downarrow \\ 1016$$

$$\frac{* (a[0] + i) + j}{20 + 1} \rightarrow a[i][j]$$

21 b[j]

$$* (a[0] + 0) \rightarrow a[0][0]$$

$$* (b+j)$$

$$* (a[i] + j)$$

$$* a[0][0]$$

$$* (* (a+i) + j)$$

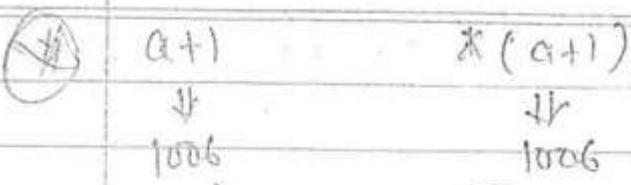
```
# Main()
{
```

int a[2][3][2] = { 2, 4, 7, 8, 3, 4, 22, 3, 5, 4 };

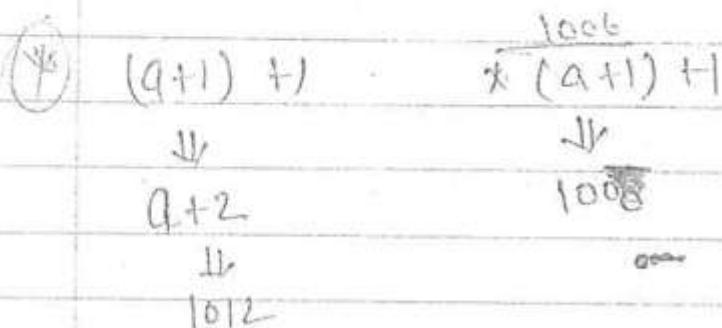
printf ("%d,%d,%d,%d,%d", ¹⁰⁰⁰a, ¹⁰⁰⁰a, ¹⁰⁰⁰a, ²a);

printf ("%d,%d,%d,%d", ¹⁰⁰²a, ¹⁰⁰²a, ¹⁰⁰²a, ²a);

printf ("%d,%d,%d,%d", ¹⁰⁰⁴a, ¹⁰⁰²a, ¹⁰⁰⁰a, ³a);

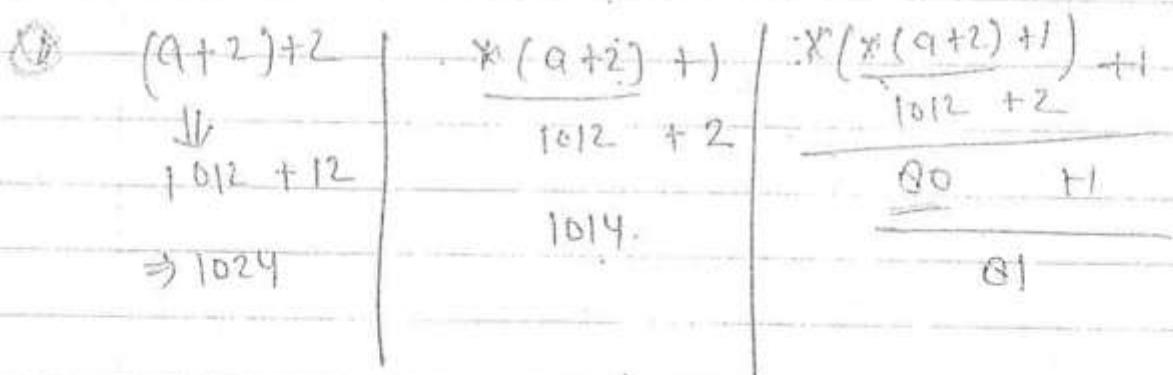


both are same but ~~&a~~ $\&(a+1)$
means that is selected.



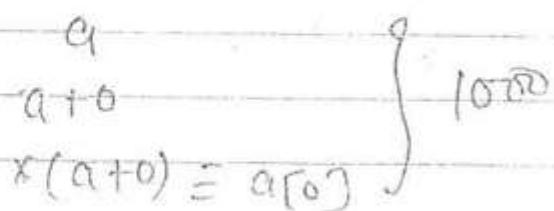
In 2D Array also there are components

Given 1st $\&(A[0])$ indicate row address



$\Rightarrow 1024$

Q1



* Bel'n two pointers addition, division and multiplication
is not possible bcoz there is no meaning.

~~**~~

Main()

{

int a[3][3];

Printf("%d", ((a == &a) && (*a == a[0])));

}

O/P = 1

	0	1	2
0	60	02	04
1	10	20	30
2	06	08	10
3	40	50	60
4	12	14	16
5	70	80	90

$\Rightarrow a[0] \Rightarrow 1000$

$\Rightarrow a[1] \Rightarrow 1006$

$\Rightarrow a[2] \Rightarrow 1012$

$$\begin{cases} a = 1000 \\ a[0] = 1000 \\ *a[0] = 1000 \\ *a = 1000 \end{cases}$$

$$a = 1000$$

$$a+0 = 1000$$

means In 2D array 0 rows skip

$$a+1 = 1006$$

" " " " " 1 row skip

$$a+2 = 1012$$

$$a[0][0] = 10$$

$$a[0][0] \Rightarrow *a[0][0]$$

$$*(r(a+0)+0)$$

~~Q3-1~~

Note

(*) for unary operator associativity is from right to left.

$++$ & $*$ both are having same associativity is right to left.

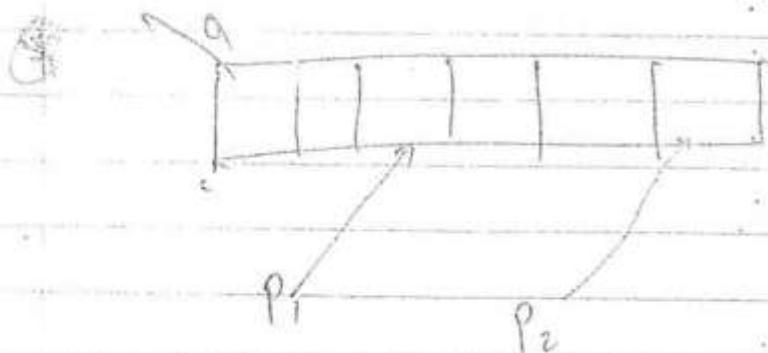
$x \text{ptr} ++$ in this $\text{ptr} ++$ is completed first

Note

(*) for the pointer variable we can add integer constant.

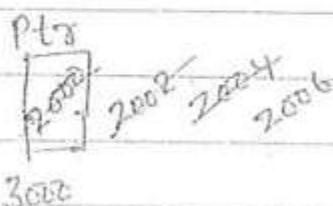
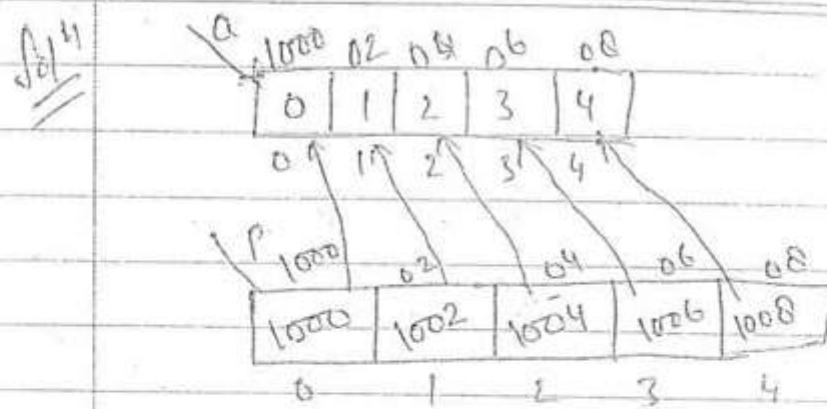
$p = p + i$ indicates p will point to after i elements from current place.

(*) For the pointer variable we can subtract integer constant.



$$p_2 - p_1 \quad (p_2 \geq p_1)$$

Subtraction b/w two pointers is allowed, but the condition is both with point to will same array.



11

Main

{

int arr{0,1,2,3,4}

int * ptr = {0, arr+1, arr+2, arr+3, arr+4}.

int ** pptr = &ptr;

ptr++;

printf("%d %d %d %d", *ptr-p, *ptr-q, *ptr);

*ptr++;

printf("%d %d %d %d", *ptr-p, *ptr-q, *ptr);

*(*ptr++);

printf("%d %d %d %d", *ptr-p, *ptr-q, *ptr);

*(*ptr++);

printf("%d %d %d %d", *ptr-p, *ptr-q, *ptr);

}

Main ()

```

    {
char a[5] = "Visual C++", b[5] = "Visual C++";
char *b = "Visual C++",
printf ("%d %d, sizeof(a), sizeof(b));
printf ("%d %d", sizeof(xa), sizeof(xb));
}
  
```

Sol

#2 Main ()

```

int a[5] = {0,1,2,3,4};
int xp[5] = {a, a+1, a+2, a+3, a+4};
int *ptr = p;
printf ("%d %d", a, *a);
printf ("%d %d", p, xp);
printf ("%d %d", ptr, *ptr);
}
  
```

X

Main()

57.

100	200	300	400	500	600
0	1	2	3	4	5

Char * str [] = { "frogs", "Dogs", "Nuts", "Die",
 "croc", "break" };

printf ("%d %d %d", sizeof(str), sizeof(str[0]));

}

4
6x2
4
12B.

4
100
44222
4
3B.

X

int a;

int a[5] =

10	20	30	40	50
0	1	2	3	4

size of(a)

if float

then

size of(a) = 20

in pa(s):

1	1	1	1	1
0	1	2	3	4

size of(a) = 10

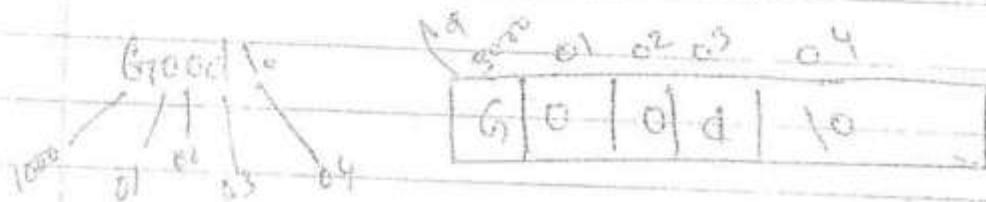
if float

then

size of(a) = 10

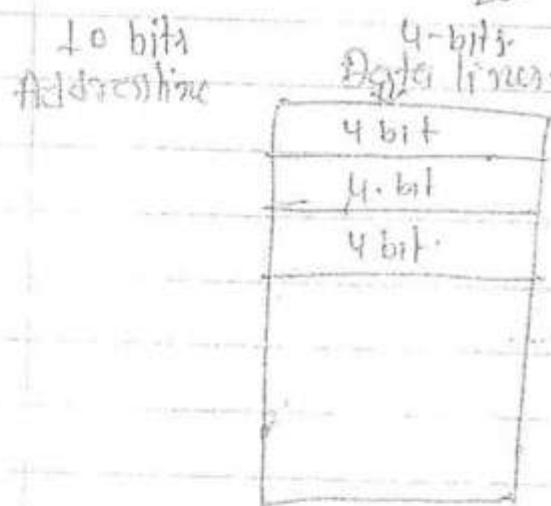
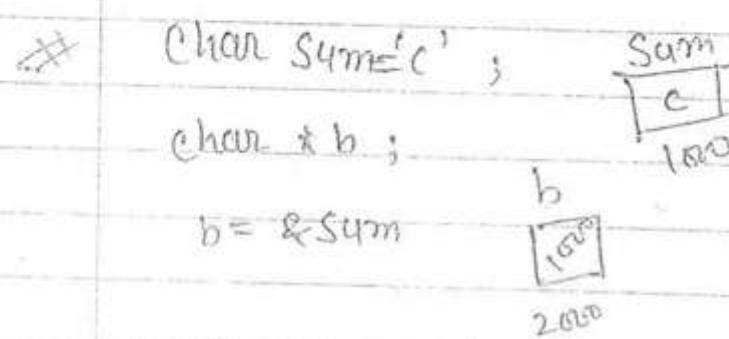
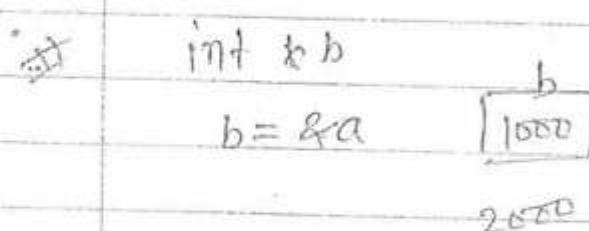
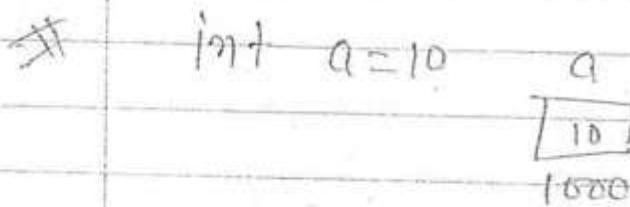
Main()

```
{  
char *p = "Good";  
char a[] = "Good";  
printf("%d%d%d", size(p), size(a), strlen(p));  
printf("%d%d%d", size(a), strlen(a));  
}
```



scanf("Good") = 4

strlen("Good") = 4



$$\Rightarrow 2^{10} \times 4 = 4 \text{ KB.}$$

16 B.
my node *p;
size of (p) = 2 B
size of (*p) = 16 B

void *p / not specified.
size of (p) = 2 B
size of (*p) = 16 B
error

Call by need

* integer $i = 100, j = 5;$

Procedure P(integer x)

{

Printf ($x + j$) $\Rightarrow 110$

$i = 200,$

$j = 20;$

Printf (x); $\Rightarrow 105$

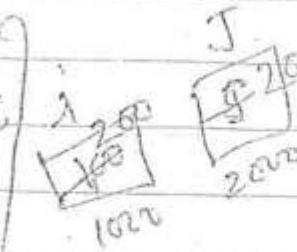
}

Main ()

{

P ($i + j$)

}



exactly calling
same
but it will evaluate
the expression
only once

Recently modified
thing will not be
affected.

Main () Assume:

Address line = 16 bit

char *p;

213

Printf (" %d %d ", sizeof (p), sizeof (*p)),

address

float x; b;

constant

Size of (b) = 2B

2B

Size of (x) = 4B

4B

* Consider the following Program . Call by name
Passing Parameters

integer i=100 , j=5

Procedure(integer u)

begin :

Printf (u+j) ; $\Rightarrow 110$

i = 200 ;

j = 20 ;

Printf (u) ; $\Rightarrow 220$

end ;

Main()

Call by Name :- 110, 220

{

p(i+j)

integer i=100 , j=5

Procedure(integer u)

begin int i=3000

Print(u+j) = 316

$i = 200$,

$j = 20$,

Printf (u) ; $\Rightarrow 120$

end;

Main()

p(i+j)

}

$$a = 1000, n = 6$$

~~(1)~~

$$\textcircled{2} \quad 12 + (1002, 5)$$

~~(1)~~

~~(2)~~

$$\textcircled{3} \quad 7 - f(1004, 4)$$

~~(1)~~

~~(2)~~

$$\textcircled{3} \quad 13 - f(1006, 3)$$

~~(1)~~

$$\textcircled{2} \quad 4 + f(1008, 2)$$

~~(1)~~

~~(2)~~

$$\textcircled{3} \quad 11 - f(1010, 1)$$

~~(1)~~

$$\textcircled{2} \quad 6 + f(1012, 0)$$

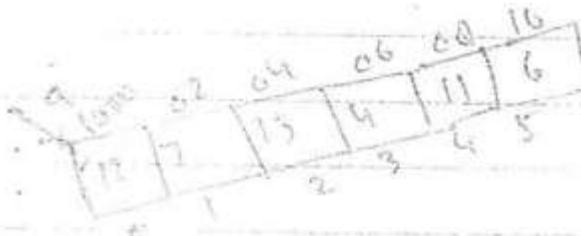
Q) What is the value printed by the following C. Program

```
#include <stdio.h>
int f (int x, int n)
{
    if (n <= 0) return 0;
    else
        if (x % 2 == 0)
            return x + f(x+1, n-1);
        else
            return x - f(x+1, n-1);
}
```

main()

```
{ int a [] = {12, 7, 13, 4, 11, 6};
```

```
printf ("%d", a[6]); }
```



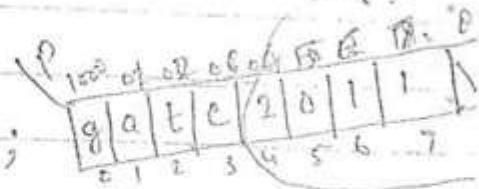
- a) -9 b) 5
c) 15 d) 19

Q) Consider the following C program.

Main

{

char p[] = "gate2011";



printf ("%s", p + p[3] - p[1]);

}

$$p + p[3] - p[1]$$

- a) gate2011
- b) 2011
- c) e2011
- d) 011

$$1000 + e - a$$

$$1000 - 101 - 97$$

$$\frac{1000 - 101 - 97}{1000}$$

Q) Consider the following C program.

int x;

Main

{

x = 5
p(&x)



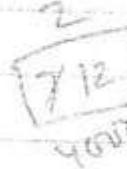
printf ("%d", x);

}

Void B(int z)

{

z += x



printf ("%d", z);

}

Void P(int *y)

{

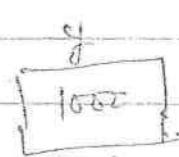
int $x = \frac{5}{2}y + 2;$

G(x);

$*y = \frac{x-1}{6};$

Printf("%d", x);

}



O/P - ?.

a) 12, 7, 5 b) 12, 7, 6

c) 14, 6, 6 d) 7, 6, 6.

~~xx8~~ Main () [Call by name]

```

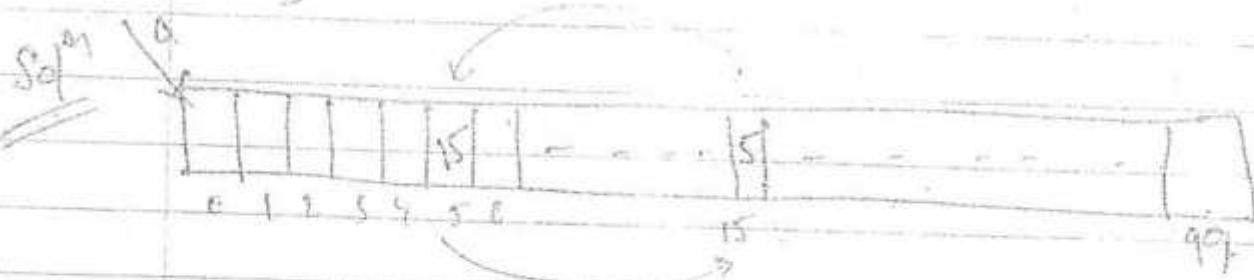
    {
        int i=5, A[100];
        A[i]=15;
        printf ("%d.%d", i, A[i]);
        swap(i, A[i]);
        printf ("%d.%d", i, A[i]);
    }
  
```



Swap (int c, int d)

```

    {
        int t;
        t=c;  $\Rightarrow$  t=5
        c=d;  $\Rightarrow$  i=a[i]
        d=t;  $\Rightarrow$  a[i]=t;
    }
  
```



Call by name

O/P = 5, 15

15, 5

Strict evolution

Call by Value -

Call by Reference -

Copy - restore -

Call by Name -

Call by need -

Call by Text -

Lazy evolution (SISI)

Note - In Lazy evolution Category we will pass the expression without execution which will leads to evaluating the same expression many time.

In the above program `c` is evaluated 5 times,

~~swap (int c, int d)~~

{ $c[m]$, m }

$$d = d + 5 \Rightarrow m = m + 5$$

$$d = d + 100 \Rightarrow m = m + 100$$

m will
be
evaluated
5 times

Name Conflict:-

If existing local variable name is as same as formal parameter then that problem is known as name conflict.

Existing Local variable name is replace by other name.

Main ()

```
{  
int c[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 } ;  
int m = 3 ;
```

```
printf ("%d %d", c[m], m) ;
```

swap (c[m], m) :

```
printf ("%d %d", c[m], m) ;
```

}

Swap (int c, int d)

```
{  
    c[m] = m ;
```

$$c = c + d \Rightarrow c[m] = c[m] + m$$

$$d = c - d \Rightarrow m \Rightarrow c[m] - m$$

$$c = c - d ; \Rightarrow c[m] = c[m] - m$$

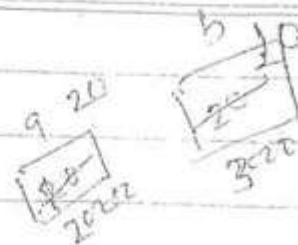
evaluted.
???

$$\begin{array}{r} 3 \\ + 9 \\ \hline 22 \end{array}$$

int a=100
main()



{
int a=10 , b=20



printf("%d %d", a, b);

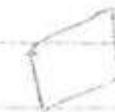
swap(a, b)

printf("%d %d", a, b);

swap(int c, int d)

a b

int t, a=500; q



t=c; t=q;
c=d; a=b;
d=t;

{} a++

Name conflict

If
new local variable call
be processed further
means existing local
variable will be ignored

C/P ⇒ 10, 20

20, 10

- int $a = 100$ $\begin{matrix} a \\ 100 \\ 100 \end{matrix}$
 main ()

{
 int $a = 10, b = 20$ $\begin{matrix} a \\ 10 \\ 20 \end{matrix}$

printf("%d %d", a, b);
 swap(a, b);

printf("%d %d", a, b);

swap(int c, int d)

{
 int t;

$t = c; \Rightarrow t = 10$, $\begin{matrix} t \\ 10 \end{matrix}$
 $a = 20$
 $b = 10$

$c = d; \Rightarrow a = b;$

$d = t; \Rightarrow b = t;$

{
 a + t

$a = 21, b = 10$

Q/F

Value	Name
-------	------

10, 20

10, 20

20, 10

21, 10

Call by Name

main()

int a=10, b=20;

printf ("%d %d", a, b)

Swap(a, b)

printf ("%d %d", a, b);

}

Swap(int a, int b)

{
in t;

t = c; \Rightarrow t = a;

c = d; \Rightarrow a = b;

d = t; \Rightarrow b = t;

}

Output (without swapping)

\Rightarrow 10, 20 (after swapping)

8

What is the value printed by the following C program?

```
int f(int u int xpy, int xppz)
```

{

```
int y, z;
```

u	94	PPZ
147	11000	20000
4500	5000	6000

 $x \& PPZ = 1;$
 $z = x \& PPZ$
 $x \& py = z;$

y	z
1	5
1000	8000

 $y = x py;$
 $u + = 3$
 $\text{return}(x + y + z);$

}

void main()

{

```
int c, d, a, b;
```

 $c = 4, b = &c, a = \&b;$
 $\text{printf}("%d,%d", f(c,b,a))$

}

 $4 \& 4 \& \$$

a	2000
1000	2000
2000	3000

a) 10 b) 19

c) 21 d) 22

Note--

Call by copy ^{swipe} will give different Answer
 comparing with Call by reference. If the
 parameter passed to the funⁿ follows
 aliasing.

Procedure (int a, int b, int c)

begin (int a, int b, int c)

$\% a=2$ 3000 6000 5000

$c=a+b;$ a b c
 end.

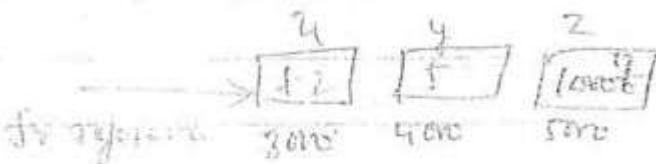
begin

int x=1, y=5, z=1000;

p(x, y, z);

Print (x, y);

end.



o/p - Call by value :- 1, 5

copy-before - 2, 5

reference - 2, 5

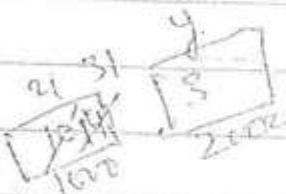
Q) Consider the following C Program.

Program P1()

```
{
    int n=10, y=3;
    fun1 (y, 21, 15);
    printf ("%d,%d", n,y);
}
```

fun1 (int n, int y, int z)

```
{
    y = y+4;
    z = y+y+n;
}
```



Aliasing

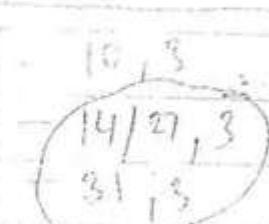
two variables
having same
addresses



Call by Value.

Copy Restore

REFERENCE



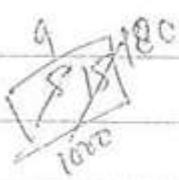
aliasing

Stack structure

Y

Main()

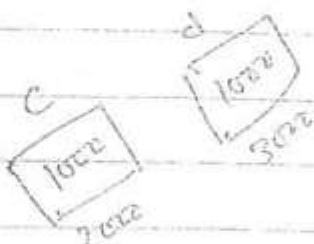
{
int a=5



Test.(a,a)

printf(a);

for (int i= , int j=)



{
i=

15
10
5
5*2

c = d * 2 ; i = c + (d * 2)

d = c - 3 ; j = d * (i - 3)

{
}

15
12
100

O/P \Rightarrow 100

a[0]=5

Q

Consider the following Program.

Q

Main()

{

int a=5;

a
5
100

Test (&a)

printf(a);

Test (int c, int d)

{

c+=d*2; $\Rightarrow c = c + (d * 2)$

c
15
200

d
60
300

d*=c-3; $\Rightarrow d = d * (c - 3)$

d
5
15-3

}

Call by Value

Copy - value

Call by Reference

15/60
100

i=4

a[i+i] = ++i

or

a[4]=5

a[5]=6

altering

i=4

5=++i

a[5]=6

a[5]=6

selected

3. Call by Copy - Restore [Indirectly modify]

Main()

{
 int a=10, b=20; $\begin{matrix} a \\ \checkmark \\ 10 \end{matrix}$ $\begin{matrix} b \\ \checkmark \\ 20 \end{matrix}$

printf("%d,%d", a, b);

swap(a, b);

printf("%d,%d", a, b);

{ } out

Swap (int c, int d)

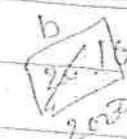
 $\begin{matrix} c \\ 10 \\ 300 \end{matrix}$ $\begin{matrix} d \\ 20 \\ 400 \end{matrix}$ {
 int t; t = c;
 c = d;
 d = t; $\begin{matrix} t \\ 10 \\ 500 \end{matrix}$ O/P - 10, 20
20, 10

Main()

```
int a=10, b=20;
```



```
printf("%d%d%d", a, b);
```

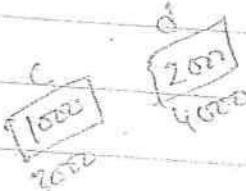


```
swap(a, b);
```

```
printf("%d%d%d", a, b);
```



```
swap(index, index);
```



```
exchange(c, d);
```

```
exchange(intc, intd);
```



```
intc;
```



```
intd;
```

```
intc = t;
```

10, 20

20, 10

* C language by default follows call by value

C II supports both call by value and call by reference;

Pap. No.	Date:
11111111	11111111

Swap without temp Variable.

$$a = 10, b = 20$$

$$a = a + b$$

$$b = a - b$$

$$a = a - b$$

Call by Reference (Directly changes in original)

Main ()

```
{  
    int a = 10, b = 20;  
    printf("%d %d", a, b);
```

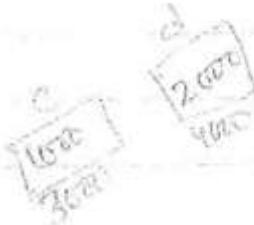


```
    swap(a, b);
```

```
    printf("%d %d", a, b);  
}
```

Swap (int & , int &)

```
{  
    int t;  
    t = a;  
    a = b;  
    b = t;  
}
```



Output - 20, 10
20, 10

so are
place only

2

called swap
swap!

~~Q8~~ Write a recursive C-Program to find the max^m element of the given array.

~~Soln~~

10	20	30	40	50
1	2	3	4	5

10, 20, 30, 40, 50
| J

Printmax (a, i, J)

```

{ if (i==J)
    {
        Print (" %d " a[ ] )
        return ;
    }
    else
    {
        Printmax (a, i+1, J)
        if (a[i]<a[J])
            Printmax (a, i+1, J)
        else
            Printmax (a, i, J-1)
    }
}

```

eg

50 10 170 110 60
 1 2 3 4 5
 i j i=j T j

Types of Recursion

- (1) Tail-Recursion.
- (2) Non-Tail-Recursion.
- (3) Nested Recursion
- (4) Indirect Recursion.

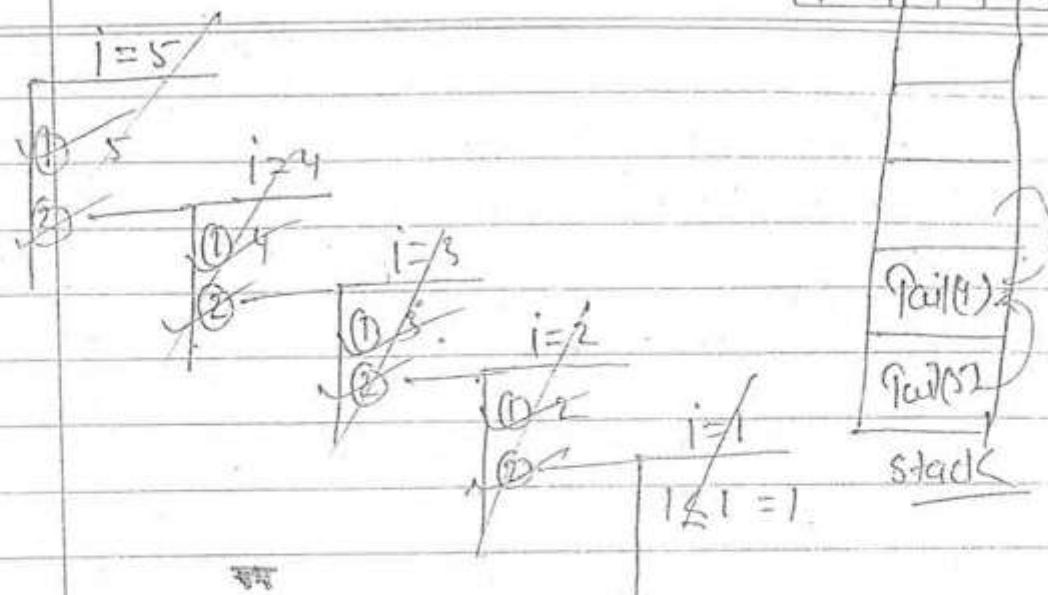
Tail-Recursion

Tail (int i) i=5

```

if (i ≤ 1) return
else
{
    (1) printf ("%d", i);
    (2) Tail (i-1)
}
    
```

O/P = 5, 4, 3, 2, 1.



⇒ In the given recursive program if recursive call is there at the end of the program then that program is called Tail recursive Program.

⇒ Tail recursive program is not advisable bcoz unnecessarily we are wasting stack space

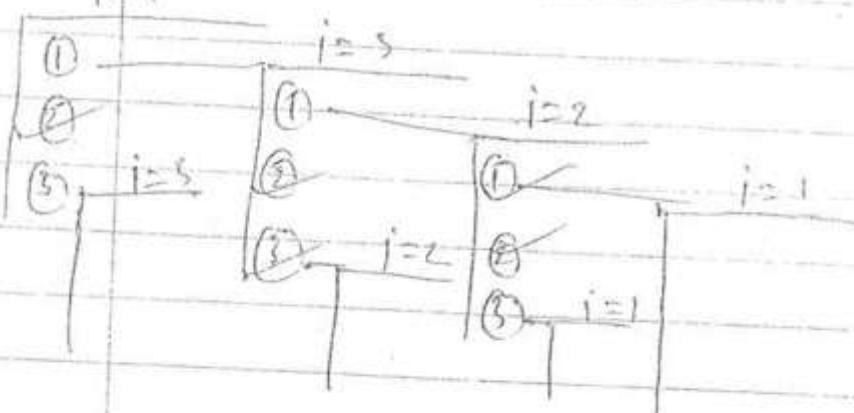
⇒ The advantage with the tail recursive program is we can write easily non-recursive equivalent program for the given tail recursive program with the help of one for loop.

Non-Tail Recursive

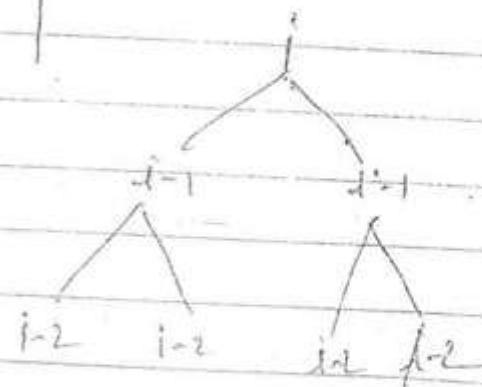
Non Tail (int i)

```
i = 4
{
    if (i ≤ 1) return;
    else
    {
        NonTail (i-1)
        printf ("%d", i)
        NonTail (i-1)
    }
}
```

i = 4



O/P = 2, 3, 2, 4, 2, 3, 2



⇒ In the given recursive program after recursive call some stmt. are there then that recursion is called Non-Tail Recursion.

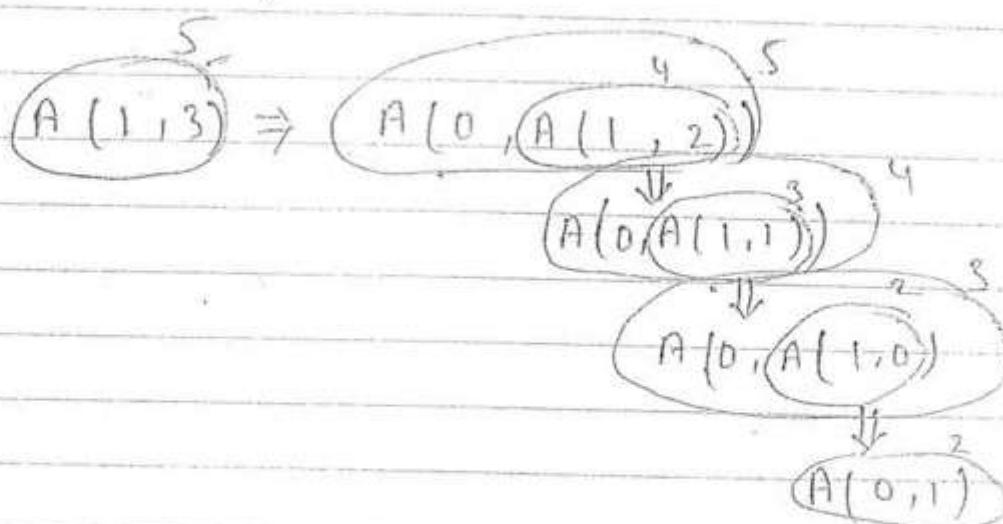
Note

We are utilizing the space efficiently and but it is difficult to write equivalent to write with equivalent non-recursive program.

Nested Recursion

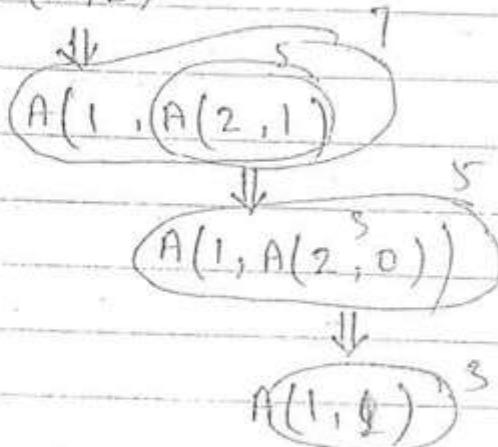
Ackermann's Recursion Relation.

$$A(m, n) = \begin{cases} n+1 & \text{if } m=0 \\ A(m-1, 1) & \text{if } n=0 \\ A(m-1, A(m, n-1)) & \text{otherwise} \end{cases}$$

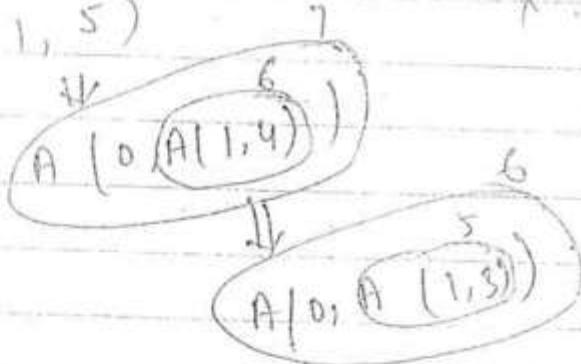


find a 7

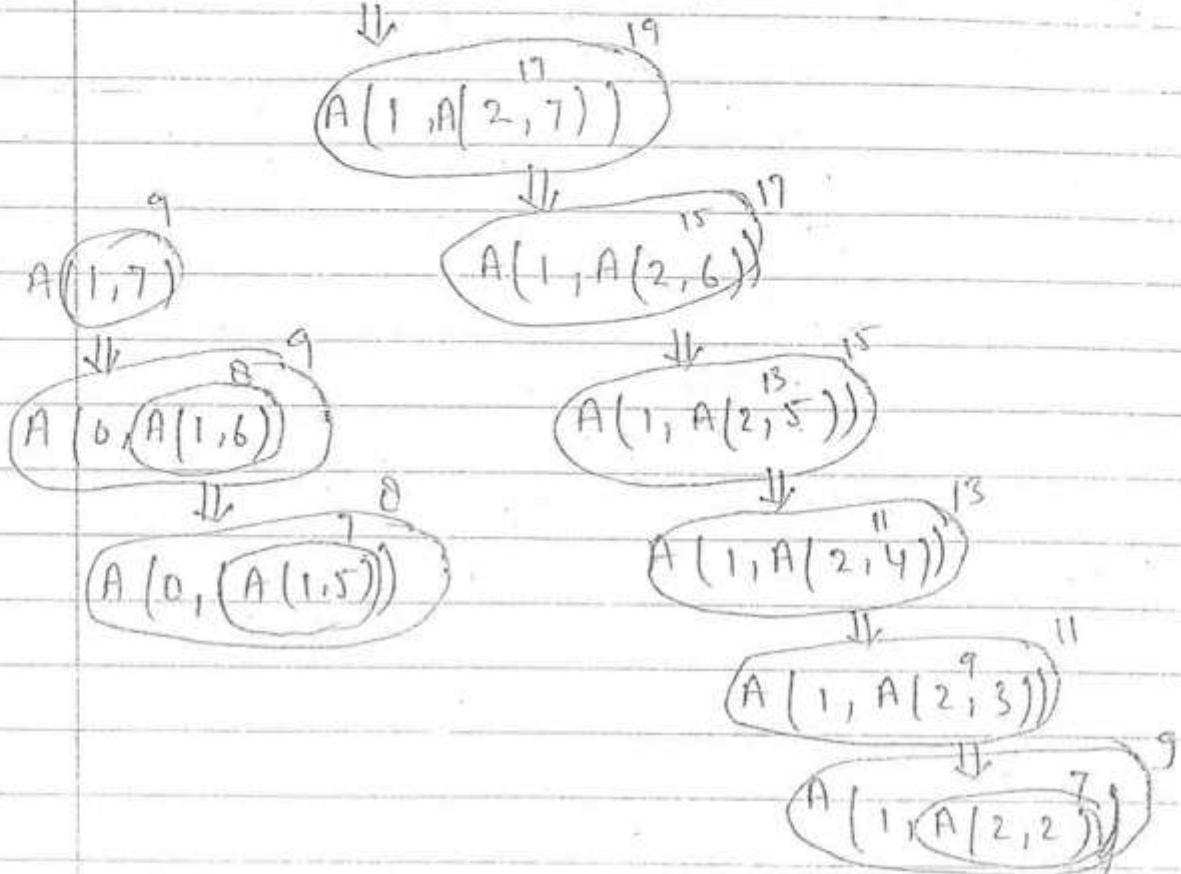
$$A(2,2)$$



$$P(1, \frac{5}{\sqrt{1}})$$



find $A(2, 8)$



Indirect Recursion

Aorder(int i)

Let
 $I = 3$

```

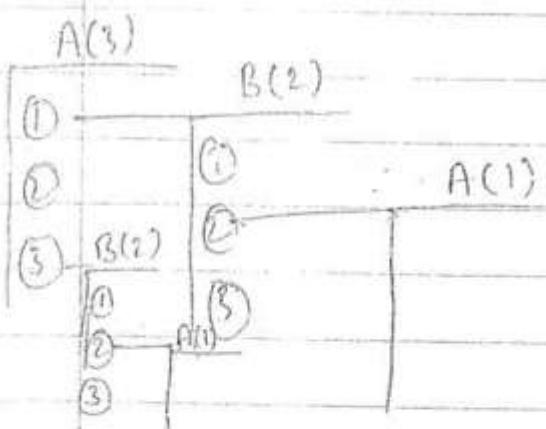
  {
    if ( i ≤ 1 ) return ;
    else
      {
        ① Border( i-1 );
        ② printf( "%d", i )
        ③ Border( i-1 )
      }
  }
  
```

Border (int i)

```

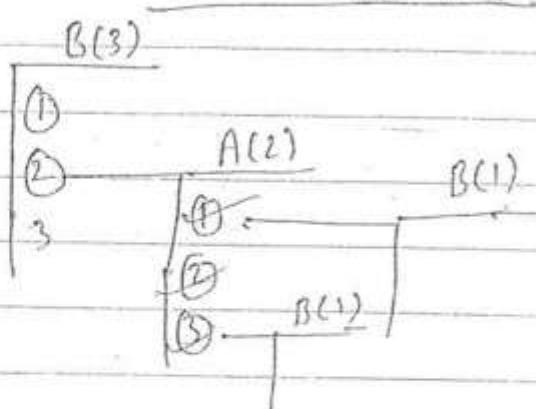
    {
        if ( i < 1 ) return;
        else
        {
            ① printf( "%d", i );
            ② Border( i - 1 );
            ③ printf( "%d", i );
        }
    }
  
```

I/P : Border (3)



O/P \Rightarrow 22322

I/P: Border (3)



O/P: - 3 2 3

Infix to Postfix Conversion

Ex

a + b * c
 Operator
 Operands

Compare operator and operands
 operands have higher priority.

Symbol	Priority	Associativity
① + , -	1	L-R
② * , /	2	L-R
③ ↑	3	R-L

~~x~~ Infix :- $a + b * c$

Postfix :- $a b c * +$

Prefix :- $+ a * b c$

Infix :- $a + b - c$

Postfix :- $a b + c -$

Prefix :- $- + a b c$

~~x~~ Infix :- $a + (b * c) / (d + e + f) * g - h + i$

Postfix :-

$$\begin{array}{c}
 a \frac{b c *}{3} d \frac{e f + +}{1} / g \frac{* h + i}{2} - \\
 \hline
 4
 \end{array}$$

Prefix :-

$$\begin{array}{c}
 + - + a \frac{* b c}{3} \frac{\overbrace{d + e}^f}{2} d c b \\
 \hline
 4
 \end{array}$$

~~Infix :- a+b/c *d - e↑f *g + h*i↑g/a↑b*c /d-e~~

Postfix :- $a+b/c*d-e \uparrow f * g + h * i f t_1 * c/d - e$

$$a + t_5 - t_6 + t_9 - e$$

Index:- $a + (b / c) \times d - e \uparrow f \times g + h \uparrow i \uparrow a \uparrow b \times c / d - e$

Postfix:- abc / d * + e f ^ g x - h i a b ^ ^ * c * d / + e -

84

cont

Prefix

- + - + * / b c d x ^ e f g / x x h ^ i r a b c d e.

Note

Infix: $a + b * c$

Postfix: abc * +

Prefix: + a * b c

Operands
order same but
operator order
changing.

Postfix

2 3 4 * +

2 1 2 +

1 4
one.

Infix

$2 + 3 * 4$

$2 + 12$

1 4
Many

Prefix

+ 2 * 3 4

+ 1 2

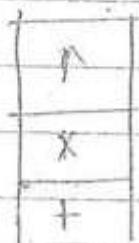
Many

Note:-

- 1 In order to evaluate infix expression many scans are required becoz we don't know where is higher priority order.
- 2 In order to evaluate prefix expression many scans are required becoz we don't know where is higher priority order.

38 In order to evaluate Postfix expression one stack is required, becoz higher priority will come at first.

* $a+b*c \uparrow d$



$abcd \uparrow x +$

$\begin{matrix} \text{Top} \\ \text{lower} \\ \text{Higher} \end{matrix} \Rightarrow \text{push(next)}$

* $a+b*c \uparrow d - e$



$a b c d \uparrow x + e -$

$\begin{matrix} \text{Higher} \\ \text{lower} \end{matrix} \Rightarrow \text{Pop(Top)}$
 $L-R \Rightarrow \text{Pop(Top)}$
 $R-L \Rightarrow \text{push(next)}$

one

try

* a↑b↑c



a b c ↑↑

a+b-c



a b + -

Prefix :- ~~a+b*c/d*efg*d-c+b.~~



Postfix

a b c * d e f g * / d * + c - b t

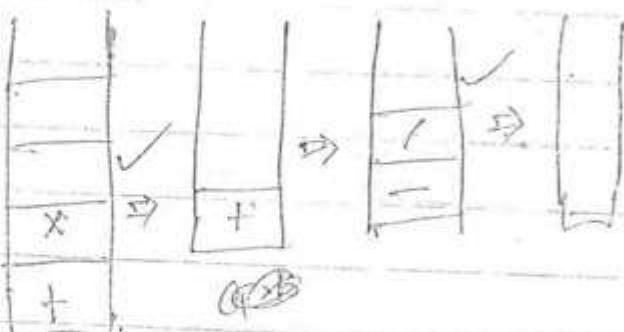
Infix : - $a + b / c * d - e \uparrow f \times g + h \times i \uparrow a \uparrow b \times c / d - e$



abc / d + e f g x - h i a b c x c x
d / + e -

\Rightarrow While converting Infix expression into Postfix expression we are pushing operator on the stack which is known as operator stack.

Q What is the maxm size of operator stack during the conversion of infix expression $a + b * c - d / e$ into postfix.

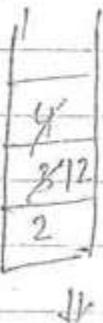


abc + de Max 2

E1d-e

Postfix Evaluation

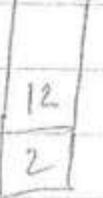
1/p : $2 + 3 * 4$ op
 Post : ~~(2 3 4) *~~ +



$^4 \text{op}_2 \Rightarrow \text{pop}()$
 $^3 \text{op}_1 \Rightarrow \text{pop}()$

$r = \text{op}_1 \text{ op } \text{op}_2$

Push(r)



$^1 \text{op}_2 = \text{pop}()$

$^2 \text{op}_1 = \text{pop}()$

$r = \text{op}_1 \text{ op } \text{op}_2$

Push(r)



Postfix-

Q

Postfix Evaluation

{
while (character is not null)

{
if (character is operand)
Push (character)

else

{
 $OP_2 = \text{Pop}()$
 $OP_1 = \text{Pop}()$

$r = OP_1 \text{ character } OP_2$

Push (r)

}

}

Print f ("%", d ", Pop());

}

$$OP_2 = 1^2$$

$$OP_1 = 2$$

$$r = 2 + 1^2$$

Push (r)



Q If -, * and \$ are used Subtraction, multiplication and exponential.

- ① (i) - is higher than * and the \$ (lowest)
(ii) All are L-R associative.

$$I/P: - 3-2*4 \$ 1 * 2 \$ 3$$

- a) 512 b) 80 c) 51 d) 4096.

- ② all are R-L.

$$2 * 2 - 1 \$ 1 \$ 4 - 2$$

- a) 256 b) 2 c) 25 d) 2.

Solⁿ

$$\begin{array}{c}
 * \\
 / \quad \backslash \\
 32 - 4 * 12 + 5^3 \\
 \boxed{3} \quad \boxed{4} \quad \boxed{5}
 \end{array}
 \quad
 \begin{array}{c}
 2^4 \\
 \times \\
 2^2 \\
 \boxed{16} \\
 \boxed{16}^3 = (2^4)^3 - 2^{12} \\
 = 4096
 \end{array}$$

$$\begin{array}{c}
 - \\
 / \quad \backslash \\
 221 - 4 * 142 + 5^3 \\
 \boxed{2} \quad \boxed{4} \quad \boxed{5}
 \end{array}
 = 221 - 4 * 142 + 5^3$$

E Evaluate the following expression.

$$1 * 2 \wedge 3 * 4 \wedge 5 * 6$$

- a) 32^{30} b) $16^{\wedge 30}$ c) 49151 d) 173458 .

Sol



$$1 2 3 \wedge 4 5 \wedge 6 *$$

E Assume that the operators $*$, $-$ and $+$ are left to right associative and \wedge is R-L associative. The order of priority from high to low is $(\wedge, *, +, -)$.

The Postfix expression corresponding to the following Infix expression.

$$a + b * c - d * e + f$$

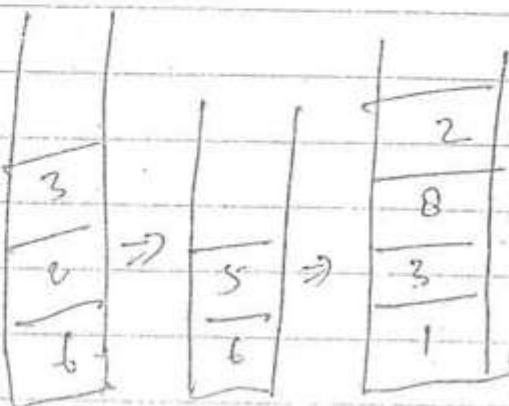
a) $a b c * + d e f n -$ b) $a b c * + d e n f g n -$

c) $a b + c * d - e n f g n d)$

Q What is the max^m size of the operand stack while evaluating the Postfix expression.

$6 \ 2 \ 3 \ + \ - \ 3 \ 8 \ 2 \ / \ + \ *$

Sol



Ans

ans

Prefix to Postfix Conversion.

✓ $a * b + c \uparrow (d * e + b - c)$

(1)

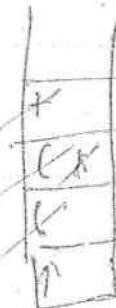


g b c d e * b + - c -

(2)

✓ $c \uparrow (a + b) * d)$

(3)



c a b + d * \uparrow

(4)

(5)

Prefix

Postfix

① a (minimal)

② ~~t a b~~

③ ~~* a * b c~~

④ ~~+ x a b / c d~~

⑤ ~~+ - * (a b c d) e f~~

$a b / c * d - e f \uparrow +$

⑥ ~~* - * a b c d - * / * a b c d + a b~~

\downarrow
 $a b c e / d - a b * c / d \uparrow a b + - *$

~~*~~ $(a b c d) - \uparrow / (* a b c d) + a b$

Prefix

* a + b c

↓

Postfix: - a b c + *

Infix: - a * (b + c)

* a[6] = "a b c d e"; $\Rightarrow S_1 = "a b"$

Substr (a, 1, 3, temp)
bcd $\Rightarrow S_2 = "c d"$

Subtract (S₁, S₂)

i = Is Operand (Symbol)

Prejo Post (Prefix, Postfix) * + a b - c d \ n
+ a b c d e f g

length = strlen (Prefix)

if (length == 1)

{
Postfix[0] = Prefix[0];

Postfix[1] = '10';

return (Postfix);

else

{
OP[0] = Prefix[0];
OP[1] = " ";

Substr(Prefix, 1, length-1, temp)
m = find(temp)

Substr(Prefix, 1, m, opnd1)

Pre to Post (opnd1, Prefix1)

Substr(Prefix, 4, 3, -cd bc)

m = find(temp)

8th Substr(Prefix, m+1, m+n, opnd2)

Pre to Post (opnd2, Postfix2)

stret(Postfix1, Postfix2)

stret(Postfix1, OP*) $\Rightarrow ab+$

strcpy(Postfix1, Postfixn) \xrightarrow{abt}

} $ab+cd-$ $ab+cd-\star$

}

(a)

Red color is
a example
and black
color is also
another
example.

$+ ab - cd$ ab
 $0\ 1\ 2$

find(str)

{
 $\{ ab - cd$
 $0\ 1\ 2\ 3\ 4$

$b - cd$
 $0\ 1\ 2$

length = strlen(str)

if (str[0] == operand)

return(1);

else

{

Substr(str, 1, length - 1, temp)

m = find(temp)

1 $ab - cd$

Substr(str, m+1, length - 1, temp)

2 5 $b - cd$

find(temp)

$b - cd$

return(m + n + 1)

}

3

Fibonacci Series.

n	0	1	2	3	4	5	6	7	8	9	10
$\text{fib}(n)$	0	1	1	2	3	5	8	13	21	34	55

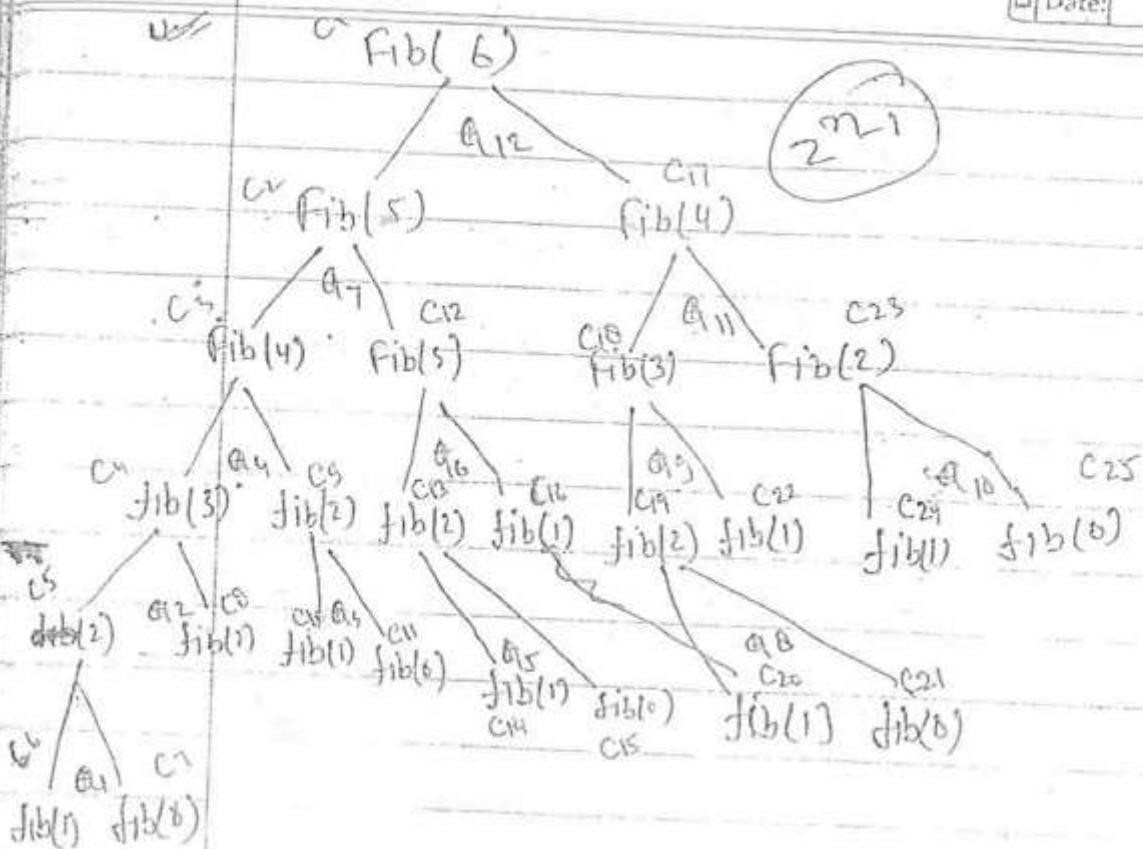
$$\text{fib}(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{otherwise} \end{cases}$$

$\text{fib}(\text{int } n)$

```

    {
        int f;
        if (n==0) return 0;
        else
            if (n==1) return 1;
            else
                {
                    f = fib(n-1) + fib(n-2);
                    return f;
                }
    }
}

```



Complete binary tree

2³⁷-1

* The "fun" calling sequence in any programming language is pre-order traversal.

Q In fibonaci of 10,000 after how many funcn.
Call it addition taken place.

801ⁿ after 100th call 10:21 1st addition will
be done.

1) Note

In fibonaci of n after $n+1$ function called after 1st addition will be done.

2) In fibonaci of n is there any funⁿ call after last addition ?.

Solⁿ no - becoz after last addition parent is available. (means final result availb)

3) In fibonaci of n is there any addition after last funⁿ call.

Solⁿ Yes. becoz after C_{25} there are three addition. a_{10}, a_{11}, a_{12} .

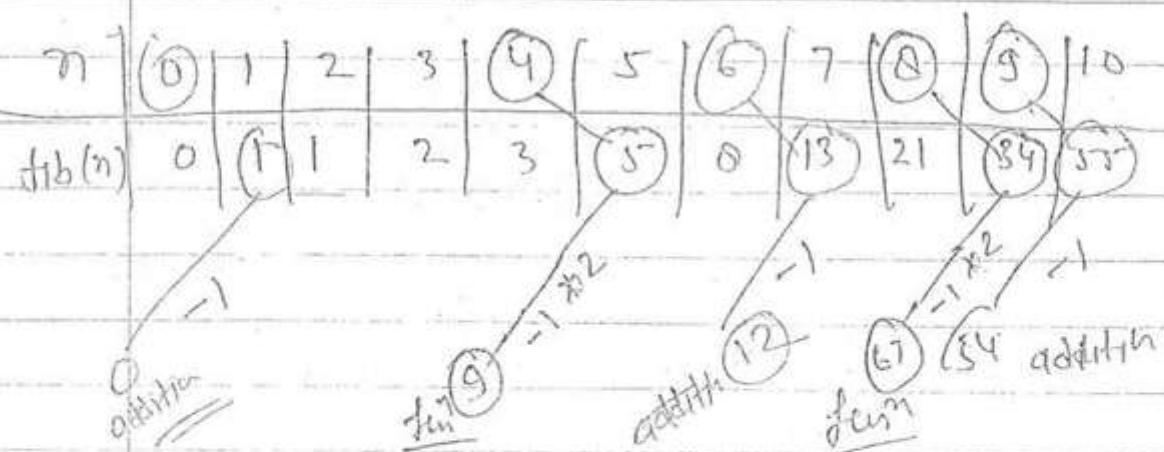
4) To Evaluate fibonaci of n value How many fun will be called.

$f(0)=1, f(1)=1, fib(2)=3, fib(3)=5, fib(4)=9$
 $f(5)=15, f(6)=25, f(7)=41, f(8)=67, fib(9)=109$
 $f(10)=177.$

5) In fibonaci of n how many additions will be performed.

$f(0)=0, f(1)=0, fib(2)=1, fib(3)=2, fib(4)=4$

$f(5)=7, f(6)=12, f(7)=20, f(8)=35$
 $f(9)=54, f(10)=88$



Note

The no of addition required to find fibnici of n is.

$$\text{addition} = \text{fib}(n+1) - \text{value}$$

(1)

(2)

(3)

The no fun^n calls. in fibonic of n .

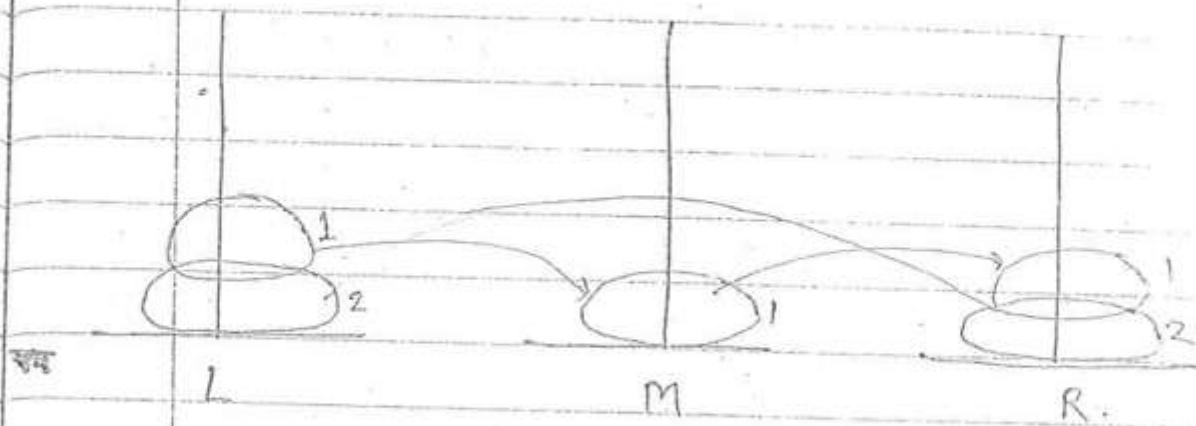
$$\text{function} = 2 * \text{fib}(n+1) - \text{value}$$

(1)

(2)

(3)

Towers of Hanai



- ① At a time only one disk.
- ② Only large size disk smaller size disk allowed.
- ③ All the disks will be there in 3-towers only.

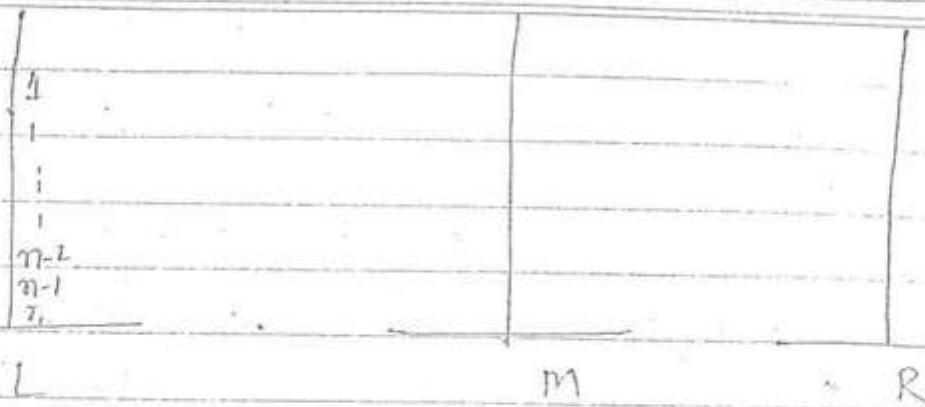


1
2
3
4
5

L

M

R.



TOH(n, L, M, R)

{

- ① TOH($n-1, L, R, M$) *left to right middle using right.*
 - ② MOVE($L \rightarrow R$)
 - ③ TOH($n-1, M, L, R$)
- }

#

TOH(n, L, M, R)

{

if($n == 0$) return;

else

{

① TOH($n-1, L, R, M$)

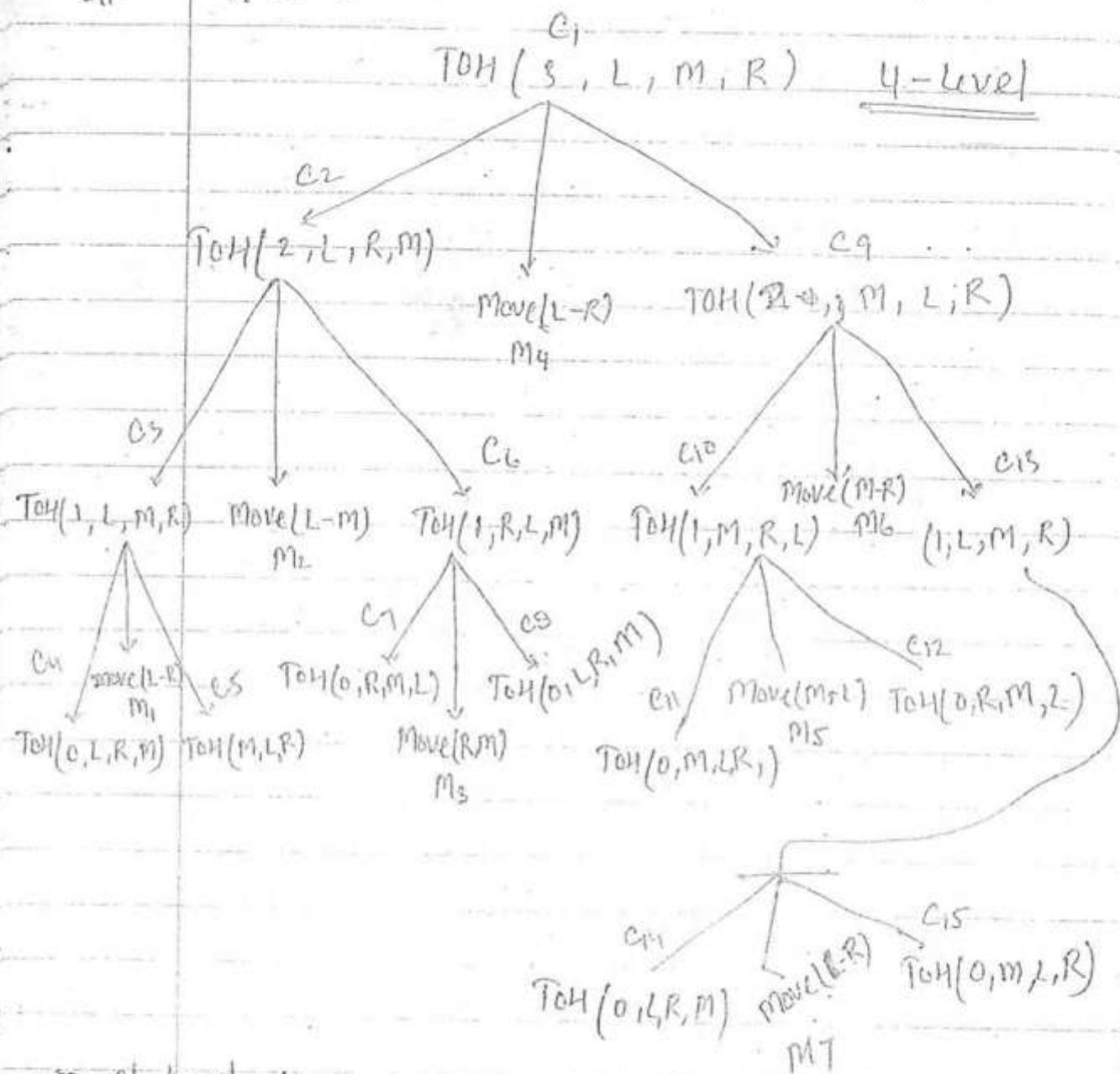
② MOVE($L \rightarrow R$);

③ TOH($n-1, M, L, R$);

}

}

#

 $n = 3$ 

no of level = 4

no of nodes = $2^n - 1$ no of funⁿ calling = $2^4 - 1 = 15$

no of move = 7

① (L - R)

② (L - M)

③ R - M

④ R - L

⑤ M - L

⑥ M - R

⑦ L - R

Q Is there any funⁿ call after last move.

Solⁿ Yes after m₇ there is C₁₅.

Q Is there any move after last funⁿ call.

Solⁿ No, after C₁₅ there is no move.

Q After How many funⁿ call first move will be taken place.

Solⁿ (n+1)

Q After How many funⁿ call there is a move from (M - R)

Solⁿ after 12 funⁿ call.

Q How many funⁿ will be call in TOH n.

Solⁿ TOH(0) = 1 TOH(1) = 3 TOH(2) = 7

TOH(3) = 15 TOH(4) = 31

$$T_{OH}(0) = 1 = (2^{0+1} - 1)$$

$$T_{OH}(1) = 3 = (2^{1+1} - 1)$$

$$T_{OH}(2) = 7 = (2^{1+2} - 1)$$

$$T_{OH}(3) = 15 = (2^{1+3} - 1)$$

$$T_{OH}(4) = 31 = (2^{1+4} - 1)$$

Note:

The number of 2^n calls in $T_{OH}(n)$

$$\boxed{2^{n+1} - 1}$$

Q

How many move are there in $T_{OH}(n)$.

$$T_{OH}(0) = 0 =$$

$$T_{OH}(1) = 1 = (2^1 - 1)$$

$$T_{OH}(2) = 3 = (2^2 - 1)$$

$$T_{OH}(3) = 7 = (2^3 - 1)$$

$$T_{OH}(4) = 15 = (2^4 - 1)$$

$$T_{OH}(n) = 2^n - 1$$

$$\boxed{T_{OH}(n) = 2^n - 1}$$

function Returning Pointer

int x fun(int a, int b)

```
{
    int u, y;
    u = 10, y = 20
    u = u + y
    u = u * a
    y = u + y / b
    return &y
}
```

fun returning integer pointer.

function Pointer

Main()

```
{
    int display() ↳ int a
    int (*fp) () ↳ int b
    fp = display; ↳ b = a
    (*fp)();
    display ↳ fp
    int display()
}
```

```
{
    printf("Hi ");
}
```

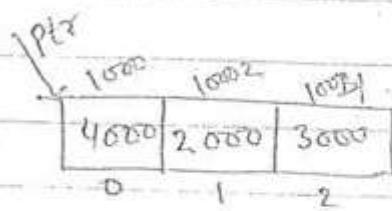
fp is a funⁿ which contains funⁿ address.

funⁿ name indicates base address.

Q. Consider the following Program.

Main()

```
{  
    int (*ptr[3])( );
```



ptr[0] = aaa;

ptr[1] = bbb;

ptr[2] = ccc;

(*ptr[2])();

}

aaa()

{

printf("aaa");

}

bbb()

{

printf("bbb");

}

ccc()

{

printf("ccc");

}

O/P : ccc

$\text{int } q \rightarrow$ 1 integer value

$\text{int } q[5] \rightarrow$ array integer value,

$\text{int } *q \rightarrow$ variable pointer.

$\text{int } *q[5] \rightarrow$ Array of variable pointer.

$\text{int } (\text{xa})() \rightarrow$ 1 funⁿ pointer

$\text{int } (*q[5])() \rightarrow$ Array of funⁿ pointer

$\text{int } (\text{xa})[10]$

a is a pointer pointing to array of 10 elements

Q What does the following C statement declaration?

$\text{int } (\text{xt})(\text{int } x)$

f is a pointer to a funⁿ which will take integer pointer as input which will be return integer as output.

f
 ↗
 (int *)
 {
 ↗ return(&u)
 }

Q

What does the following C Statement declare.

Void.

Void (*abc) (int, (void (*def) ()))

Sol

a bc is a pointer to a funⁿ which will take two parameter as input integer.

- ① def is a ptr to a funⁿ will return void.
- ② Which will return void.

Q

What does the following C-Statement declare?

Sol

Char * p (*q [5]) ()()

- ① array of five pointers point into a funⁿ which will return a pointer. which will pointing to a funⁿ, which will return their pointer.

Q) What does the following C statement declare:

$\Rightarrow (\text{char} * (\text{x}))(\text{x PTR}[N])();$

Sol:

Array of n pointer to funⁿ which will returning pointers to fun which will return char pointer.

Q) Match the following statement.

Group - I

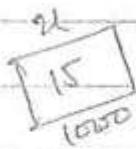
- | | |
|---|---------------------|
| (1) A pointer to an array
of 8 floats. | a) float *(xf) () |
| (2) A pointer to an array
of 8 pointer to float | b) float (xf) () |
| (3) A pointer to a fun ⁿ that
returns float. | c) float (*a)[8] |
| (4) A pointer to a fun ⁿ
that return a pointer
to a float. | d) float * (xa)[8]. |

Q) Consider the following C-program.

int main().

```

    {
        int u=15;
        printf ("%d", fun(s, &u));
    }
  
```

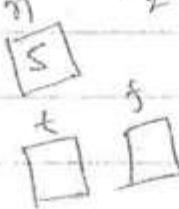


fun (int n, int *fp)



```

    {
        int t,f;
        if (n<=1)
    }
  
```



```

    {
        *fp = 1;
        return 1;
    }
  
```

t = fun(n-1, fp)

t = t + *fp

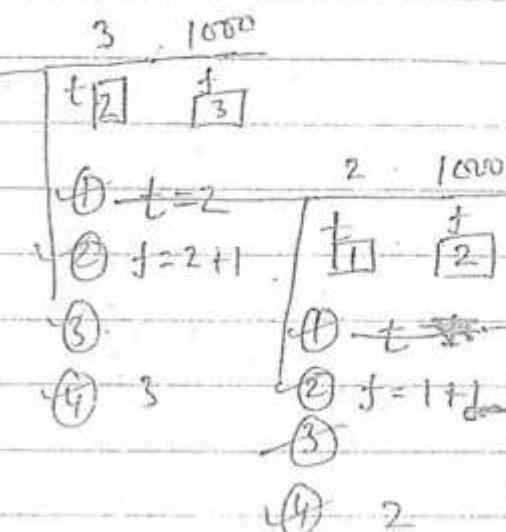
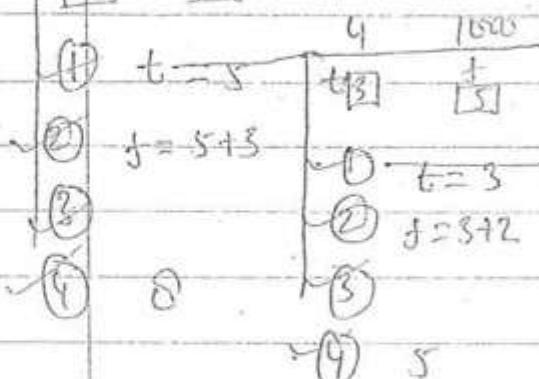
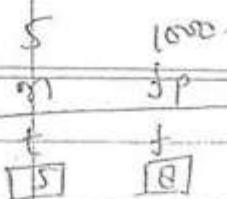
*fp = t

return t; }

}

The value pointed is.

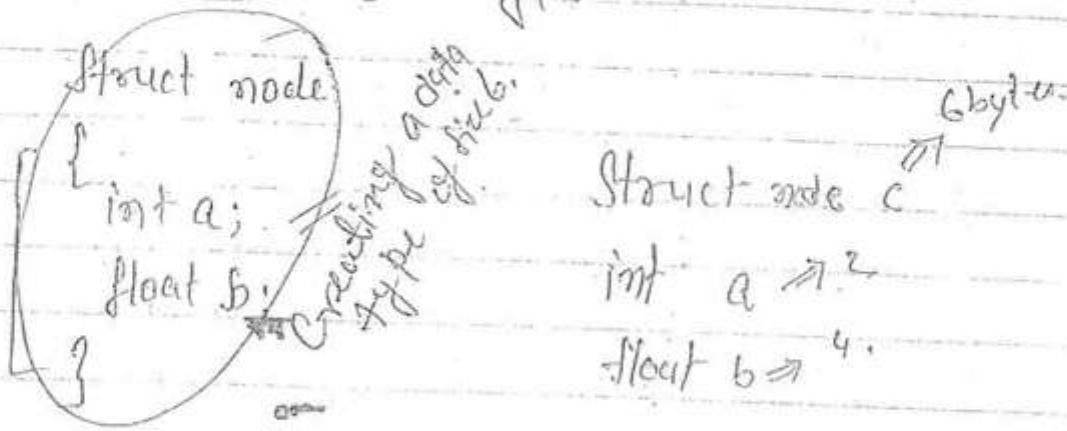
- a) 6 b) 0 c) 14 d) 15



1 1000

Structure

↳ User defined
Data type.



→ Struct node

```

    struct node
    {
        int a;
        float b;
    };
  
```

(not allowed)

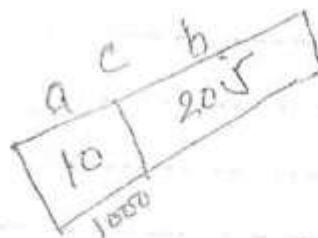
Main ()

{

Struct node c = {10, 20.5};

Printf ("%d", a);

g.



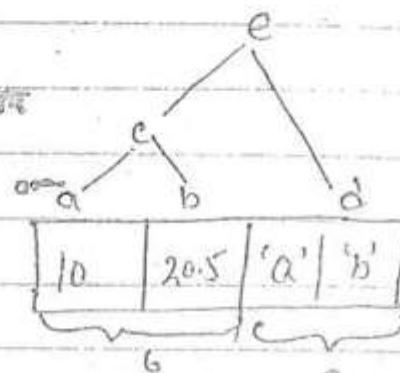
⇒ Struct node \Rightarrow^b

```
{  
    int a;  $\Rightarrow^2$   
    float b;  $\Rightarrow^4$   
};
```

6 bytes

Struct node \Rightarrow^B

```
{  
    struct node c;  $\Rightarrow^b$   
    char d[2];  $\Rightarrow^2$   
}
```



Main()

```
{  
    struct node e = {10, 20.5, {'a', 'b'}};
```

```
    printf("%d", e.c.a);
```

e.c.b

e.d[0]

e.d[1]

}

→ Struct node

```
{
    int a;
    float b;
}
```

int a[20],

Struct node d [20].

array of 20 struct node
variable.

Main()

{

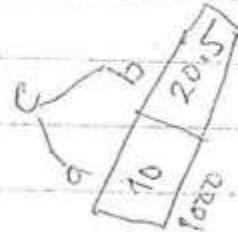
① Struct node c = { 10, 20.5 };

② Struct node pd;

③ d = &c

Print f (" %d ",

) ;



int q = 10

q
10
1000

② int * b

b
1000
2000

③ b = & q

④ b

⑤ x b

d = 1000

* d = a, b

(* d), a = 10

(* d), b → 20.5

(* d), a → d → a

(* d), b → d → b

O/P ⇒

Main()

↑ 9 bytes

struct a

{ ↑ 1 byte

char ch[7] ↑ 20 bytes
char *str; ↑ 9 bytes

}

↓ 9 bytes

struct b

{ ↑ 1 byte

char &c; ↑ 1 byte
struct a sst; ↑ 9 bytes

}

↑ 11 bytes

struct b s2 = { "Raipur", "Kapur", "Jaipur" };

printf("%s.%s.%s", s2.c, s2.sst.str);

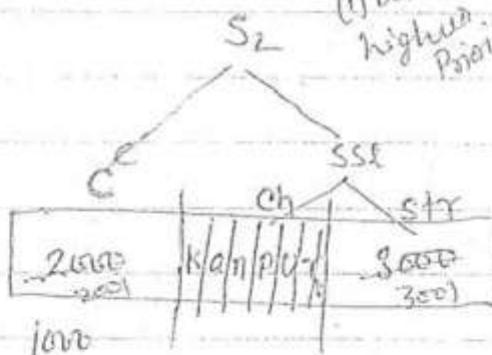
printf("%s.%s", +s2.c, +s2.sst.str);

2nd

3rd

(1) Dot have
higher
Priority

Raipur



O/P → Raipur . Jaipur.

cipur , cipur

"Raipur."

↑ 2003
↑ 2002
↑ 2001

"Jaipur"

↑ 3002
↑ 3001

Consider the following C Program.

Main()

```

    {
        Struct st;
        Char str;
        Int i;
        Struct st *ptr;
    }

```

Struct st a [] = {{"Nagpur", 1, a+1}, {"Raipur", 2, a+2}, {"Kanpur", 3, a+3}}

Struct st *pa = a;

Int i;

for (i=0; i<=2; i++)

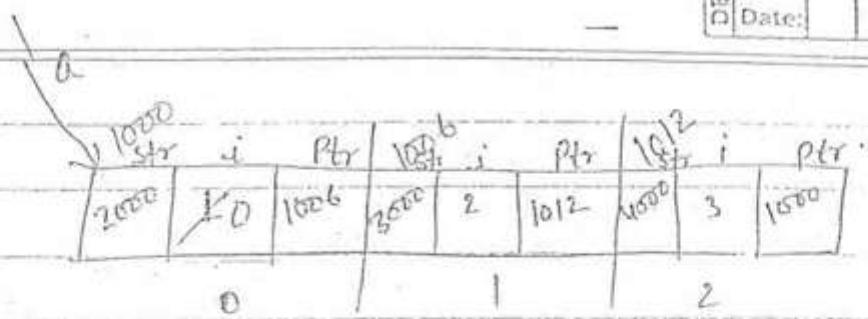
```

    {
        Printf ("%d", a[i].i);
        Printf ("%s", a[i].str);
    }

```

} O/P → 0; 1 | 2

Nagpur; Raipur; Kanpur



L

Main()

```

    {
        struct S1
        {
            char s[2];
            int i;
            struct S1 *p;
        };
    }
  
```

by Jn.

① $\text{++}(\text{ptr} \rightarrow 2)$

② $a[(\text{++} \text{ptr}) \rightarrow 1].z$

③ $a[--(\text{ptr} \rightarrow p \rightarrow i)].z$

4000

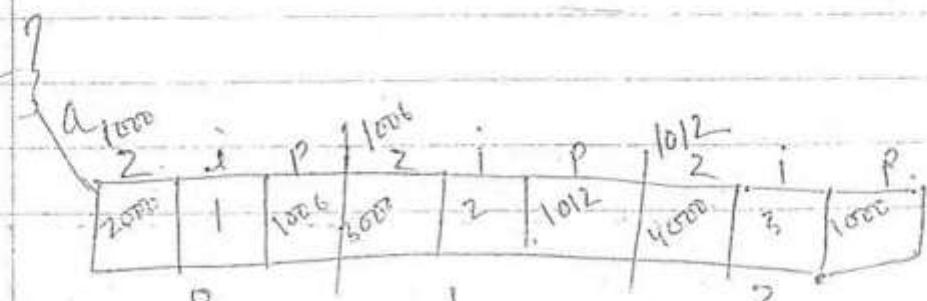
struct S1 a[] = {{"Nagpur", 1, a+1},
 {"Raipur", 2, a+2},
 {"Kanpur", 3, a}};

Struct S1 * ptr=a

~~Printf("%s", ++(ptr->z))~~ \Rightarrow Nagpur

~~Printf("%s", a[(++ptr)->z].z)~~ \Rightarrow Raipur

~~Printf("%s", a[--(ptr->p->i)].z)~~ \Rightarrow Kanpur



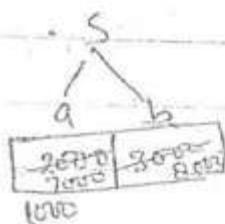
```
Typedef Struct P
{
    char *a, *b;
} t;
```

```
Void f1(t s);
```

```
Void f2(t *p);
```

```
Main()
```

```
{
    t s = {"a", "b"};
    printf ("%s.%s", s.a, s.b);
    f1(s);
    printf ("%s.%s", s.a, s.b);
    f2(&s);
}
Void f1(t s)
{
    s.a = "u";
    s.b = "v";
    printf ("%s.%s", s.a, s.b);
    return;
}
```



Void f1(t * p)

{
p → a = "V"
 7000
 2000 4000
 2000 8000

p → b = "W";
 8000
 10000
 6000

· printf ("%s %s", p→a, p→b);
 ↓ ↓
 v w.
}.

c/p:-

- a) a b b) a b c) a b d) a b
 u v u v u v u v
 v w a b u v v w
 v w v w v w u v

Y

~~2/11~~

Struct Test

```
{ int i;
char *c;
};
```

Struct test st [] = { { 5, "become" }, { 4, "better" }, { 6, "Jungle" }, { 8, "ancestry" }, { 7, "brothe" } }

Main()

{

Struct Test * p = st;

Pt = 1;

printf("%d.%s", (int)(p + Pt), (char*)(p + Pt));

printf("%c", *(P + Pt));

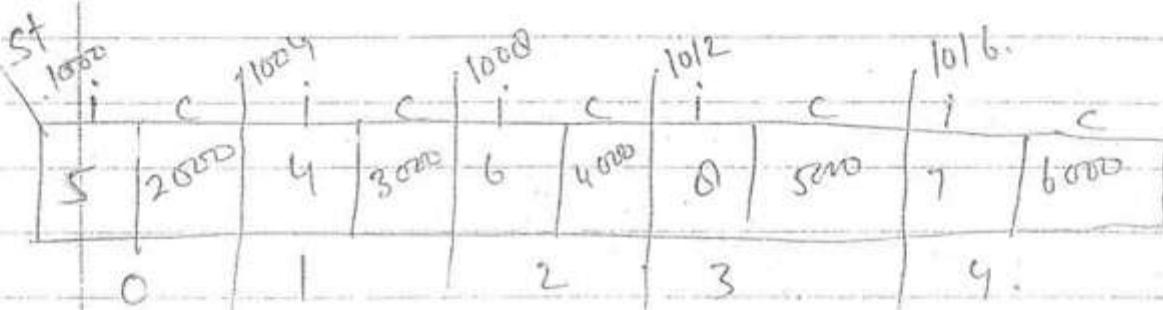
printf("%d,%d", *(P + Pt), *(P + Pt + 1));

Prints : (1.f, p-c);

}

$$0 \mid P := -$$

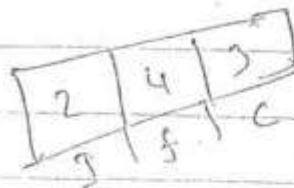
- a) jungle, n, ə, nʌdʒl
 - b) effe~~r~~, u, b, ~~w~~, ʌŋgl
 - c) ceffey, k, b, ʌŋglə
 - d) effey, u, ə, nɛstə



struct s

```
{ int a; → 2
    float b; → 4 }
```

```
char c = "abc"; → 3
    / c[3]
```



3
Size of (struct s) = 9.

Unions

```
union s
{ int a; → 2 }
```

```
    float b; → 4
```

```
    char c[3]; → 3
```

};

Size of (union s) = 4.
(max)



Union s1

```
union s1
{ int a; → 2
    float b; → 4
    char c[3] → 3
    struct s d; → 9 }
```

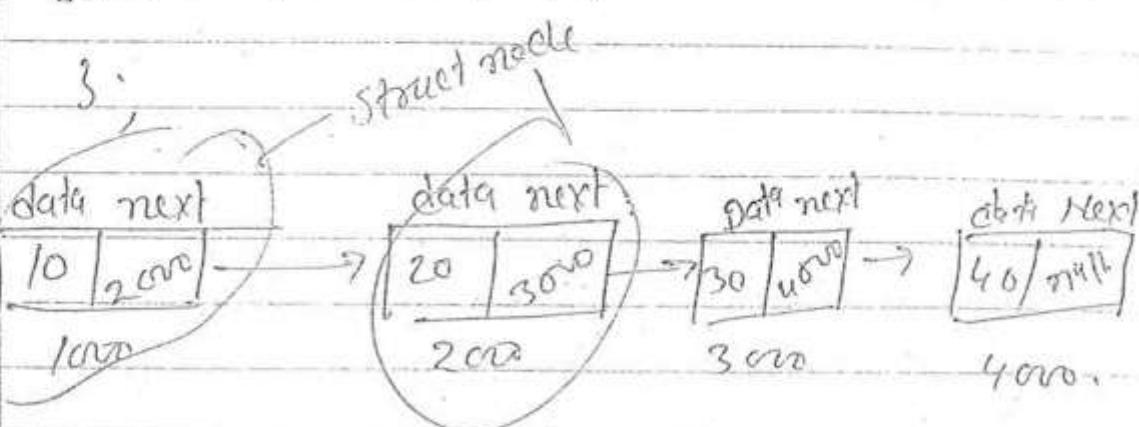
3
Size of (union s1) = 9.

~~S~~ Struct node

S
int data;

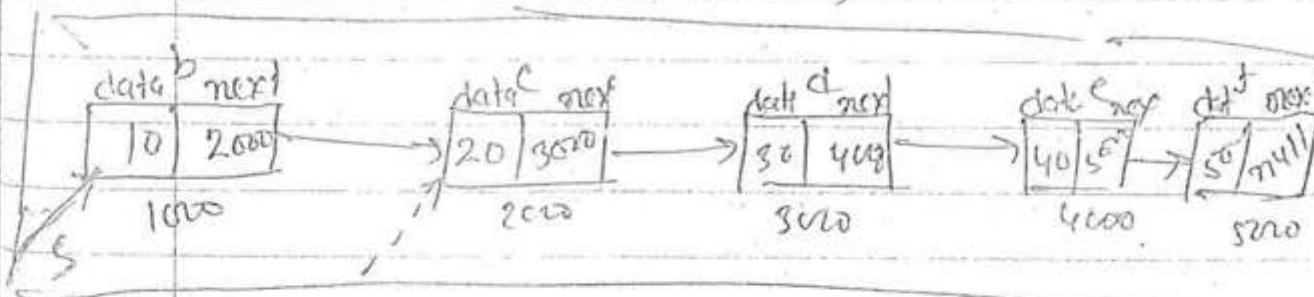
struct node * next;

S:



self referential
data structure

Struct node b, c, d, e, f;



b. data = 10

b. next = 2000

s++ ; 2000 X

↓

1004

$s = s + 1 \Rightarrow 2000 X$



loop ✓

$s = s \rightarrow \text{next} \Rightarrow 2000$

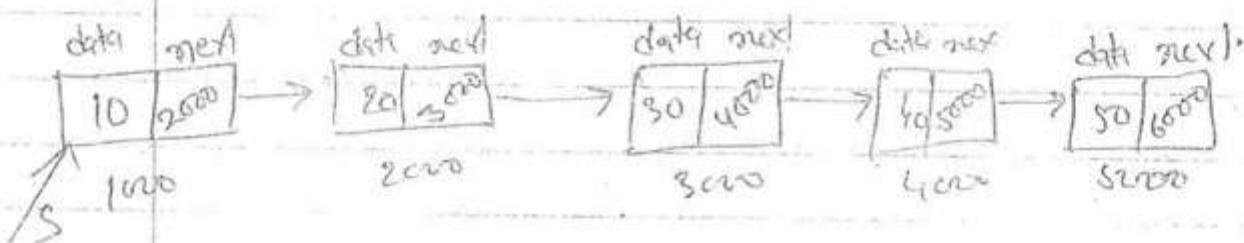


loop X

Q

solⁿ

Q Write a C Program to print the data
in the given linked list.



Solⁿ

Struct node

s
1000

```

struct node {
    int data;
    struct node *next;
};

void display (struct node *ps)
{
    while (s != null)
    {
        printf ("%d", s->data);
        s = s->next;
    }
}

```

$s \Rightarrow s \rightarrow \text{next};$

{

}

$0/p \Rightarrow .10, 20, 30, 40, 50$

Q. Write a C Program to print data value of a last node.

~~sel~~
sel

100

$s \rightarrow \text{data} \Rightarrow 100$
 $s \rightarrow \text{next} \Rightarrow 200$

$s \leftarrow$ Dangling pointer
[null]

$s \rightarrow \text{data}$ } Segmentation error
 $s \rightarrow \text{next}$ } "

Print last (struct node *s)

{
if (s == null)

return (-1);
else

{ while (s -> next != null)

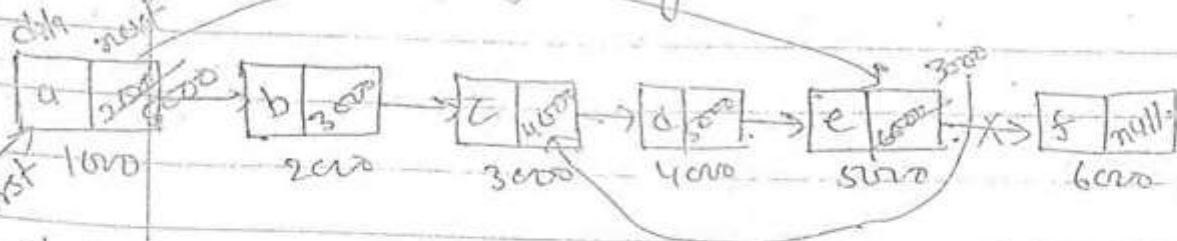
{
s = s -> next

}
printf ("%d", s -> data);

~~Q 4~~

Consider the following linked list.

2



What will be the output of after following execution.

W

(i) struct node * p;

①

(ii) p = (first → next) → next

②

(iii) first → next = (p → next) → next

③

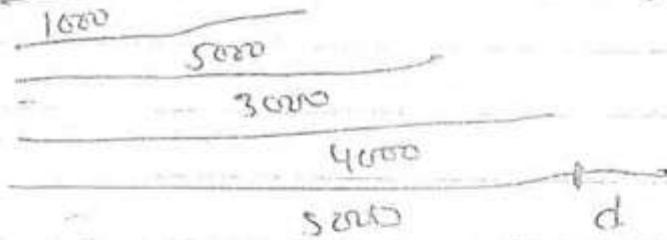
(iv) (p → next) → next → next = p

④

(v) printf ("%c", first → next → next → next → data).

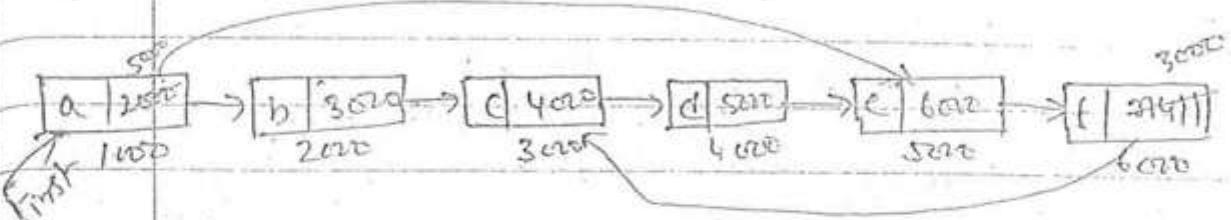
⑤

Soln



O/p ⇒ d.

Q Consider the following linked list.



What will be the o/p after following execution.

(1) struct node * p;

(2) $p = \text{first} \rightarrow \text{next} \rightarrow \text{next};$

(3) $\text{first} \rightarrow \text{next} = p \rightarrow \text{next} \rightarrow \text{next};$

(4) $p \rightarrow \text{next} \rightarrow \text{next} \rightarrow \text{next} \rightarrow \text{next} = p_{\text{new}}$

(5) $\text{printf}(\underline{\text{first} \rightarrow \text{next} \rightarrow \text{next} \rightarrow \text{next} \rightarrow \text{data}});$

$q \overset{P}{\Rightarrow} C$

~~Main();~~

Struct node $d = \{10, \text{null}\}$.



3

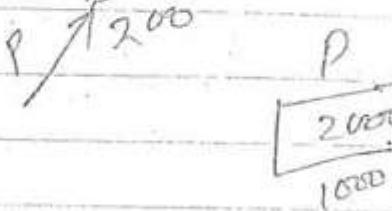
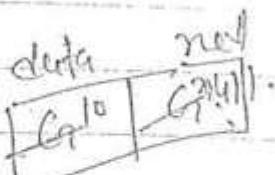
Static <sup>Compile time memory
allocation</sup>

Stacks

Heap

Dynamic memory allocation means Heap Storage

Malloc
memory allocation.



Struct node *P;

= (struct node *) malloc(size of (struct node))

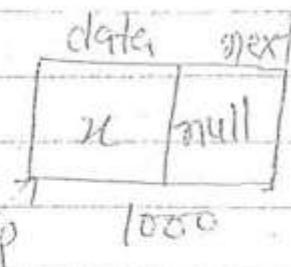
Typecasting

(*P).data or P->data = 10

(*P).next = null or P->next = null

return (P)

3



~~Getnode(u)~~

{
}

Struct node *p;

a



int x
float x
p = q
q = p
Error

int *p
float q
q = p;
Error.

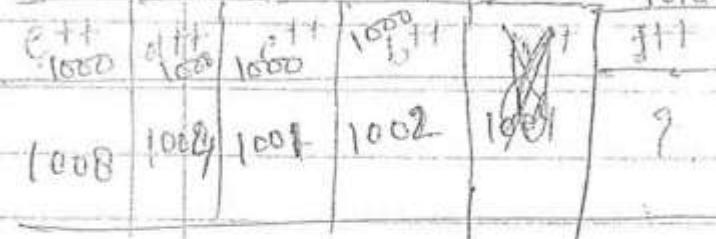
double a = 10.5

double x;
int *b = (int *) &a

char x; c = (char *) &a

float x; d = (float) &a

double float char int



int a = 10
b = (float) a

int x; c = 10
b = (float) &a
c = &a.

a b c
10 1000 1005
1000 1004 1002
b + t = 1004
c + t = 1002

~~XX~~ Getnode (u)

{

Struct node * p;

data next

u null

p 1000

p = (struct node) malloc (sizeof (struct node));

If (p == Null)

{

printf (" memory over")

exit (i);

}

else

{

p → data = u ;

p → next = null ;

return (p)

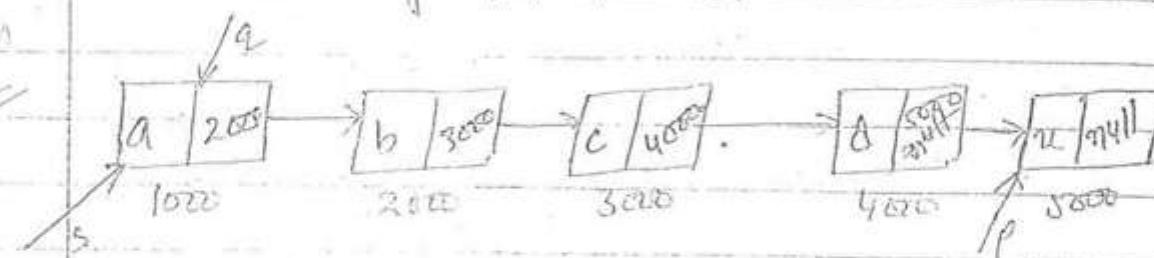
}

{

p
1000
022

data next
u null
1000, 1001, 1002, 1003

~~X~~ WAPP to add a node with data n at the end of linked List:



Struct node Addend LL (struct node *s , int n)

{

Struct node *p, *q;

$q = s;$

$p = (\text{struct node}) \text{ malloc}(\text{sizeof}(\text{struct node}))$

$p \rightarrow \text{data} = n;$

$p \rightarrow \text{next} = \text{null};$

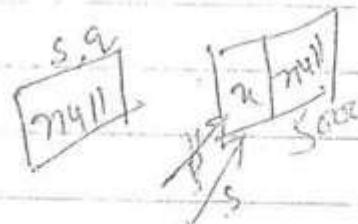
if ($q == \text{Null}$)

{

$s = p;$

return (s)

}



else

{

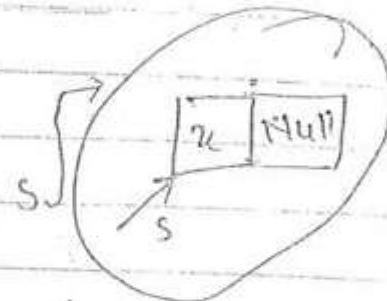
while ($q \rightarrow \text{next} \neq \text{null}$)

$q = q \rightarrow \text{next};$

$q \rightarrow \text{next} = p;$

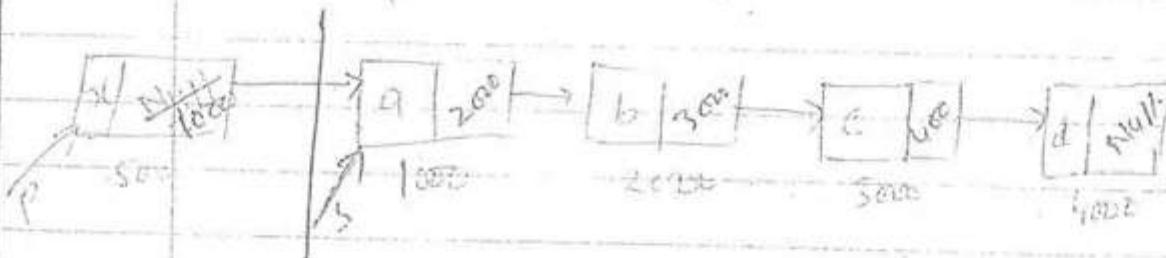
return (s);

3



3

WAP to add a node with data 'u' at the start of linked list.



Add 'u' before (L / struct nodes * s, int u)

{

struct node * p;

p = (struct node *) malloc (sizeof (struct node));

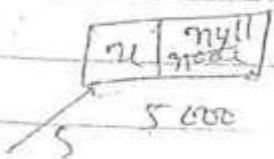
$p \rightarrow \text{data} = u$, $p \rightarrow \text{next} = \text{null}$,

$P \rightarrow \text{next} = s$

$s = P;$

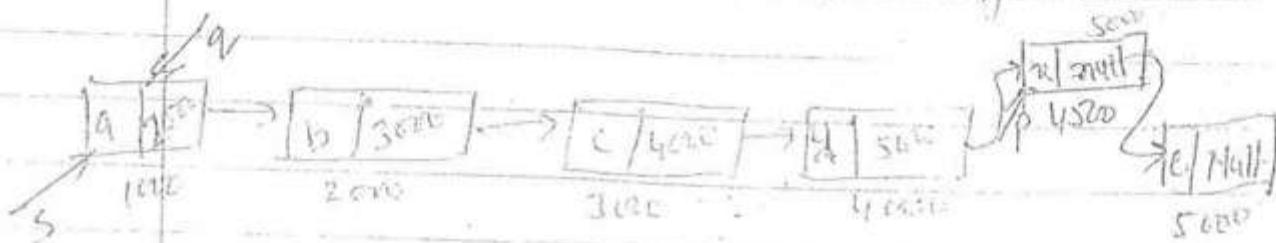
$\text{return}(s);$

}



$s \rightarrow$

WACP to insert a node with data 20 after a node with data 40.



Add y after x ($\text{structnode} s, \text{int} x, \text{int} y$).

struct node * p, * q;

$p = (\text{newnode})(x);$

$q = s;$

if ($s == \text{null}$)

{
Print ("noope L is empty");
exit(1);
}

else

if ($s \rightarrow \text{data} == y$)

{

$p \rightarrow \text{next} = s \rightarrow \text{next}$

$s \rightarrow \text{next} = p;$

}

else

while ($q \rightarrow \text{data} != y \text{ & } q \rightarrow \text{next} != \text{null}$)

$q = q \rightarrow \text{next}$

if ($q \rightarrow \text{data} == y$)

{

$p \rightarrow \text{next} = q \rightarrow \text{next}$

$q \rightarrow \text{next} = p;$

}

else

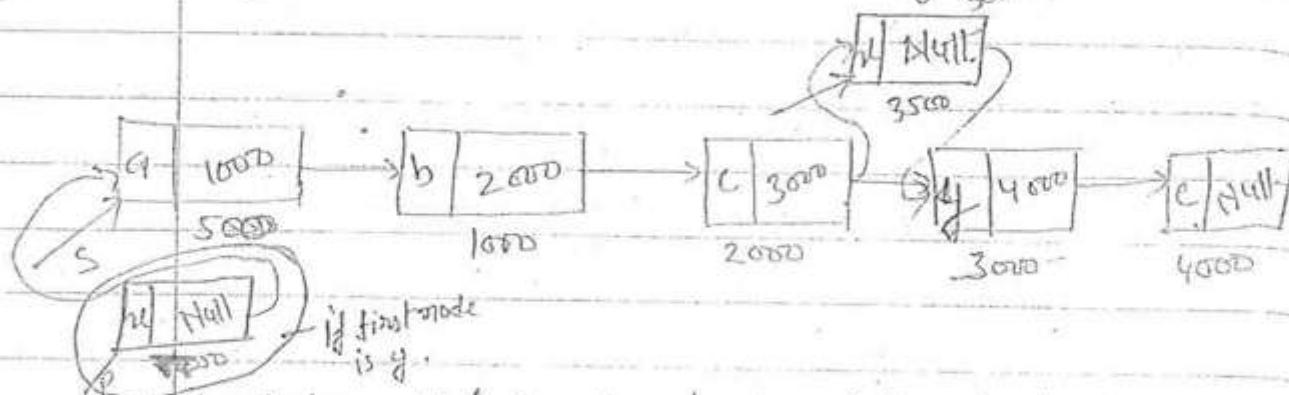
{

printf ("y not present")

} exit(1);

}

WACP to add a node with data x before a node with data y.



Add u before LL (struct node *s , int u , int y)

{
struct node *p , *q , *r ;

p = get node(u);

q = s ; r = null ;

if (s == null)

{
printf (" LL is empty ")

exit(1);

}

else

{

if (s -> data == y)

{

p -> next = s ;

s = p ;

}

else

{

\Rightarrow

while($q \rightarrow \text{data} \neq q \& q \rightarrow \text{next} \neq \text{null}$)

{

$r = q;$

$q = q \rightarrow \text{next};$

}

with 2 pointers

if ($q \rightarrow \text{data} == q$)

{

$r \rightarrow \text{next} = p;$

$p = \text{next} = q;$

}

else

{

printf (" Not Possible ");

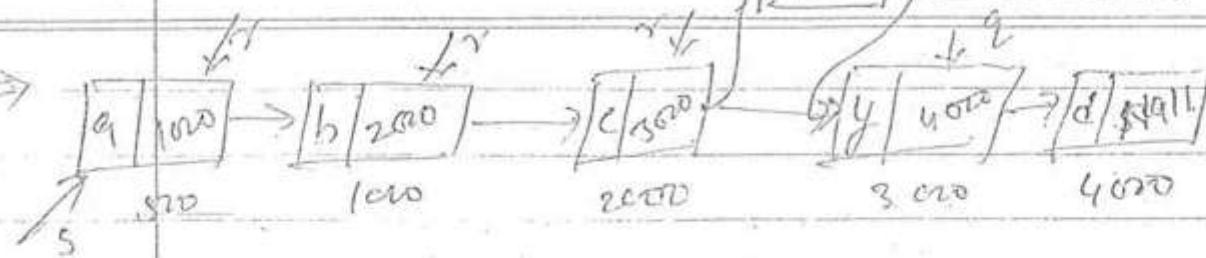
}

}

}

[More than one node]

Page No. _____
Date: _____



$q = s;$

while ($q \rightarrow \text{next} \neq \text{Null}$ && $q \rightarrow \text{next} \rightarrow \text{data}, i = y$)

(1)

$q = q \rightarrow \text{next};$

if ($q \rightarrow \text{next} = \text{Null}$)

{
 printf("y is not available"); using single
 exit(1);
}

(2)

else

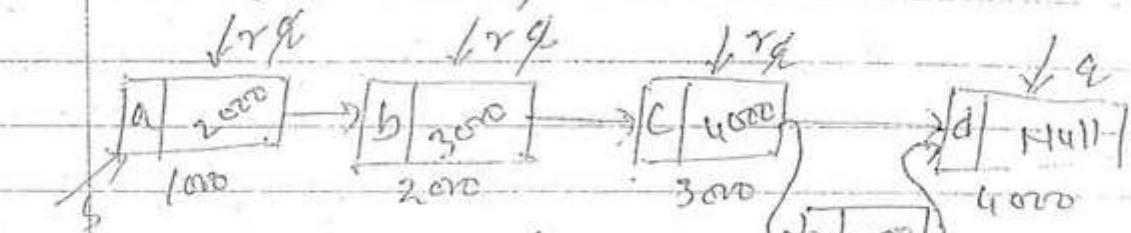
{

$p \rightarrow \text{next} = q \rightarrow \text{next};$

$q \rightarrow \text{next} = p;$

}

WACP to add a node with data 'x' before
the last node;



① $q = s, r = \text{null};$
 if ($s == \text{null}$)

{
 Print("not possible");
 exit(1);
}

② if ($s \rightarrow \text{next} == \text{null}$)

{
 $p \rightarrow \text{next} = s;$
 $s = p;$
}

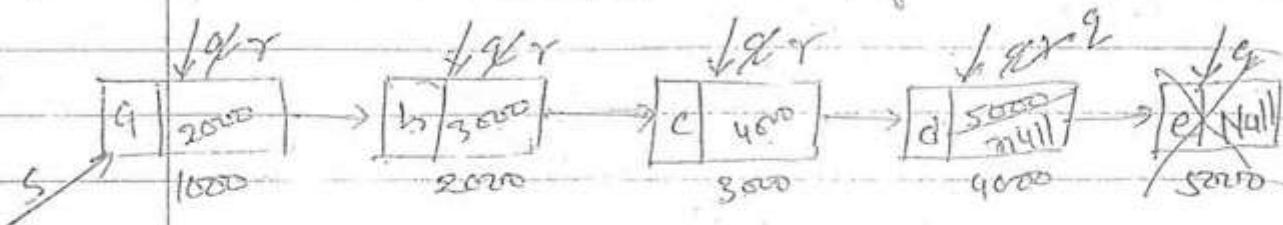
while ($q \rightarrow \text{next} != \text{null}$)

{
 $r = q;$
 $q = q \rightarrow \text{next};$
}

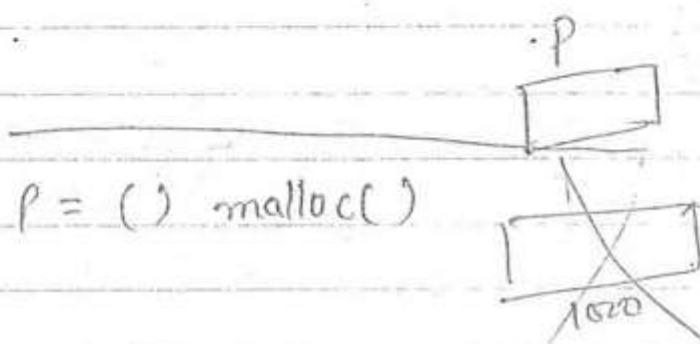
$r \rightarrow \text{next} = p;$
 $p \rightarrow \text{next} = q;$

(node)

WACP to delete last element of Linked list.

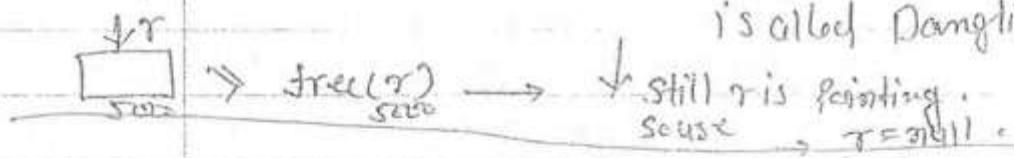


Struct node * p



* no one can point where no memory.

is called Damaging Pointer



Delete node LL(struct node * s)

{

struct node * q;

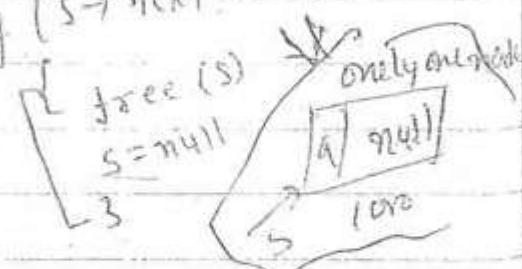
q = s

① empty :

② while(q->next != null)

{
 r = q;
 q = q->next

if (s->next == null)



more than

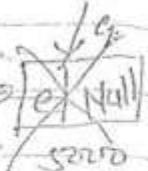
list:

$x \rightarrow \text{next} = \text{null};$

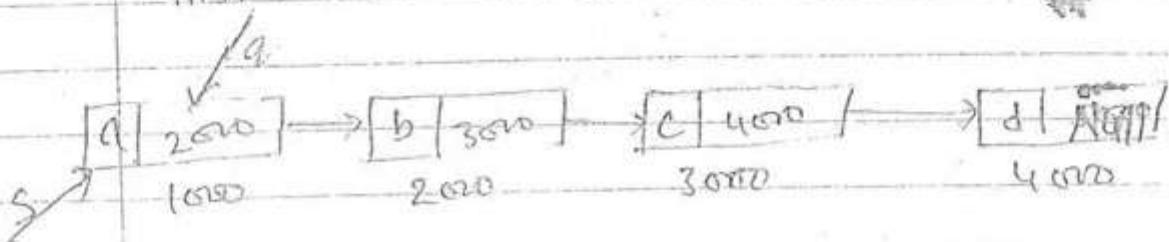
$\text{free}(q);$

$q = \text{null};$

↓ node.



Q WAP to delete a node from start of linked list.



`deletestart (struct node * s)`

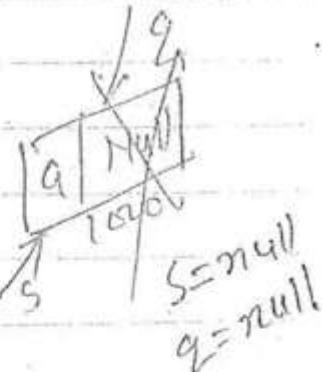
{
① empty ↑ t - node:

② $q = s;$

$s = s \rightarrow \text{next};$

$\text{free}(r)$

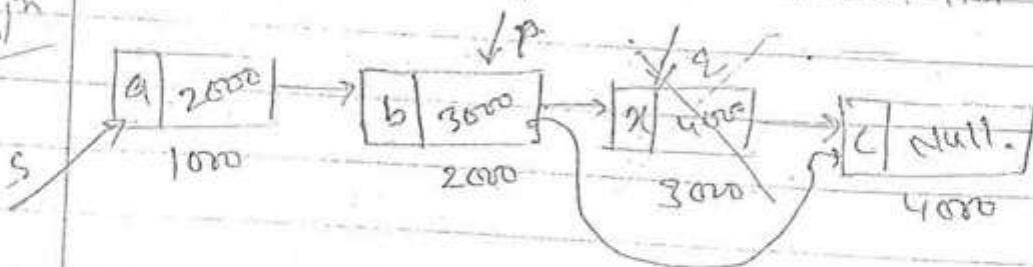
$q = \text{null}$



#

WAP to delete a node which contains data as x from the linked list:

Solⁿ



Delete NULL (struct node *s, int x)

{

struct node *p, *q;

if ($s == \text{NULL}$).

{ empty

}

else

{ if ($s \rightarrow \text{data} == x$)

{

$p = s$

$s = s \rightarrow \text{next};$

free p

$p = \text{NULL};$

}

else

$q = s$

while ($q \rightarrow \text{data} \neq u \& q \rightarrow \text{next} \neq \text{null}$)

{
 $p = q;$

$q = q \rightarrow \text{next};$

}

if ($q \rightarrow \text{data} == u$)

{
 $p \rightarrow \text{next} = q \rightarrow \text{next};$

 free(q); $q = \text{Null};$

}

else

{
 u is not there

}

3

3

3

© Wiki Engineering

www.raghul.org

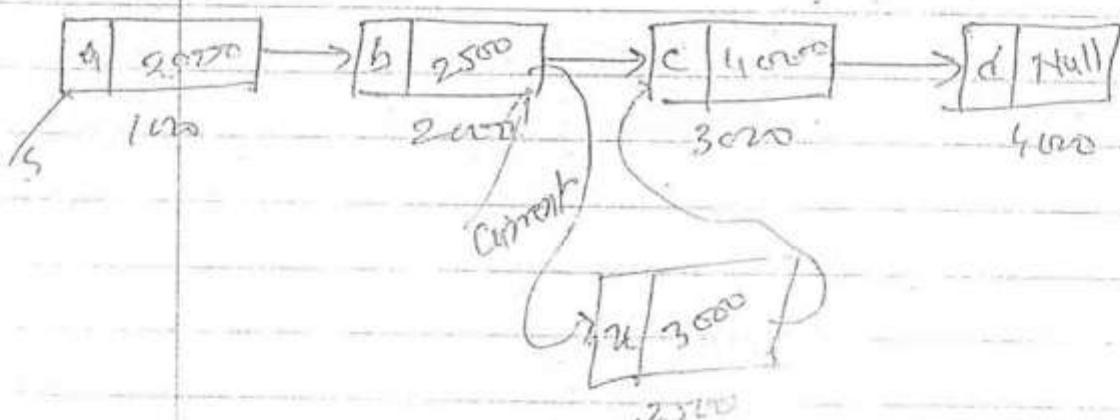
2

Which one of the following statement insert an element u after Position current.

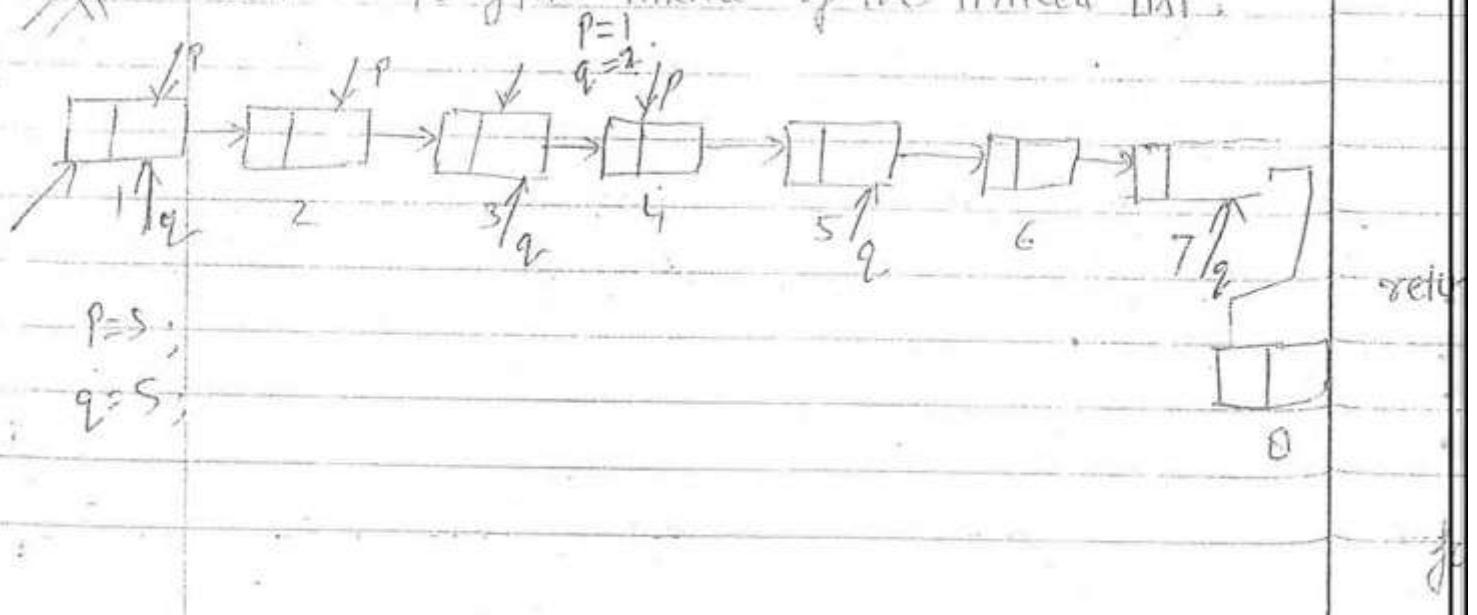
- a) Current = NewNode (u, current)
- b) current = NewNode (u, current->next)

c) Current->Next=NewNode (u, current)

~~d) Current->next = NewNode (u, current->next)~~



WAP to find middle of the linked list.



$p = q = s ;$

while ($q \neq \text{null} \& \& q \rightarrow \text{next} \neq \text{null} \& \& q \rightarrow \text{next} \rightarrow \text{next}$
 $\neq \text{null})$

{

$p = p \rightarrow \text{next} ;$

$q = q \rightarrow \text{next} \rightarrow \text{next} ;$

}

return (p)

* Consider the function defined below.

Struct item

{

int data

struct item * next;

}

int f (struct item * p)

{

return (($p == \text{null}$) || ($p \rightarrow \text{next} == \text{null}$) || ($p \rightarrow \text{data} <= p \rightarrow \text{next} \rightarrow \text{data}$)

|| f ($p \rightarrow \text{next}$))

}

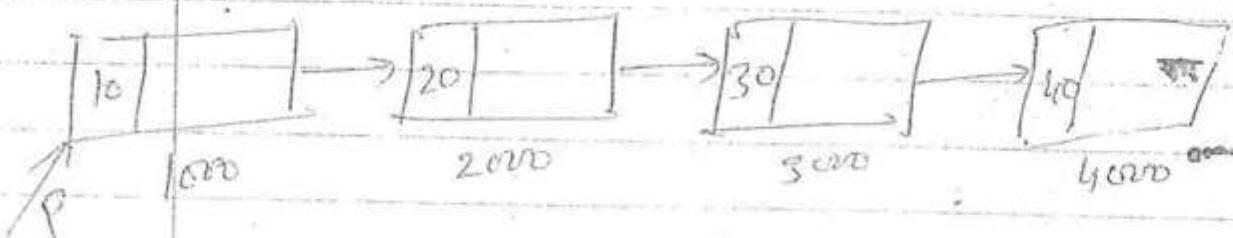
for a given linked list P the above function

f returns iff.

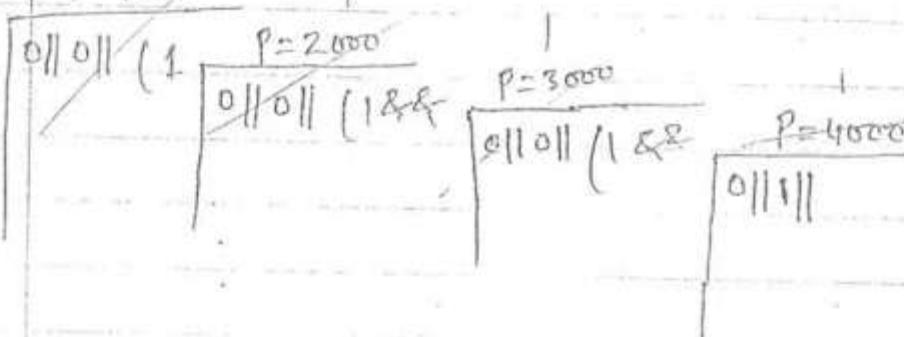
2.

- a) P is empty (or) exactly 1-element.
- b) non-decreasing order.
- c) non-increasing order.
- d) none.

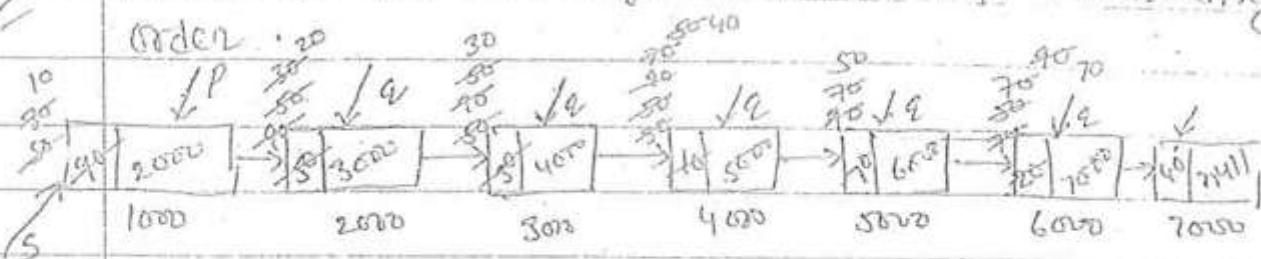
10
96
91
87
11
5



$$I = P = 10000$$



Q) WAP to sort the given linked list in ascending



selection

Sort (struct node * s)

{ struct node * p, * q;

p = s

while (p != null)

{

for (q = p->next, q != null, q = q->q->next)

{

if (p->data > q->data)

swap (p->data, q->data)

p = p->next;

}
}

selection sort

#8

The following C function takes single linked list of integers as a parameter and rearranges the ~~random~~ elements of the list. The "fun" is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution.

SOL

Struct node

Sol

Tr

list

```
{  
    int value;  
    struct node *next;  
};  
void rearrange(struct node *list)
```

#8

```
{  
    struct node *p, *q;  
    int temp;  
    if ((!list) || (!list->next)) return;  
    p = list; q = list->next;  
    while (q)  
    {  
        temp = p->value;  
        p->value = q->value;  
        q->value = temp;  
        p = q->next;  
        q = p->next;  
    }  
}
```

10

Sol

$\text{P} \rightarrow$

a) 1, 2, 3, 4, 5, 6, 7

list 1000

b) 2, 3, 4, 5, 6, 7, 1

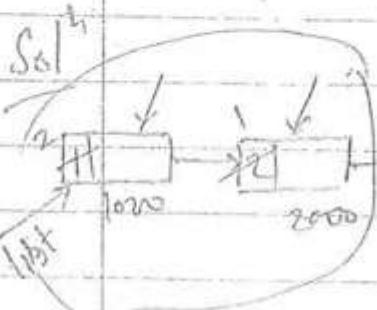
if (list)

if (!list)

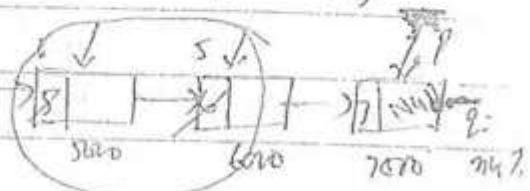
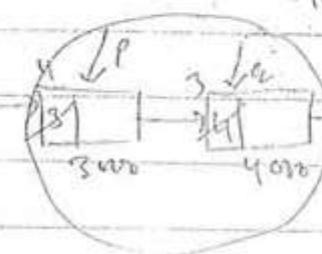
c) 1, 3, 2, 5, 4, 7, 6 if (list != null)

if (list == null)

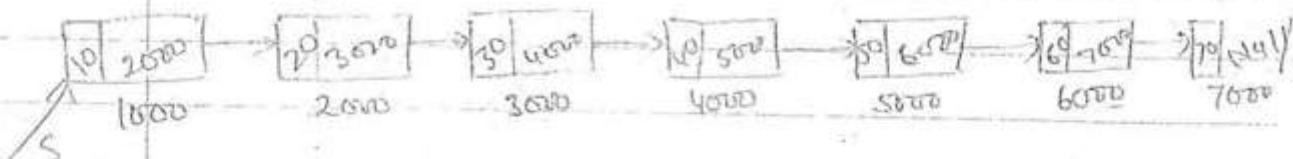
d) 2, 1, 4, 3, 6, 5, 7



#define NULL 0;
#define NO 0;



#P C program to Print the data value of node in given linked list in reverse order.



- Solⁿ
- ① Store in an array and print in reverse order.
 - ② recursion.
 - ③ Reverse the L.L and print one by one.

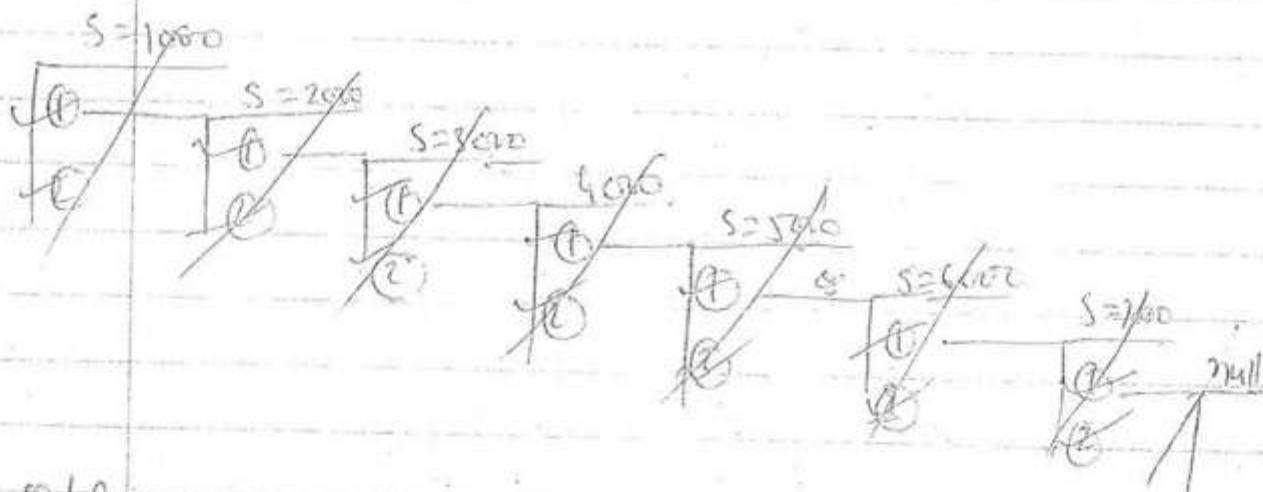
- Recursion

Print LL reverse (struct node *s)

```

{
    if ( s == null )
        return ;
    else
    {
        Print LL Reverse ( s->next )
        Printf ("%d", s->data );
    }
}

```



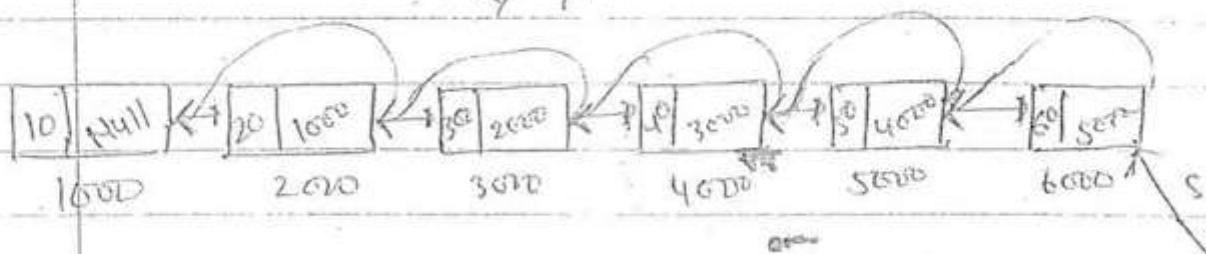
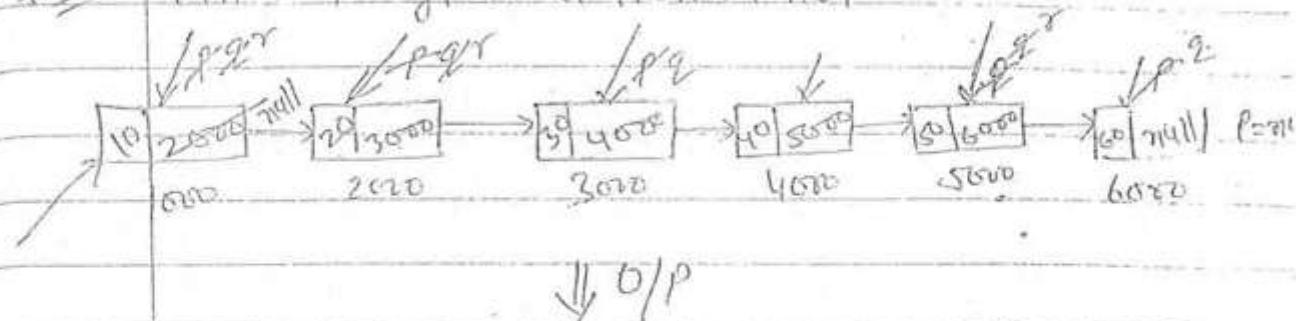
O/P :- 10 20 30 40 50 60 100

10 shows a linked list there pointers are
compulsory → current, previous.
and.

DETA	Page No.	
	Date:	

#8

WACP to↑ given a linked list.



Reverse of LL (struct node *s)

{ Struct node *p, *q, **r ; }

p=s;

q=r=null;

while (p!=null)

T.C $\Rightarrow O(n)$ P.L $\Rightarrow 3 \times 2 = 6$
 $O(1)$

{ r=q

q=p

p=p->next

q->next=r

} S=q

Note:-

DELYA	Page No.	
	Date:	

Reversing the given linked list require giving 3-pointers.

The following funⁿ takes a single linked list as input arguments. It modify the list by moving the last node to the front of the list and return the modified list. Some parts of the function left blank (meanwhile summae gap is given)

Typedef struct node.

```
{  
    int value;  
    struct node *next;  
} node;
```

node * modified list (node * head)

node * p, * q.

```
If ((head == null) || (head->next == null))  
    return (head);
```

q = null, p = head;

while (p->next != Null)

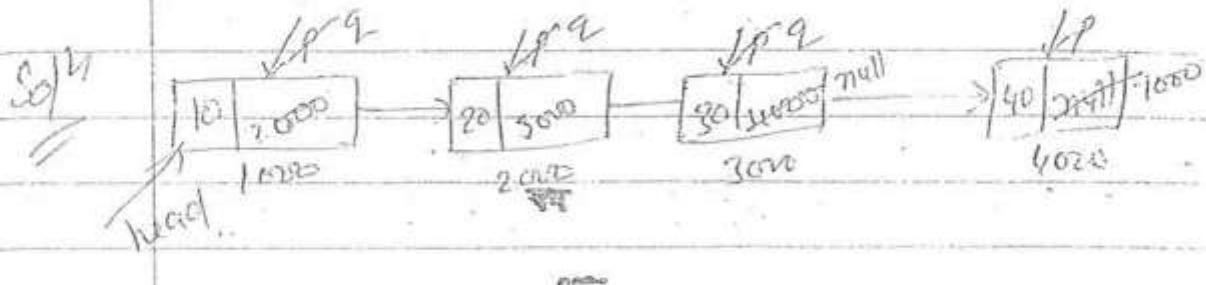
```
{  
    q = p; p = p->next;  
}
```

sim.

$q \Rightarrow \text{next} = \text{null}$; $p \rightarrow \text{next} = \text{head}$; $\text{head} = p$;
 $\text{return}(\text{head})$

clust

by



O/P -

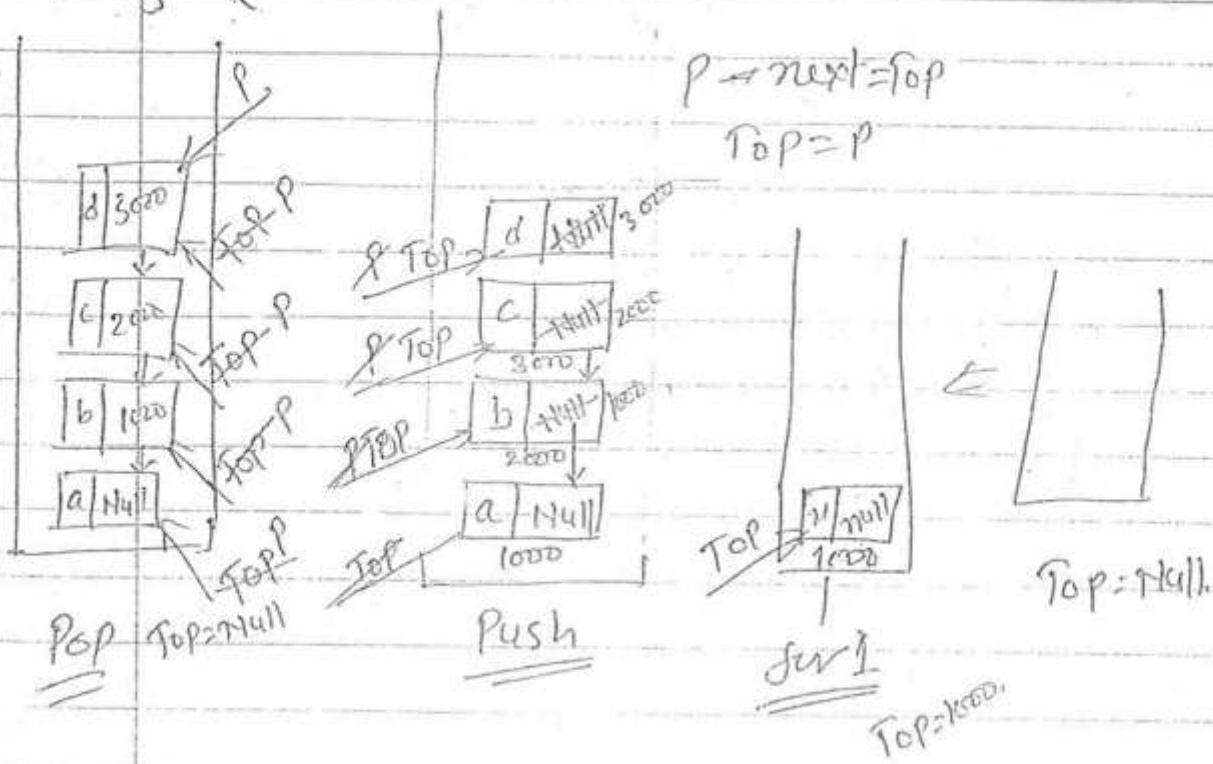
- a) $q = \text{null}$, $p \rightarrow \text{next} = \text{head}$, $\text{head} = p$;
- b) $q \rightarrow \text{next} = \text{null}$; $\text{head} = p$; $p \rightarrow \text{next} = \text{head}$, $\text{head} = p$;
- c) $\text{head} = p$; $p \rightarrow \text{next} = q$; $q \rightarrow \text{next} = \text{null}$;
- d) $q \rightarrow \text{next} = \text{null}$, $p \rightarrow \text{next} = \text{head}$; $\text{head} = p$;

null)

Q

WAP to implement linked stack. (i.e implement push to pop operation) means last in first out .

$P = \text{top}$
 $\text{top} = \text{top} \rightarrow \text{next}$
~~free(p)~~
 $P = \text{null}$



Push(Struct node *top , int u)

{

 Struct node *P;

 P = (struct node *) malloc(sizeof(struct node))

 if (P == NULL)

{

 printf("stack is overflow");

 exit (i);

}

 P->data = u;

 P->next = top;

 top = P;

}

Pop (struct node * Top)

```
{
    struct node * p;
    int y;
```

```
if (Top == Null)
```

```
{}
    cout << "Prints ("stack is undunderflow")";
    exit(1);
}
```

```
{ y = Top -> data;
```

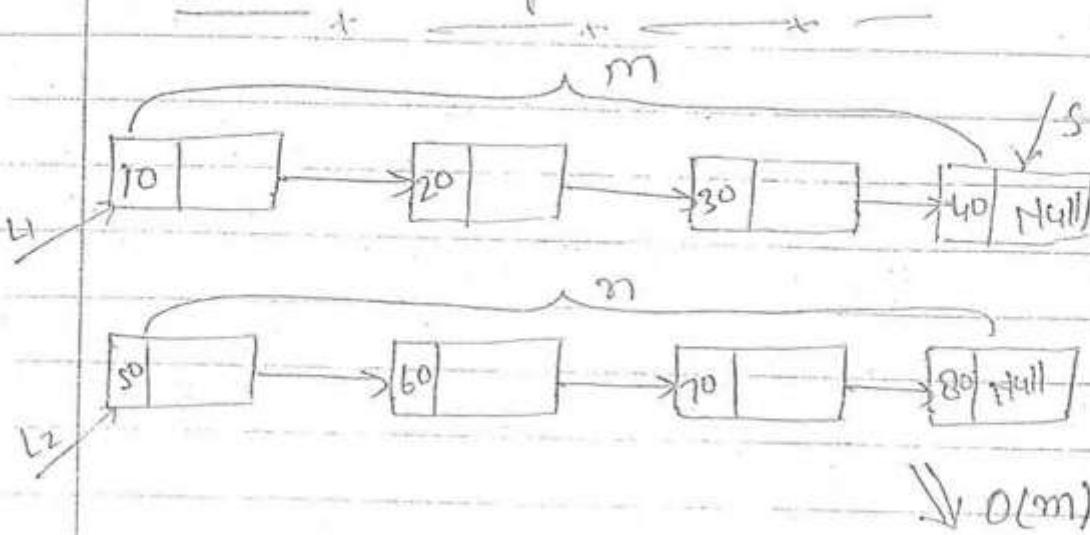
```
    P = top; Top = Top -> next;
```

```
    free(p); p = Null
```

```
return(y)
```

```
}
```

Concatenation of 2-Linked list.



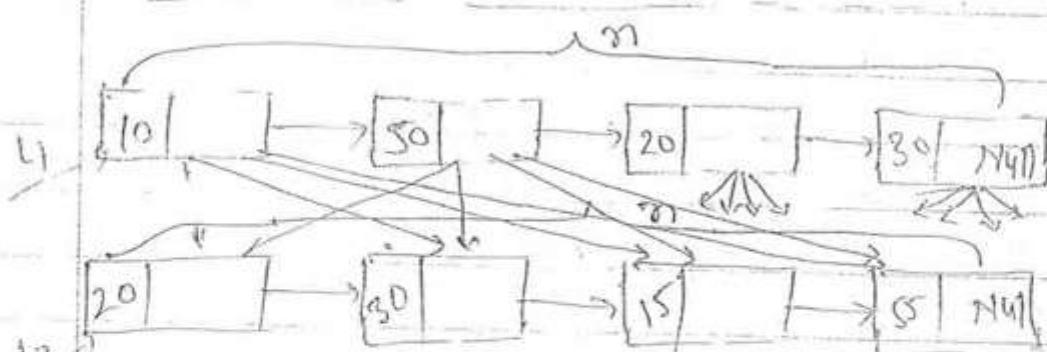
If both L1 and L2
aren't there while / s

While ($s \rightarrow \text{next} \neq \text{null}$) {
 $\quad s = s \rightarrow \text{next}$, } $O(m)$

$$S_{\text{next}} = L_2;$$

```
return(L)
```

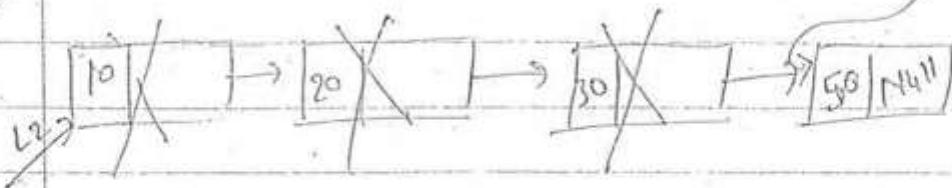
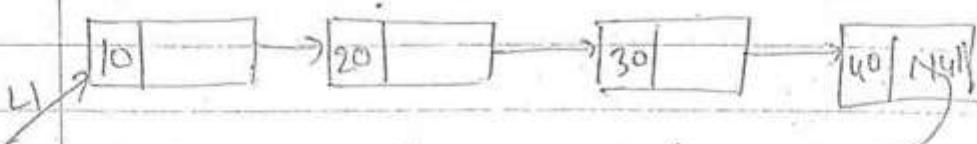
Intersection of 2 - Linked-list



\Rightarrow for every element of 1st LL compare total
 \Rightarrow 2nd list
 $\Rightarrow O(n^2)$

Union of 2 LL

DELT	Page No.	_____
	Date:	_____



Algo

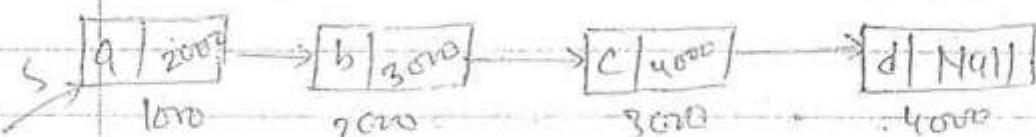
resulted 3rd linked list

- (1) Add 1st LL to the another counter.
- (2) using 1st LL remove common element from 2nd linked list.

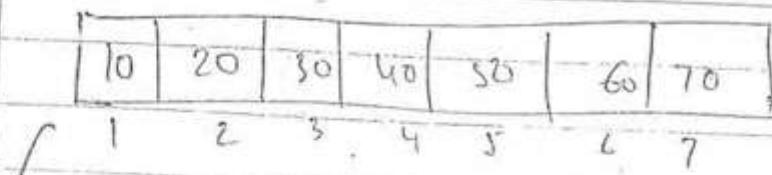
- (3) Add modified 2nd Linked list to 3rd LL.

Time complexity = $O(n^2)$.

Drawback of S. Linked list (linear or single)



- (1) Random access is not possible.



Dis Advantage

\Rightarrow Delete $\Rightarrow O(n)$

Insertion $\Rightarrow O(n)$

go place $\Rightarrow O(1)$

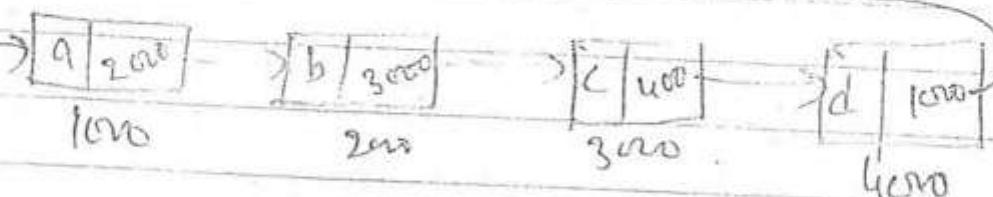
Advantage

Random access.

- (2) From every node we always move only one direction. i.e. we can not go back.
- (3) We are not using efficiently last node next part becoz it is always null.

Circular Single linked List

If we keep 1st node address in last node next part that single linked list is known as circular linked list.



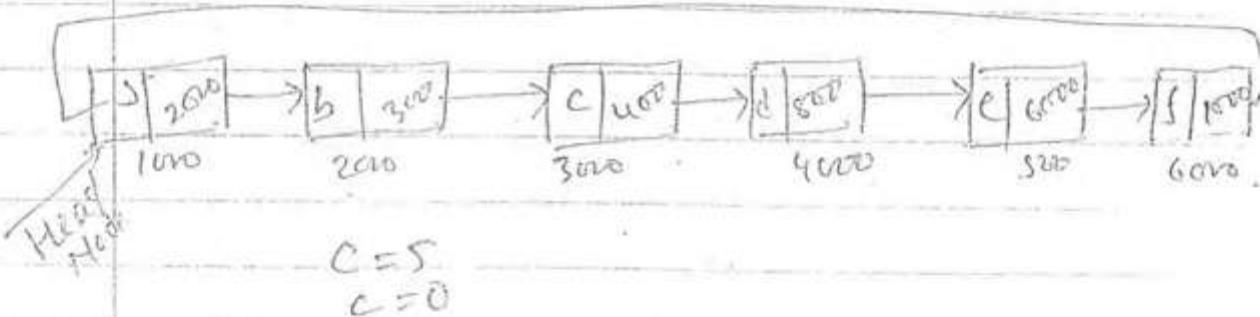
Using CSLL from a given node we can go back also.

Drawback

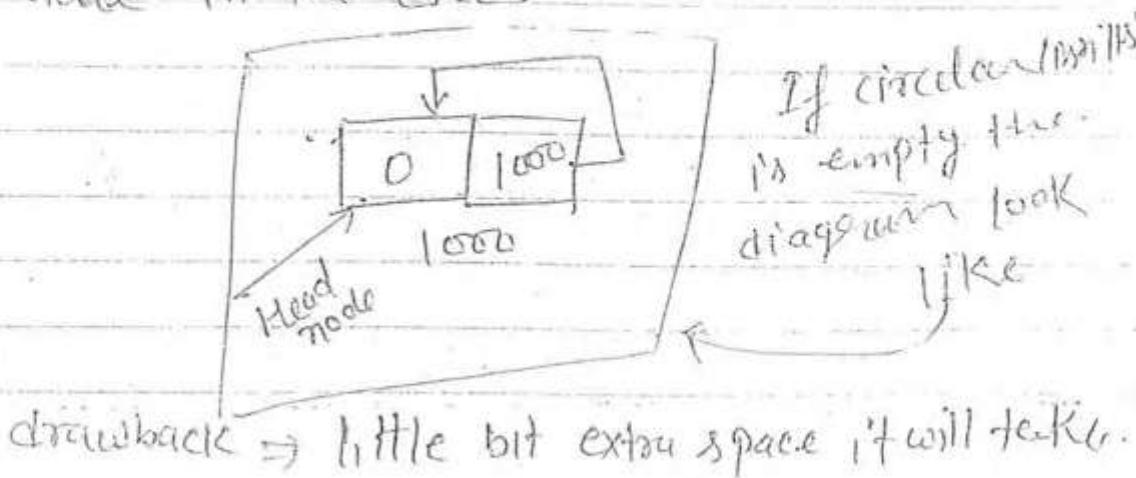
The draw back of CSLL is it will go to infinite loop if we don't write the code properly.

Head Node

Using headnode we can stop infinite loop in circular linked list.

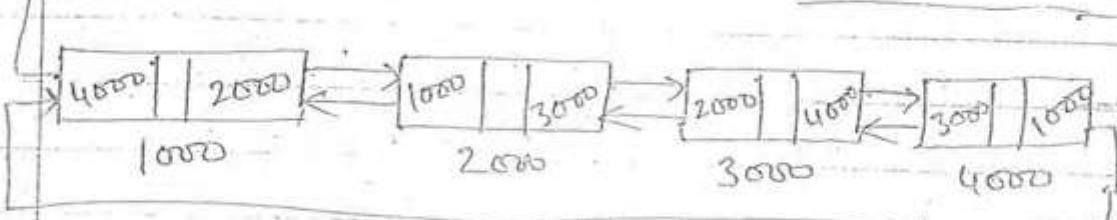


Using Head node data part will contain valuable information like no. of node in the CSLL.



Double linked list

(Double Circular linked list (DCLL))

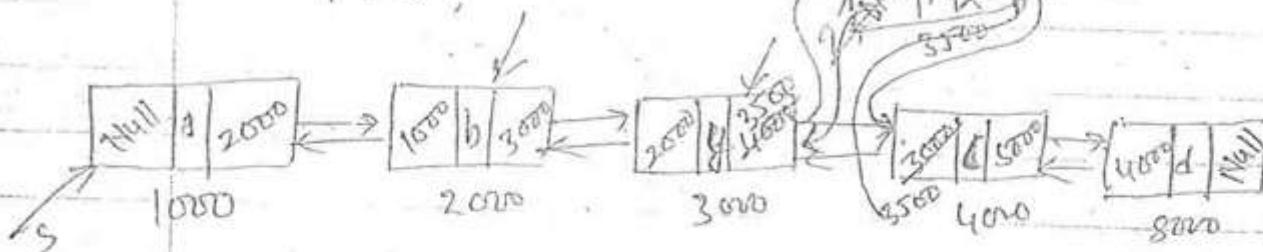


The advantage of DLL is if we lost one pointer we will get it back with the help of another pointer.

P

WAP to insert a node with data 22 after a node with data 44. In Double linked list.

P = S;



P = S;

while (P->data != y && p->next != null)

P = P->next;

if (P->data == y)

{

$q = \text{Get node}(x)$

$q \rightarrow rp = p \rightarrow rp$

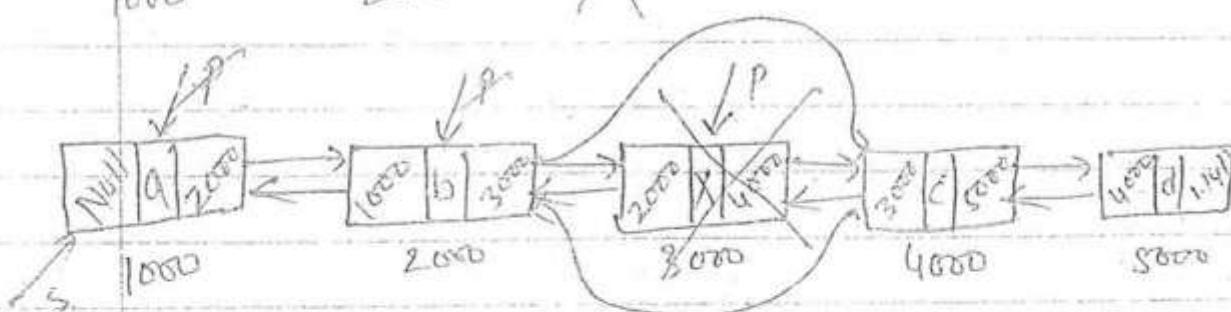
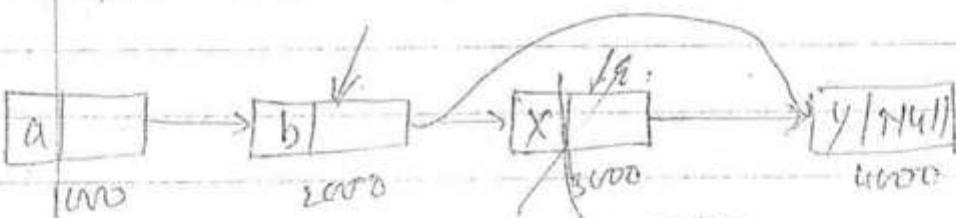
$p \rightarrow rp = q$

$q \rightarrow rp \rightarrow lp = q$

$q \rightarrow lp = p;$

}

#? WACP to delete a node from Double linked list which contain data as x :



$p = s;$

while ($p \rightarrow \text{data} != n \&\& p \rightarrow rp != \text{null}$)

$p = p \rightarrow rp.$

Polynomial Addition (Using Single LL)

$$P_1 = 10u^5 + 4u^3 + 5u^2 + 15u + 100$$

$$P_2 = 20u^5 + 15u^3 + 10u^2 + 30u + 200$$

$$P_3 = 30u^5 + 19u^3 + 15u^2 + 45u + 300$$

Struct Poly

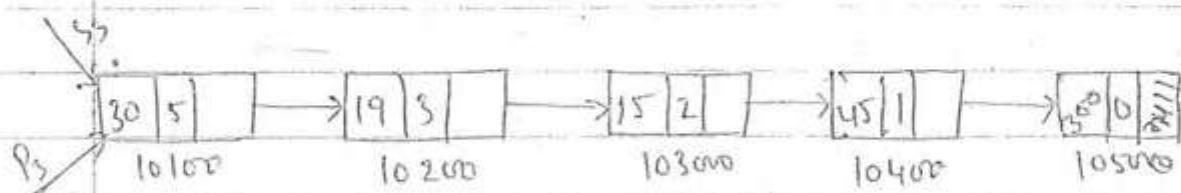
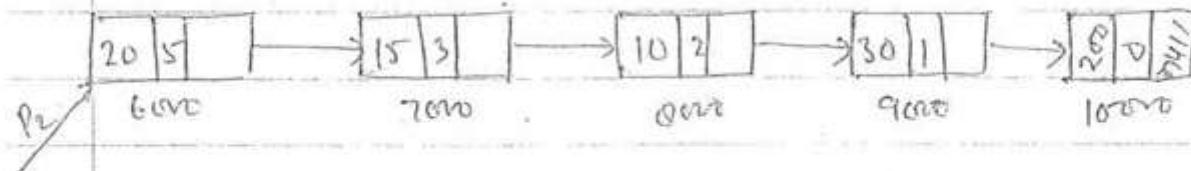
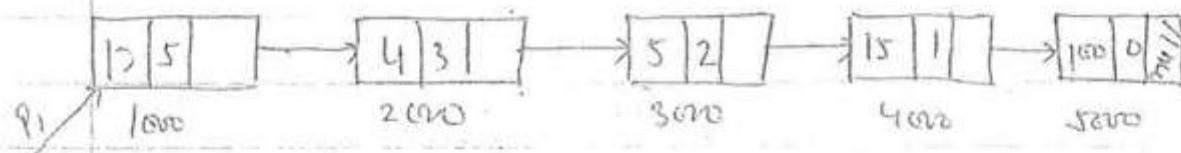
{

int coe;

int pow;

struct Poly *next;

}



Polyaddition (struct Poly *p₁, struct Poly *p₂, struct Poly *p₃)

{ S₃ = p₃; $\Theta(n+m)$
 if m > n
 while (~~p₁ != null & p₂ != null~~) : o(n)

{ if (p₁ → Pow == p₂ → Pow)

{ p₃ → Pow = p₂ → Pow) $\Theta(n)$

 p₃ → Coe = p₁ → Coe + p₂ → Coe;

 p₁ = p₁ → next , p₂ = p₂ → next ;

 p₃ = p₃ → next ;

}

else

{ if (p₁ → Pow > p₂ → Pow),

{ p₃ → Pow = p₁ → Pow;

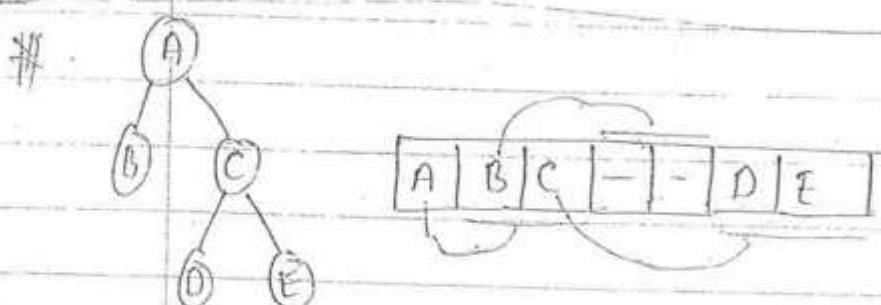
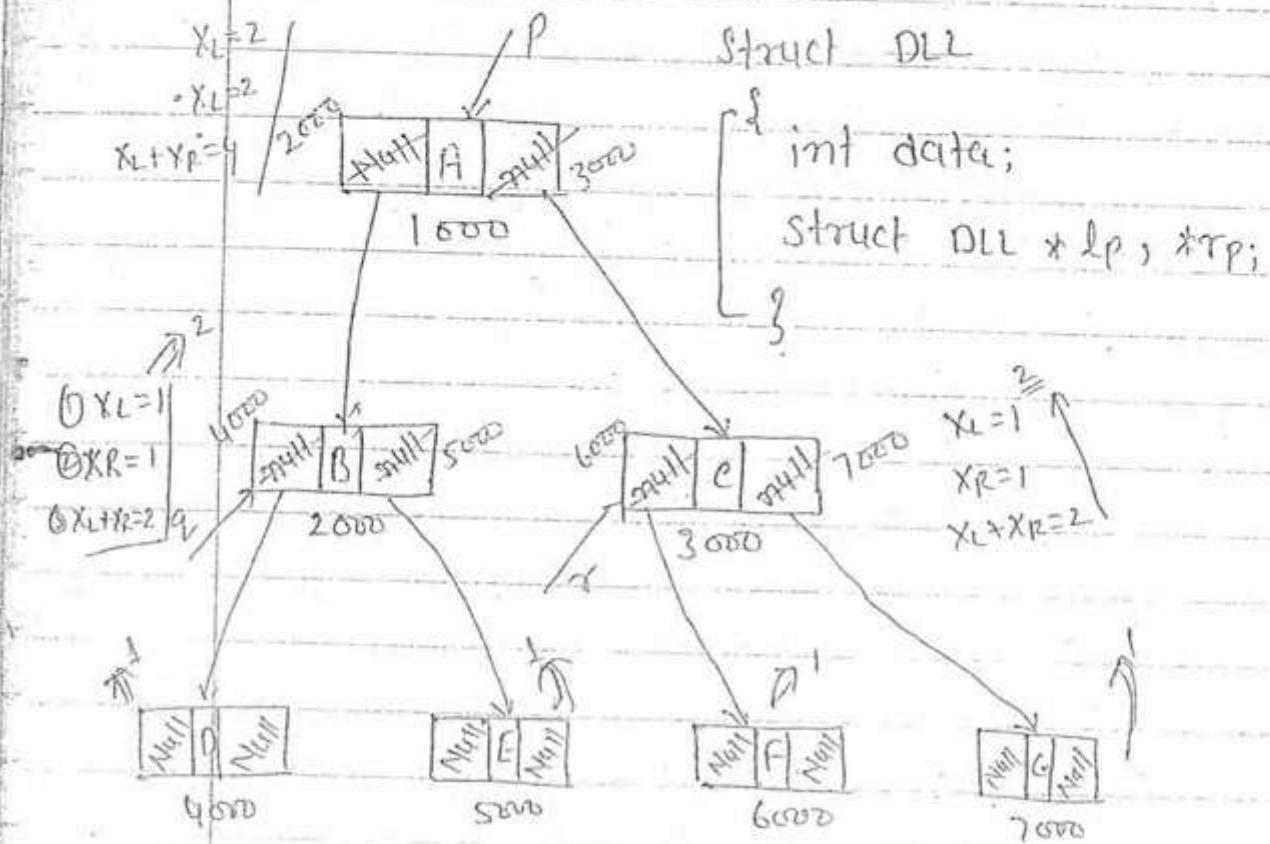
 p₃ → Coe = p₁ → Coe;

 p₁ = p₁ → next ;

 p₃ = p₃ → next ;

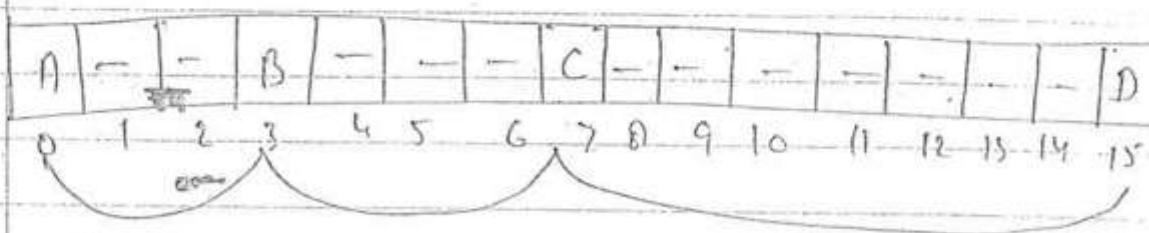
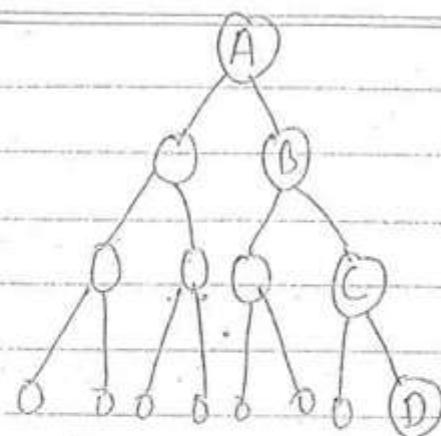
}

Binary Tree



linked list is better approach for Binary tree not a array

#



WACP to find no of leaf nodes in the given binary tree.

Numleafnodes(struct BtNode *p)

{ if (p == null)

return 0;

else

{ if (p->lp == null and p->rp == null)

return 1;

array

Else

{

$$X_L = \text{Numleafnode} (P \rightarrow L_P)$$

$$X_R = \text{Numleafnode} (P \rightarrow R_P)$$

$$\text{return}(X_L + X_R)$$

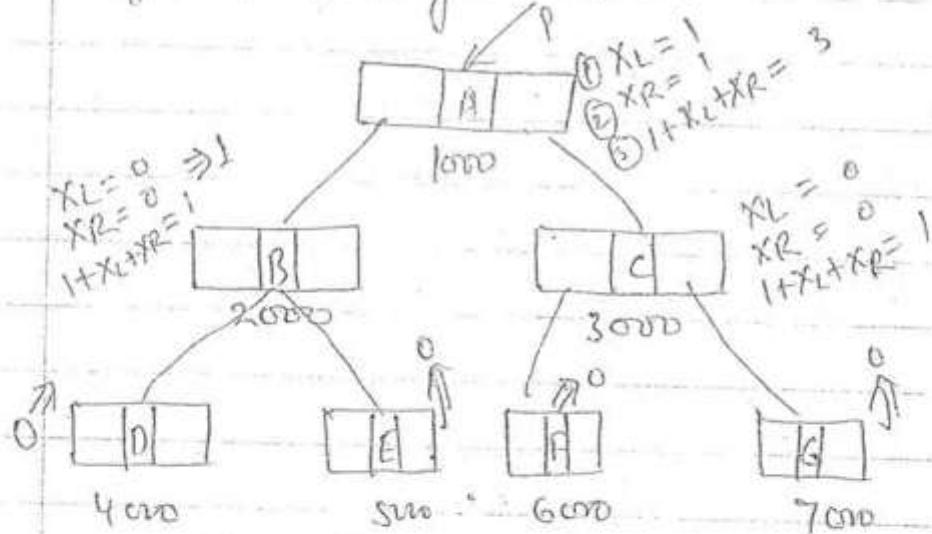
{

{

{



WACP to find no of internal nodes in the given Binary tree ..



leaf node means internal node is zero.

Numinternalnodes (struct BtNode *p)

{
if ($p == \text{Null}$)

return (0);

else

{
if ($p \rightarrow l_p == \text{Null} \& p \rightarrow r_p == \text{Null}$)

return (0);

else

{

$X_L = \text{Numinternalnodes}(p \rightarrow l_p)$

$X_R = \text{Numinternalnodes}(p \rightarrow r_p)$

return ($1 + X_L + X_R$)

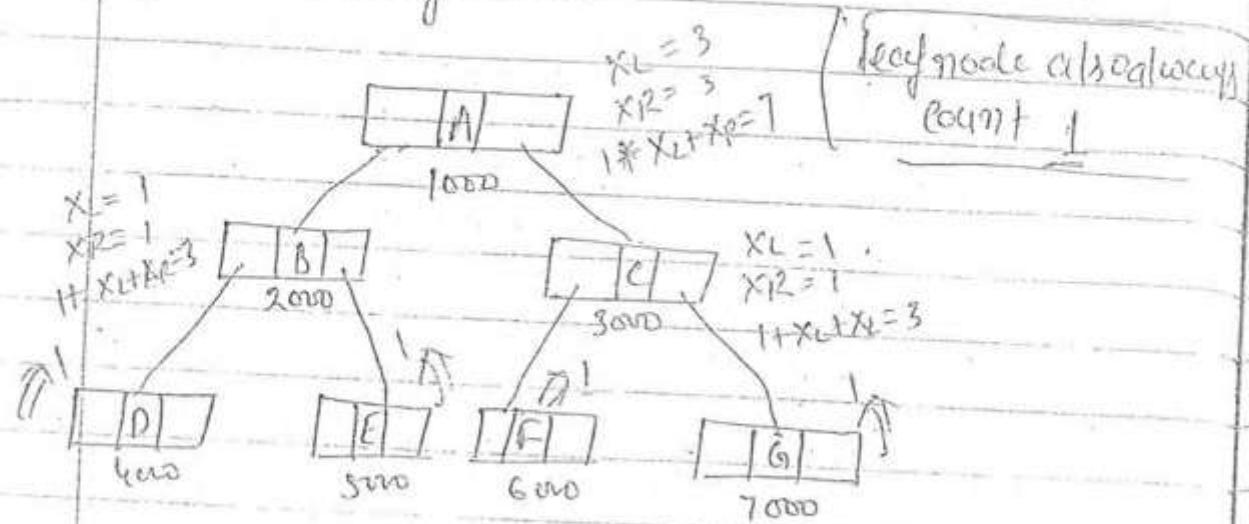
3

3

3

2

WACP to count total no of nodes in the given binary tree.

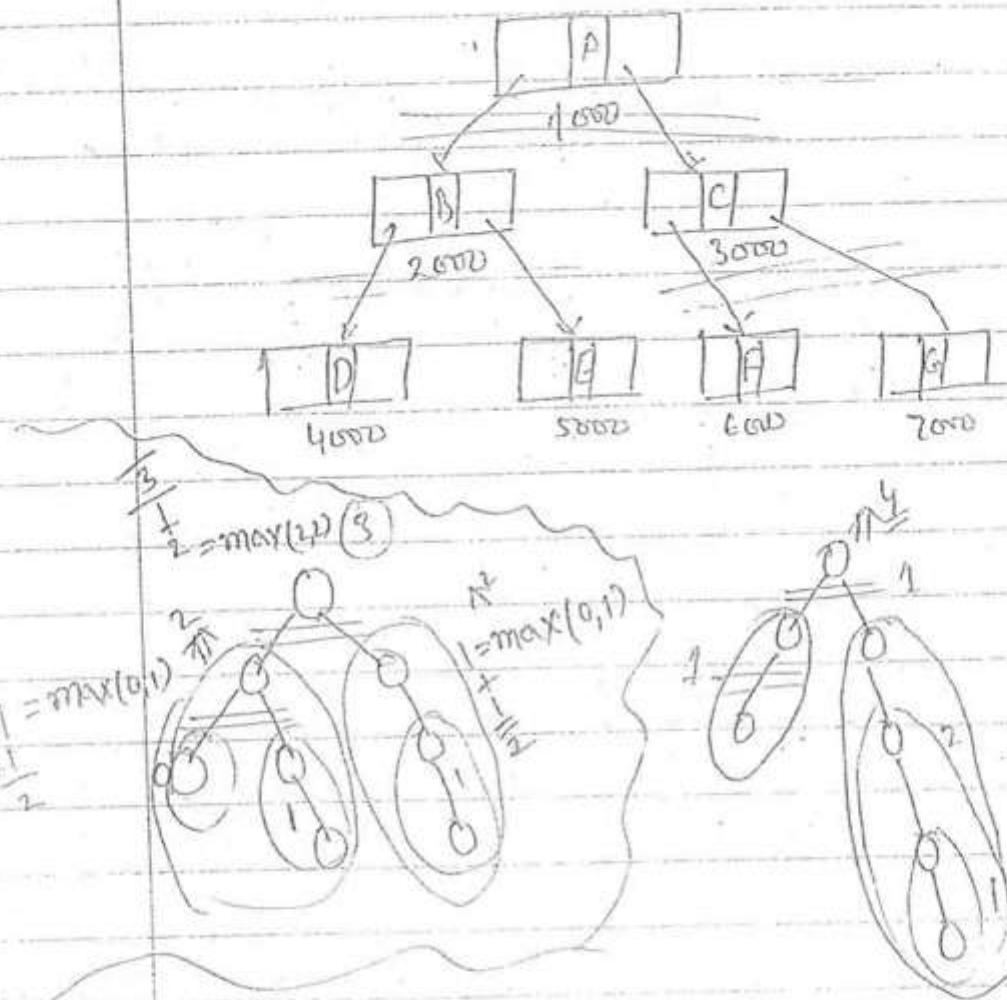


Total no of nodes(struct. BtNode *p)

```

    {
        if ( p == NULL )
            return 0;
        else
        {
            if ( p->lp == NULL && p->rp == NULL )
                return (1);
            else
            {
                X_L = total no of nodes ( p->lp );
                X_R = total no of nodes ( p->rp );
                return ( 1 + X_L + X_R );
            }
        }
    }
  
```

Ques. # 2 WACP to find height of a given binary tree.



Height of BT (struct BtNode *)

```

if ( p == NULL )
    return 0;
else
    if ( p->l == NULL & p->r == NULL )
        return 0;
  
```

else

{

$x_L = \text{Height of BT} (P \rightarrow l_p)$

$x_R = \text{Height of BT} (P \rightarrow r_p)$

if ($x_L < x_R$) return ($x_R + 1$)

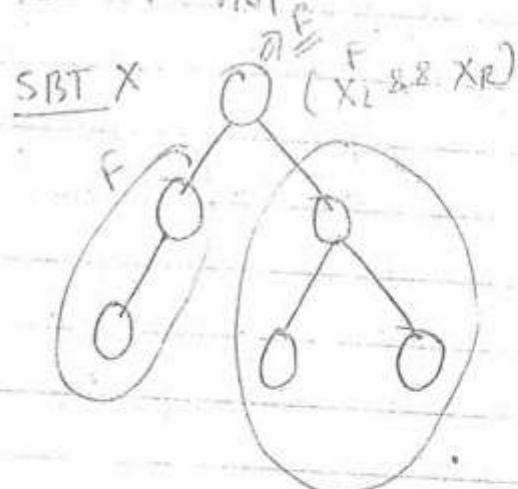
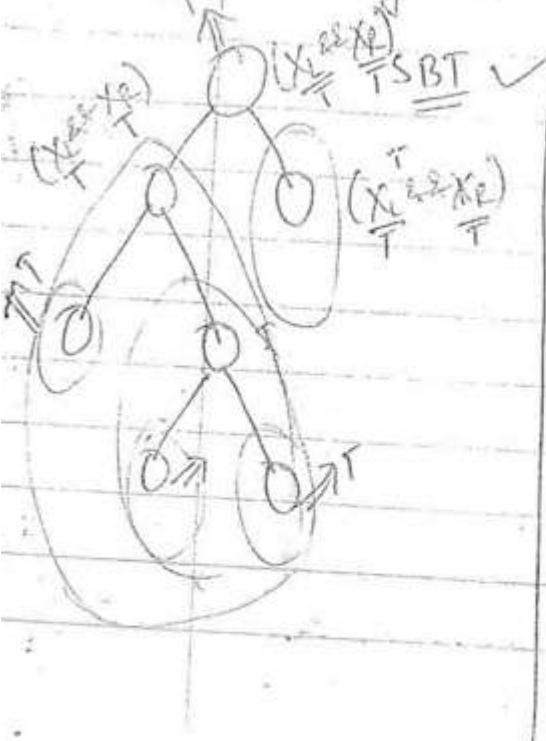
else

return ($x_L + 1$)

}
3
3
3

Q

WACP to check given binary tree is strict
binary tree is or not



Strict binarytree(Struct BtNode *p)

{

if (p == Null)

return (True);

else

{

if (p->lp == Null && p->rp == Null)

return (True)

else

{

if (p->lp != null && p->rp != null)

return (Strictbinarytree(p->lp) && Strictbinarytree(p->rp))

else

return (False)

}

}

2 If we apply the code below on B.S.T. (Q), it will return -?

int SomeValue (struct node *node)

{

 struct node *current = node;

 while (current → left != null)

 current = current → left;

}

return (current → data);

Sol:

B.S.T.

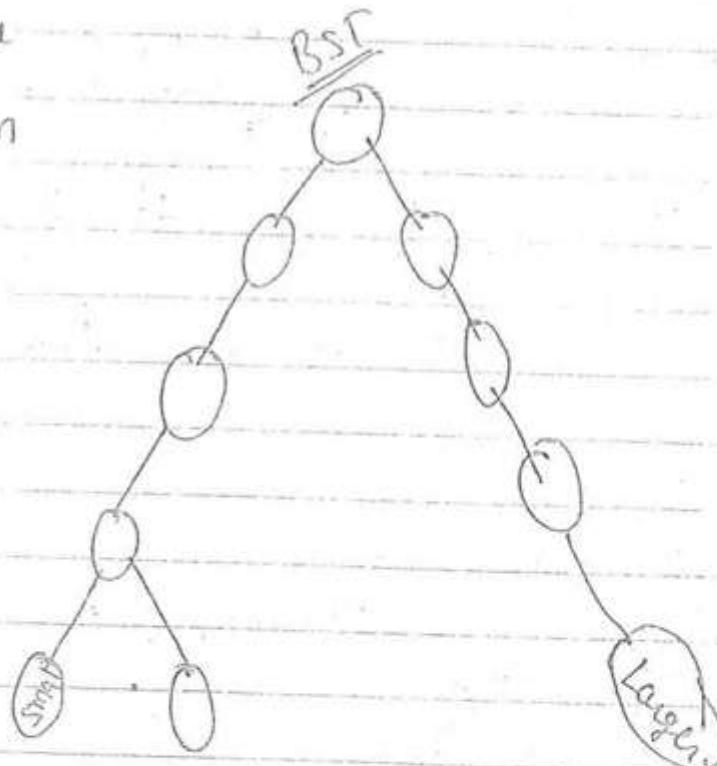
↳

left most value

→ minimum

T.C $\geq \Theta(n)$

wl or right worst case $= \Omega(n)$

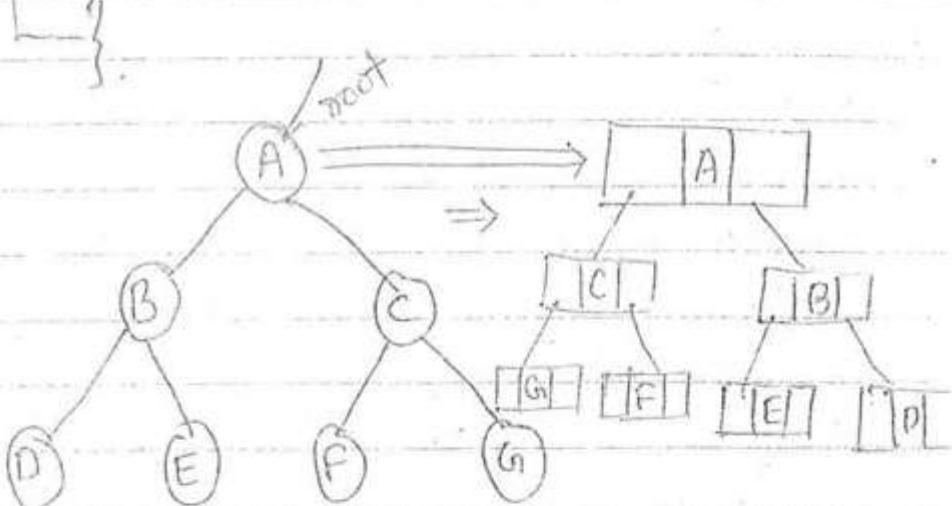


~~bad :)~~~~XX?~~

~~MyNode* copy(MyNode* root)~~

```

    {
        MyNode* temp;
        if (root == null) return null;
        temp = (MyNode*) malloc(sizeof(MyNode));
        temp->value = root->value;
        temp->left = copy(root->right);
        temp->right = copy(root->left);
        return temp;
    }
  
```



~~above code will create mirror image for given binary tree.~~

Raghul

8 A ds is comprised of nodes each of which has exactly 2-pointers to other nodes with no null pointers. The following c program is to be used to count the no of nodes in the given B.B. It uses a mark field assumed to be initially zero for all nodes. There is a statement missing for that code find it.

$$\textcircled{1} \quad X_L = 0$$

$$\textcircled{2} \quad X_R = 3$$

$$\textcircled{3} \quad I + X_L + X_R$$

Struct Test

```

struct test
{
    int info, mark;
    struct test *p, *q;
};

```

```
int node_count ( struct test *a )
```

```

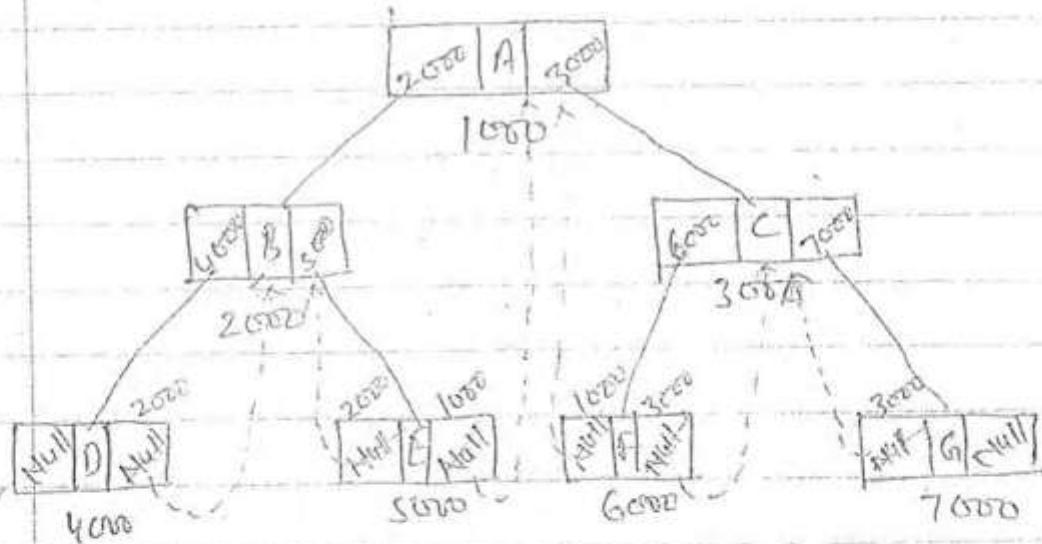
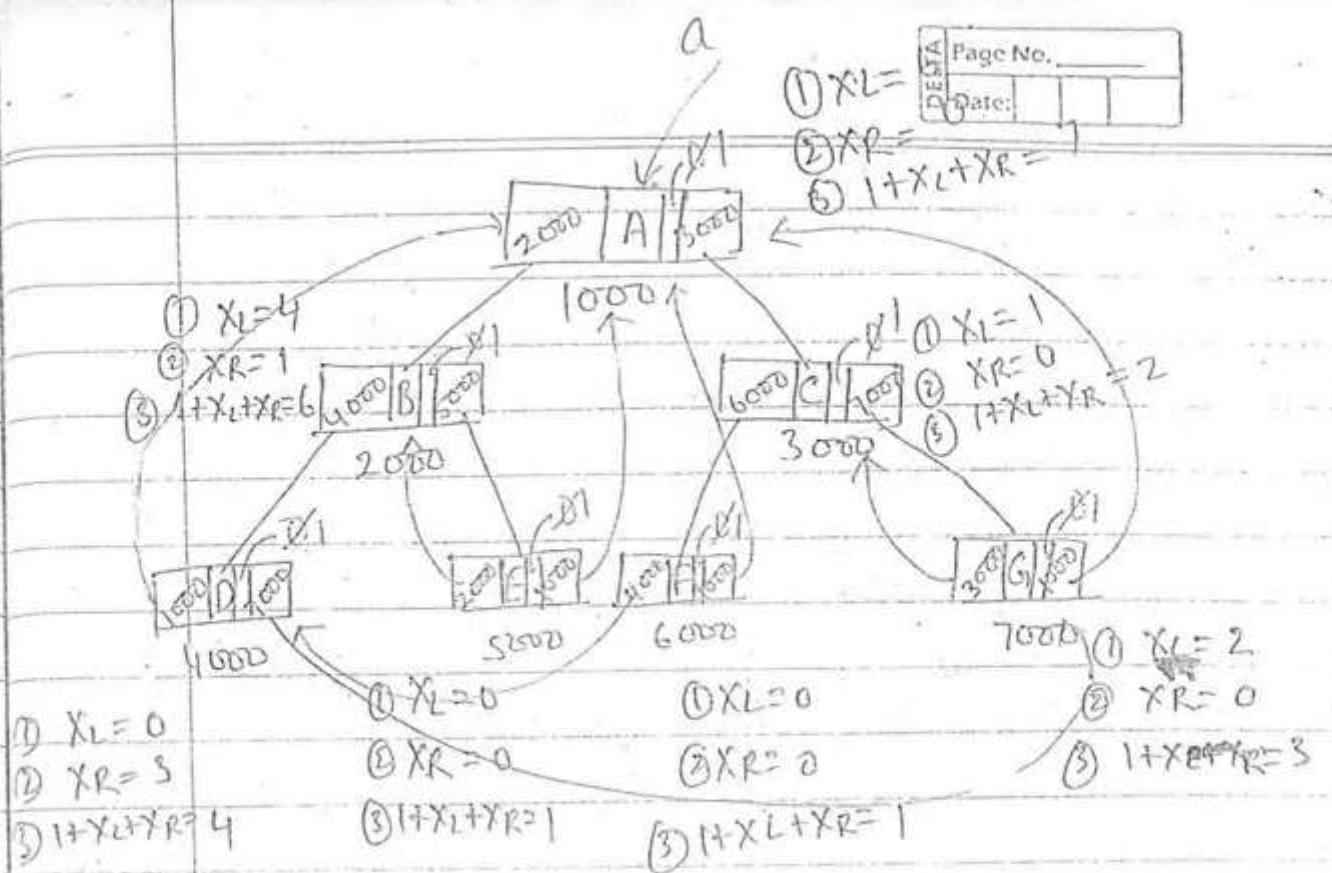
{
    if ( a->mark ) return ( 0 );
}
```

```
else
```

```

{
    a->mark = 1
    return ( node_count ( a->p ) + node 
```

```
count ( a->q ) + 1 );
```



Drawback of leaf nodes: left pointer and right pointer always null.
 We cannot use that space efficiently.

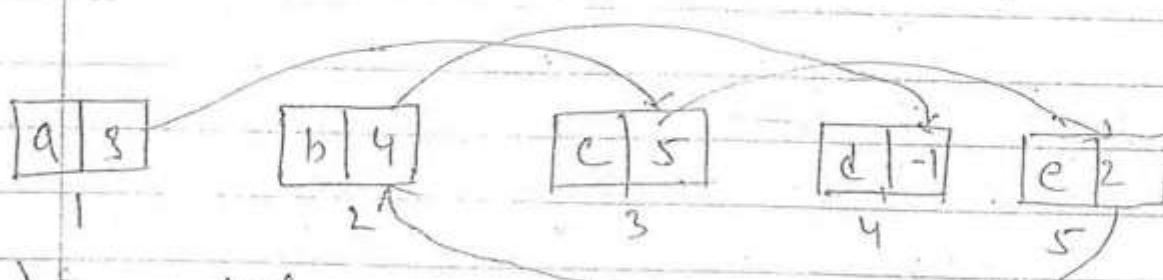
Inorder: - D B E A F C G

Inorder Threaded Binary tree.

Inorder Successor \Rightarrow Right link }
 Inorder predecessor \Rightarrow left link } leaf nodes.

How Implementing L.L. Using Array ?

data		link	
1	a	1	3
2	b	2	4
3	c	3	5
4	d	4	-1
5	e	5	2



Student's address is 1
 temp = 2
 a = 1
 b = 2
 c = 3
 d = 4
 e = 5

#2 WACP to print data of every node in the given linked list using array representation.

display(data, link, s)

{
int temp;

temp = s;

while (temp != -1)

O/P:- acebd

{
 printf("%d", data[temp])

 temp = link[temp]

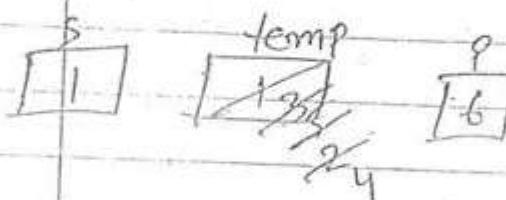
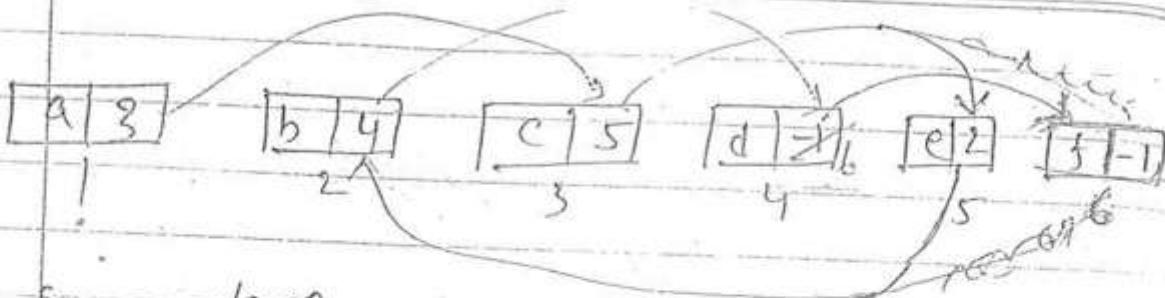
}

O/P:

WACP to add a node with data X, in the given linked list using array representation at the end.

data	
1	9
2	b
3	c
4	d
5	e
6	f

link	
1	3
2	4
3	5
4	-1
5	2
6	-1



add x at end (data, link, s, p)

```
{ int temp;
```

```
temp s;
```

```
}
```

```
while (link[temp] != -1)
```

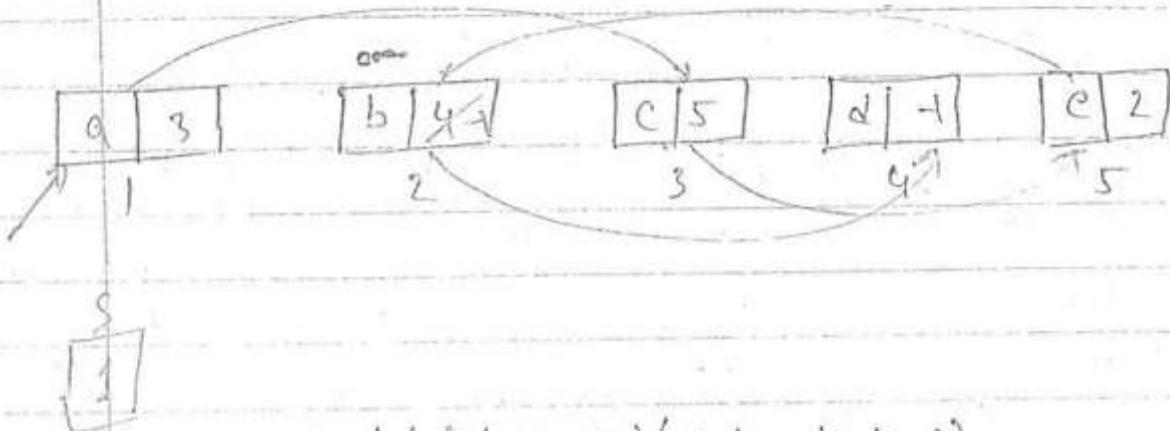
```
temp = Link[temp]
```

```
Link[temp] = p;
```

```
}
```

WAP to delete a node from end of the linked list in array representation.

data		link	
1	a	1	3
2	b	2	4
3	c	3	5
4	d	4	-1
5	e	5	2



deletefromend(data, link, s) :

```

    {
        int t1, t2, t;
        if (s == -1)
            emptyL();
    }

```

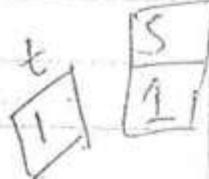
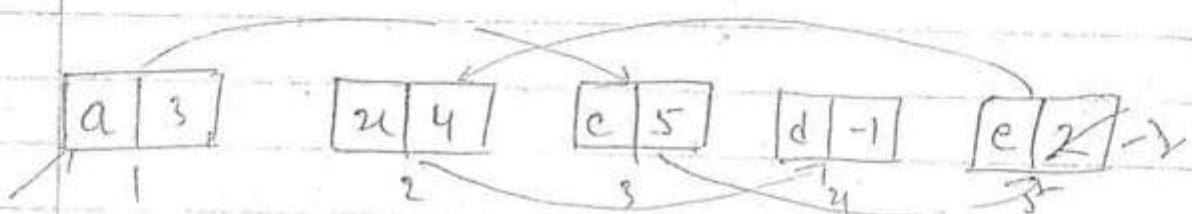
$\text{if } (\text{Link}[s] == -1)$] 0-element
 $s = -1$] 1-element

while ($\text{Link}[t_1] \neq -1$)

more than
one element {
 t2 = t1;
 t1 = link(t1);
 link[t2] = -1;

WACP to delete a node which contain data as x in the given linked list in the array representation.

data		link	
1	a	1	3
2	b	2	4
3	c	3	5
4	d	4	-1
5	e	5	2



deletefromend (data , link , s)

```
{ int t1,t2;
    t1=s ;
    if ( s == -1 )
        empty;
    if ( data[s] == x )
```

```
    {
        t=s;
        s=link[s];
        link[t]=-1
    }
```

in the

else

{ while ($|link[t_1]| = u \text{ & } link[t_1] \neq -1$)

{ $t_2 = t_1$

$t_1 = link(t_1)$;

}

if ($data[t_1] = u$)

{ $link[t_2] = -1$ $link[t_1]$

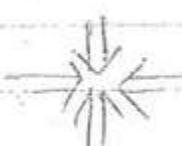
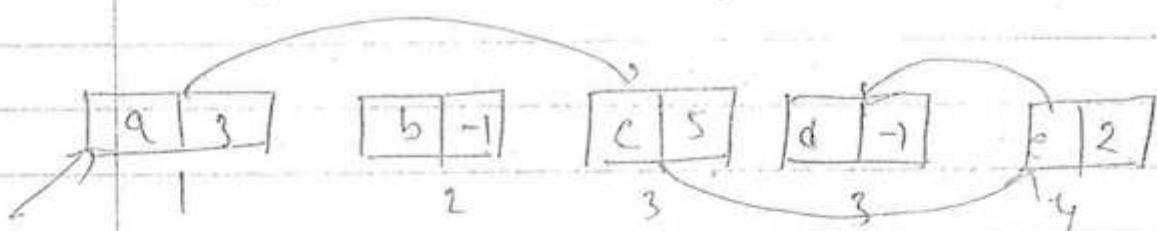
$link[t_1] = -1$

}

else

u is not there.

}

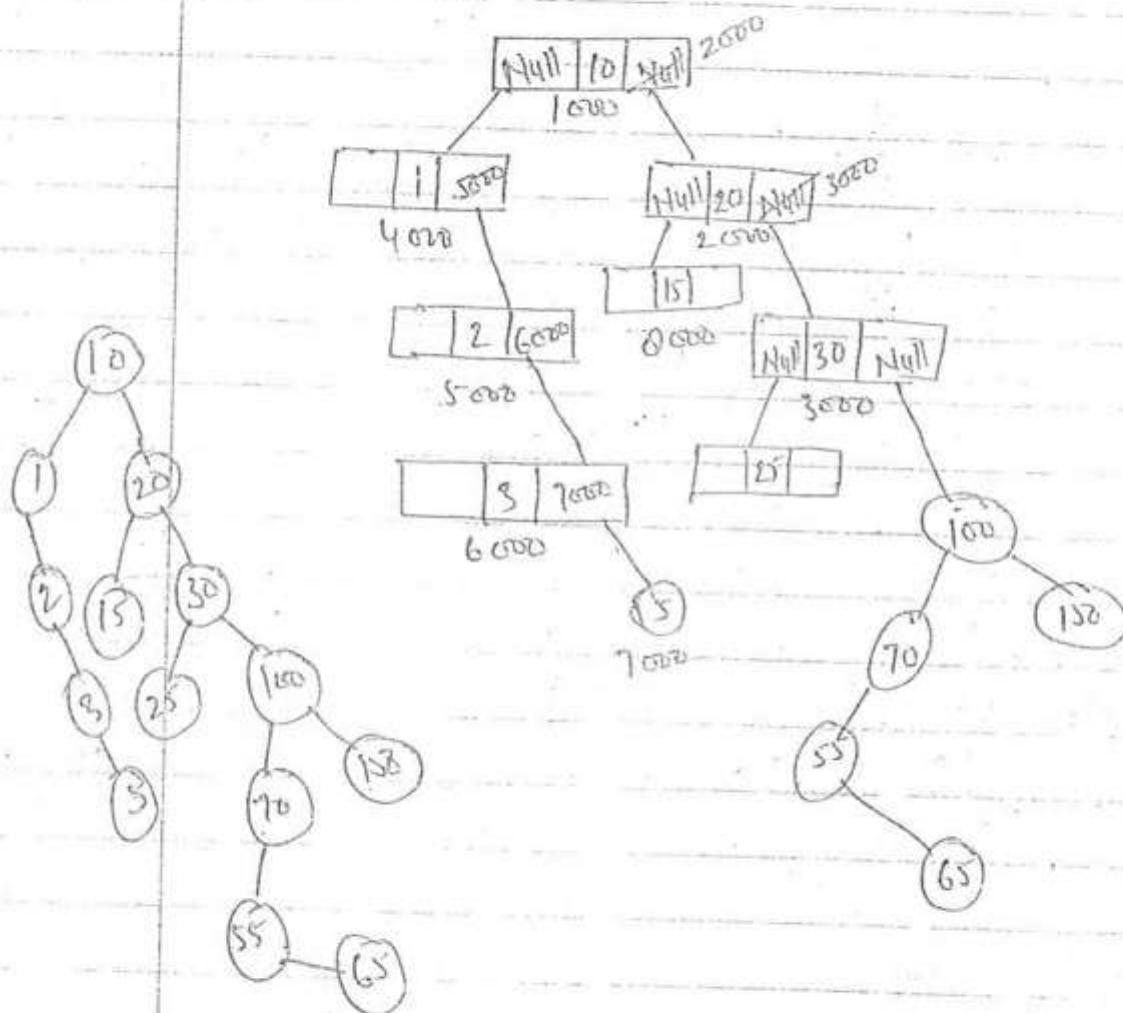


Binary Search Tree

Insert

Ex

10, 20, 30, 1, 2, 3, 5, 15, 25, 100, 150, 70, 55, 65.



Insert Algo

(1) Find correct place of the element to be inserted.

$$\Rightarrow O(n)$$

$$\Rightarrow \mathcal{O}(1)$$

$$\Rightarrow \text{Avg is } O(\log n)$$

(2) Insert the element by changing pointers.

$$\Rightarrow O(1)$$

$$\overbrace{O(n), \mathcal{O}(1), O(\log n)}^{\text{Avg}}$$

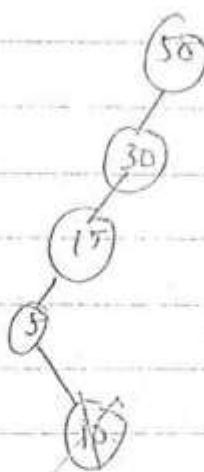
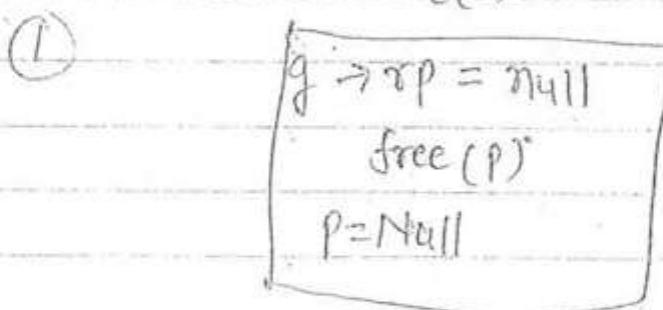
Delete

(1) 0 - children.

(2) 1 - children.

(3) 2 - children.

$$O(1)$$

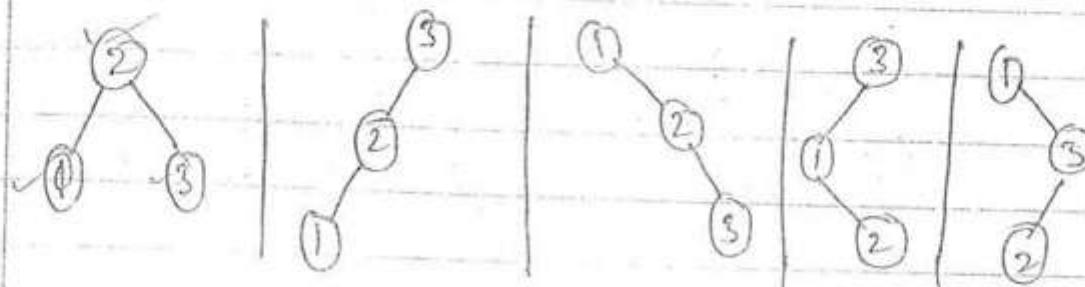


(4)

AVL Trees (Adelson, Velski , Landis)

- (1) Binary Search Tree.
 - (2) Balanced property.
- } Maximum $\log_2 n$

Q $n = 3 (1, 2, 3)$.



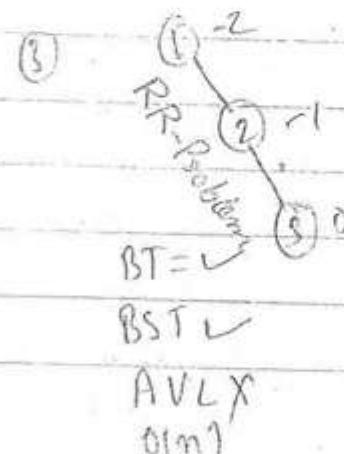
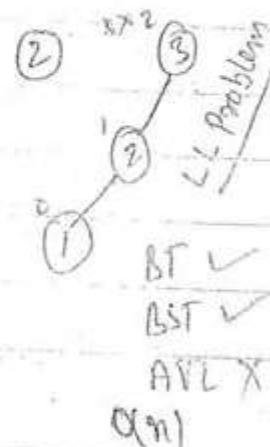
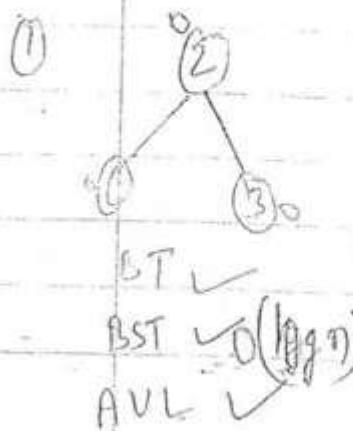
Balance factor: -
$$\left| H(LST) - H(RST) \right|$$

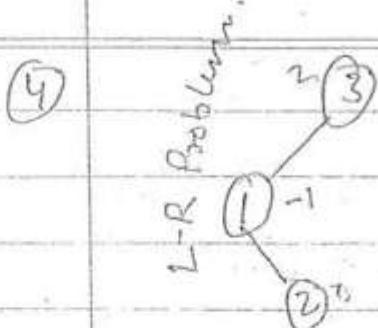
BST



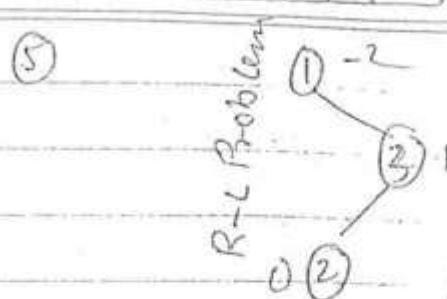
at every node

balance factor ($0 \leq 1 \leq -1$)



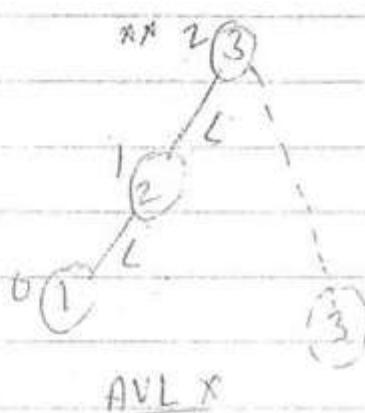


BT ✓
BST ✓
AVL X $O(n)$

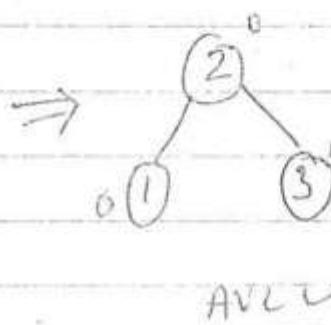


BT ✓
BST ✓
 $O(n)$ AVL X

(1) LL Problem (rotate Top half part right) carefully.

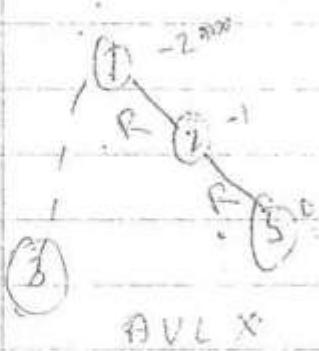


Inorder = 1, 2, 3

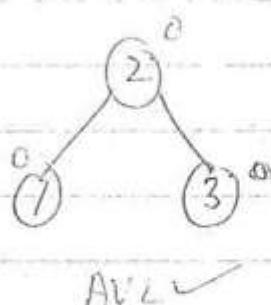


Inorder = 1, 2, 3.

(2) RR Problem (rotate Top half part left) carefully.

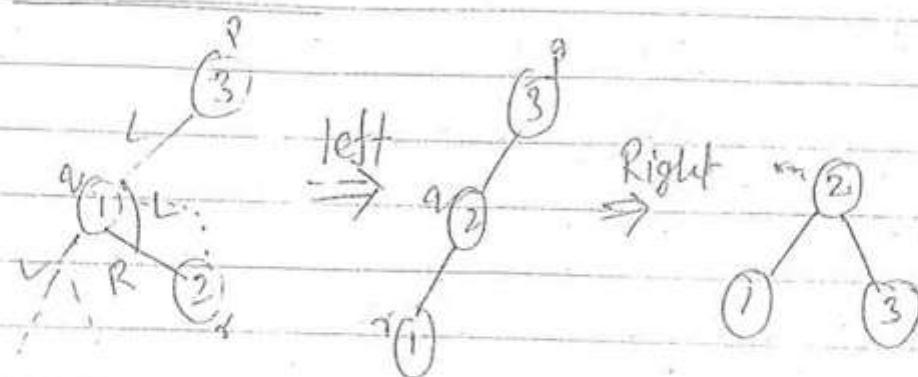


Inorder = 1, 2, 3.



Inorder = 1, 2, 3.

(3) LR Problem



$P \rightarrow \alpha p = \gamma$

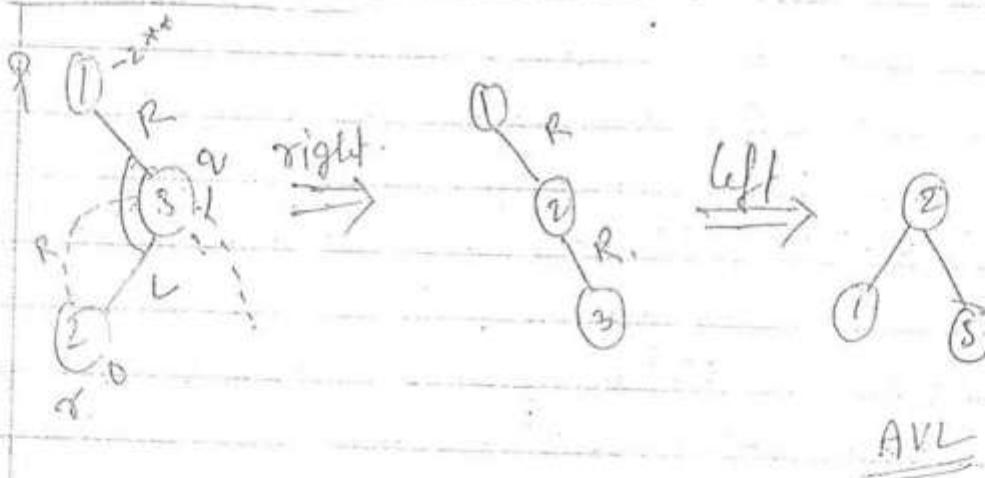
$r \rightarrow \alpha p = q$

$q \rightarrow \alpha p = p$

$r \rightarrow \alpha p = \text{null}$

AVL

(4) RL Problem



$P \rightarrow \alpha p = \gamma$

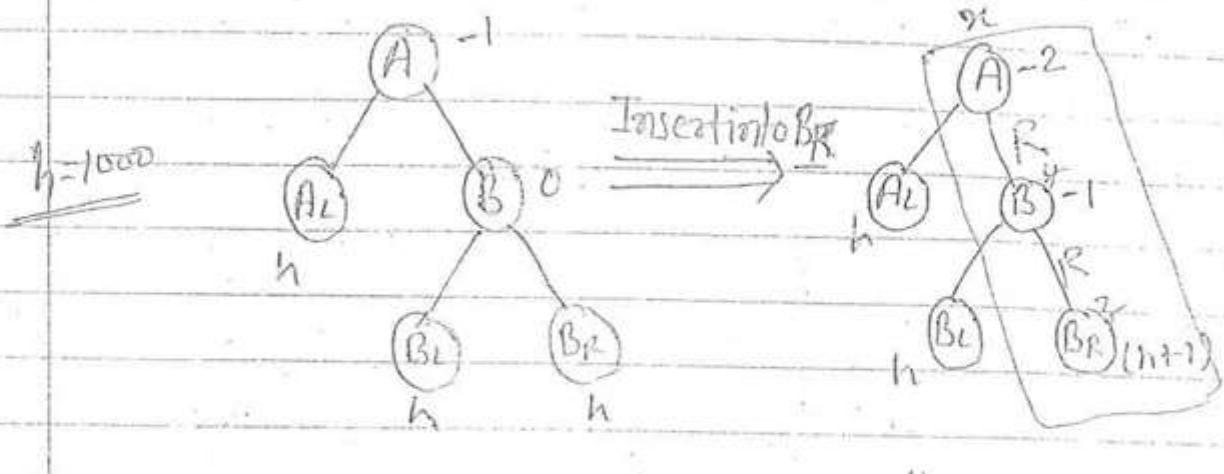
$r \rightarrow \alpha p = q;$

AVL

All of the above four problem takes $O(1)$

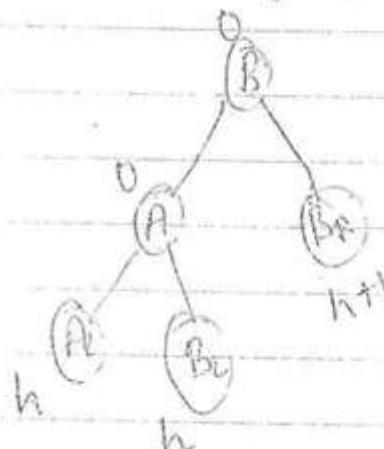
Insertion

RR- Problem



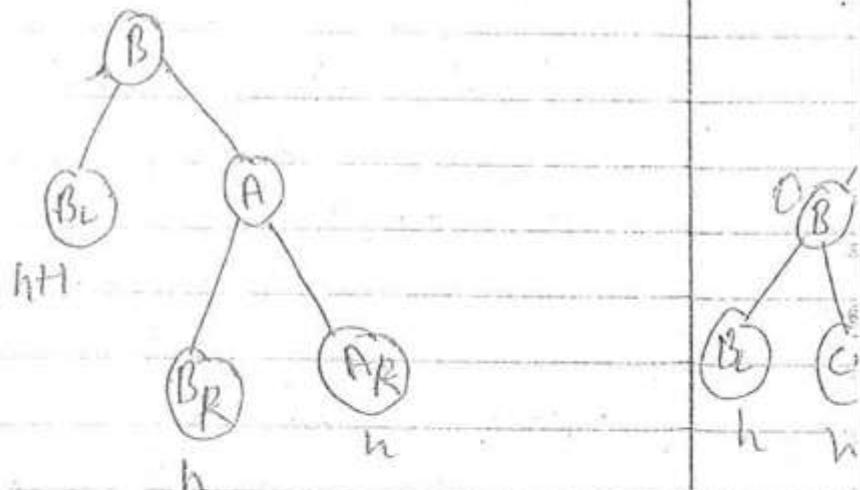
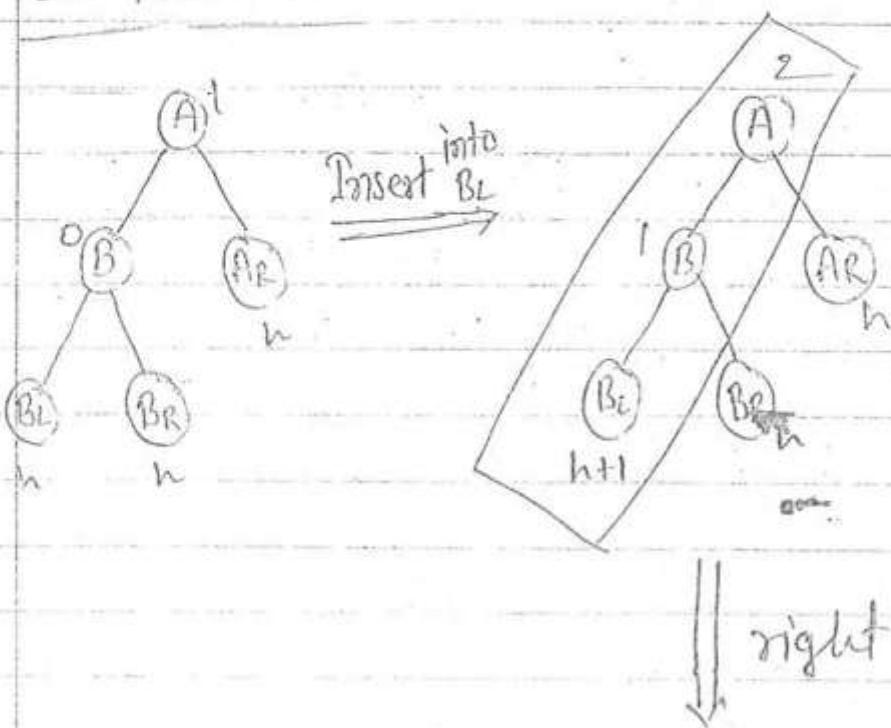
↓ left

$$y \rightarrow l_p = n$$



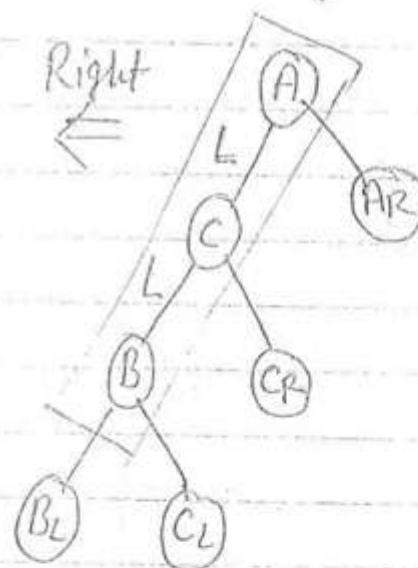
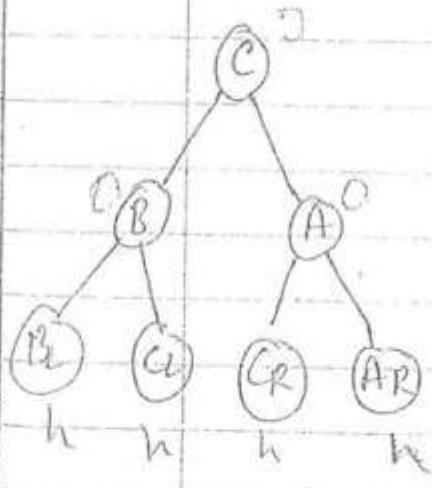
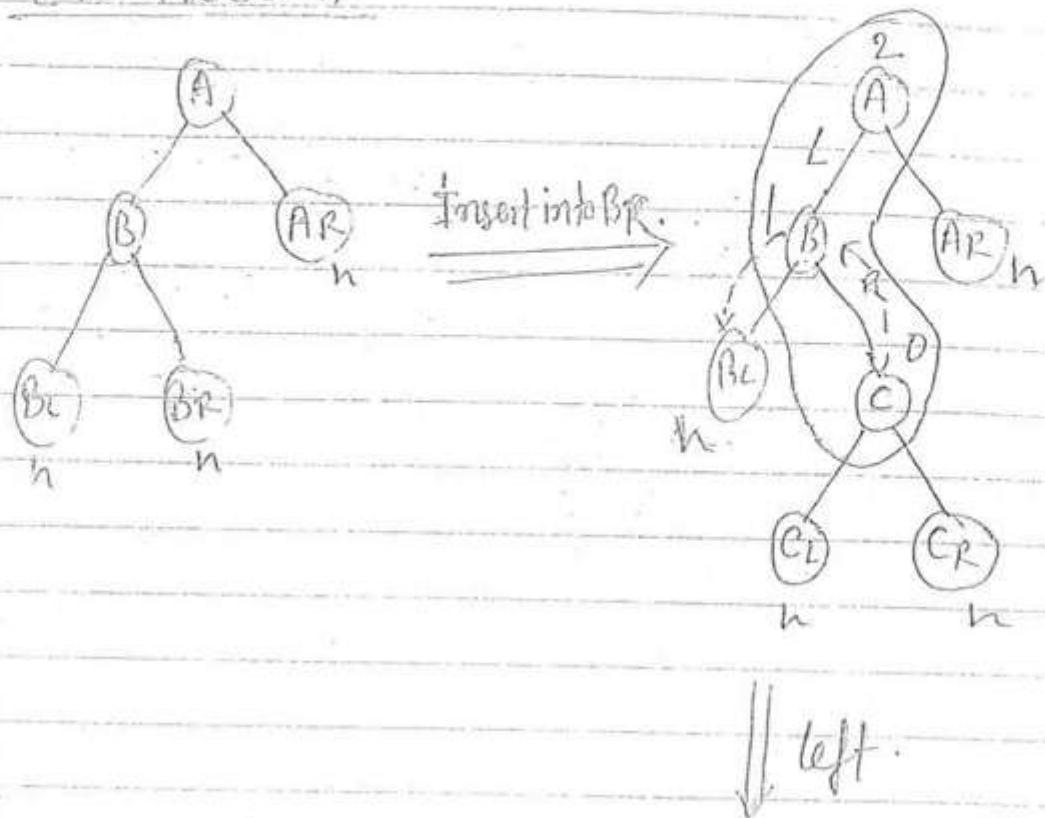
Insertion

LL Problem



Inception

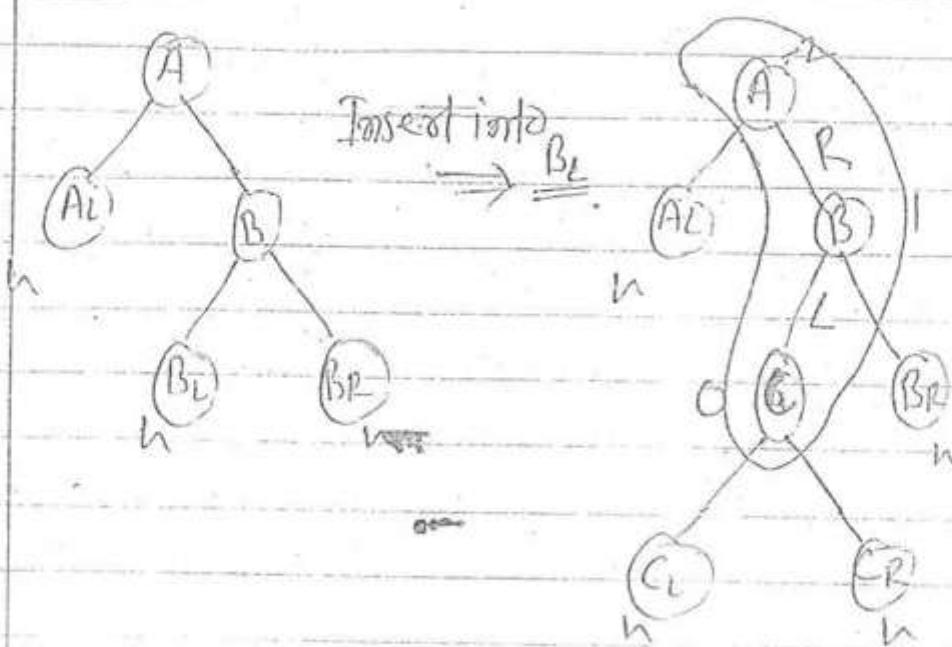
L.R. Problem.



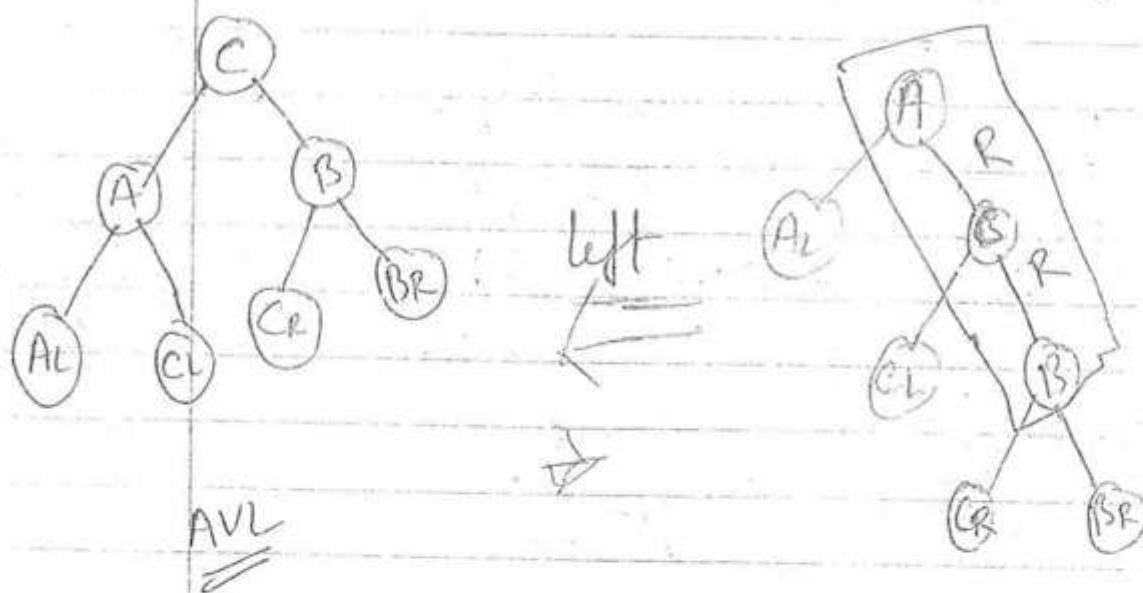
Rotation take $O(1)$
but R insertion take
 $O(\log n)$ time.

Inception

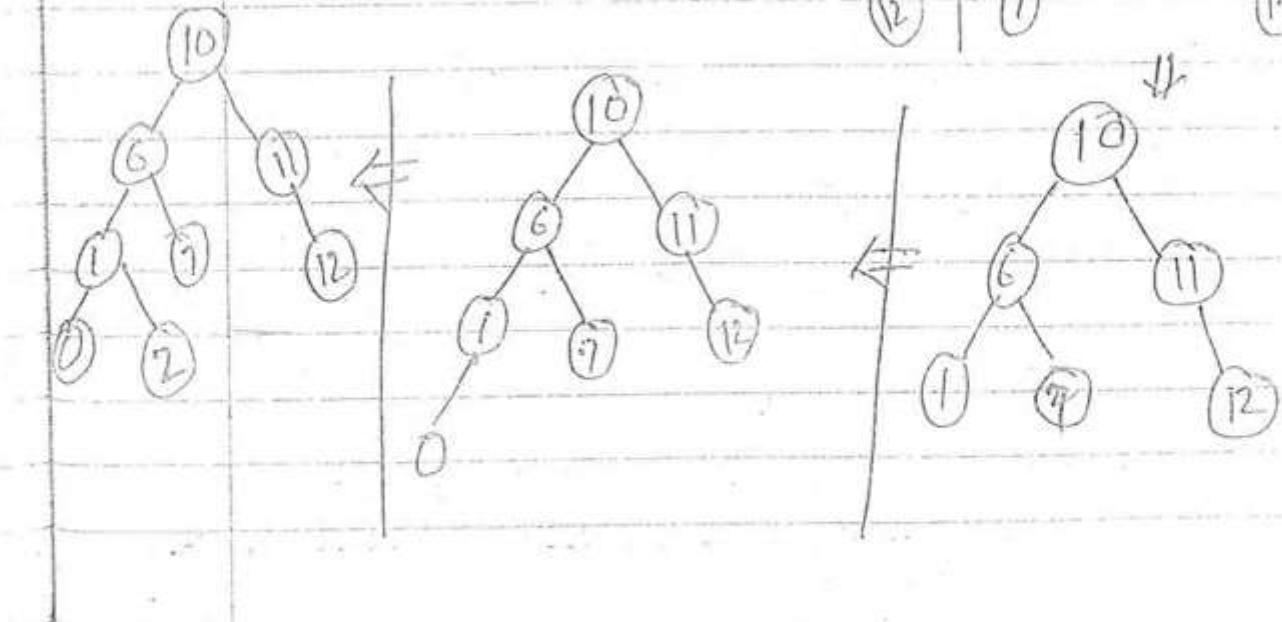
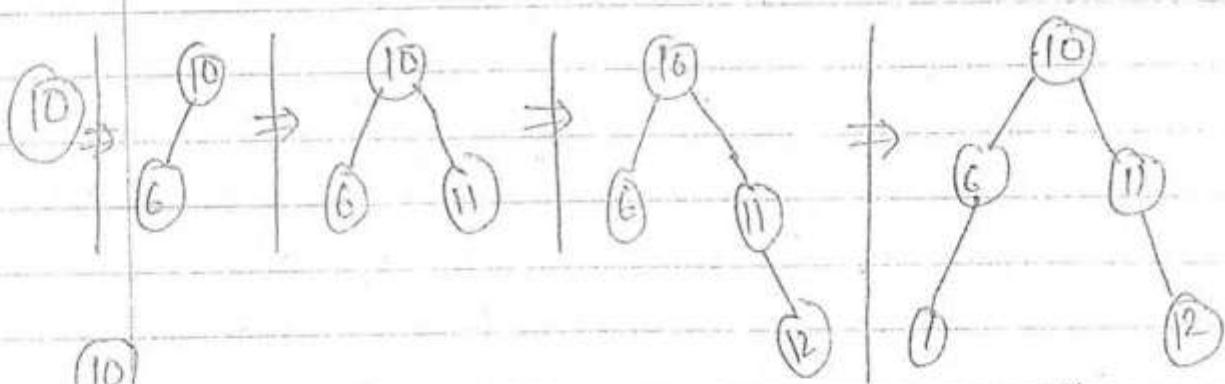
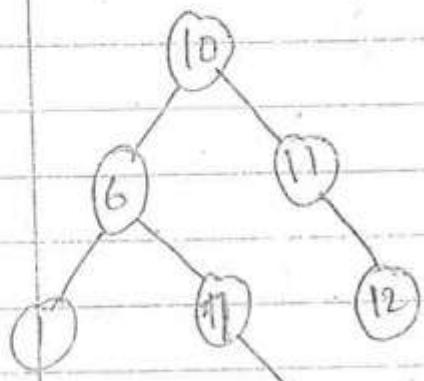
RL Problem

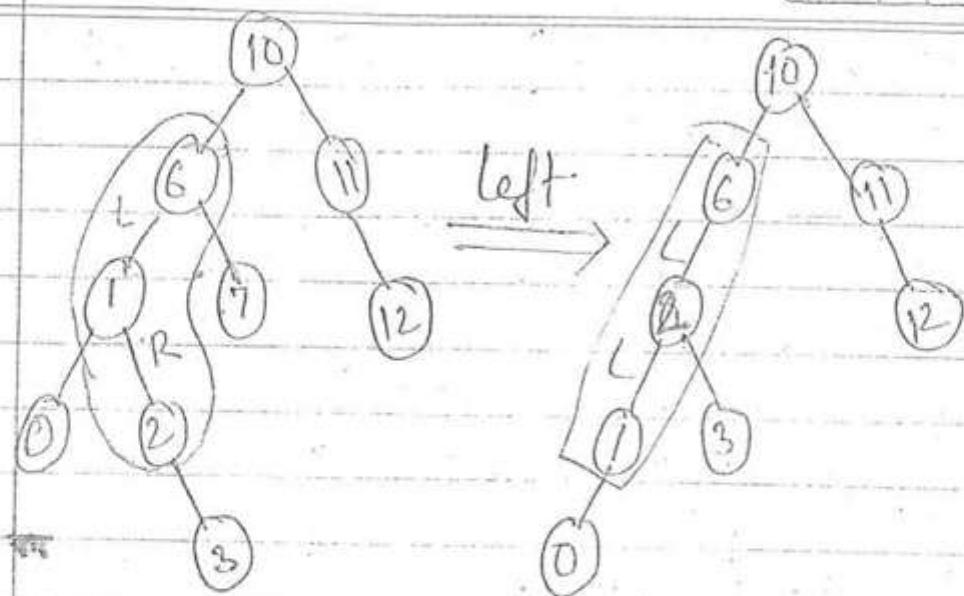


↓ Right.

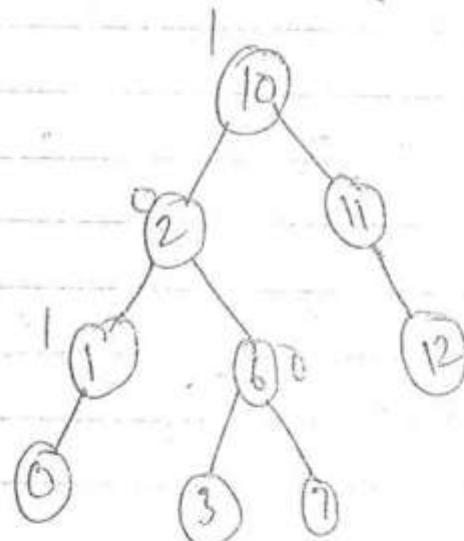


#8 What will be the resulting AVL tree if we insert 10, 6, 11, 12, 1, 7, 0, 2, 3. in that order.



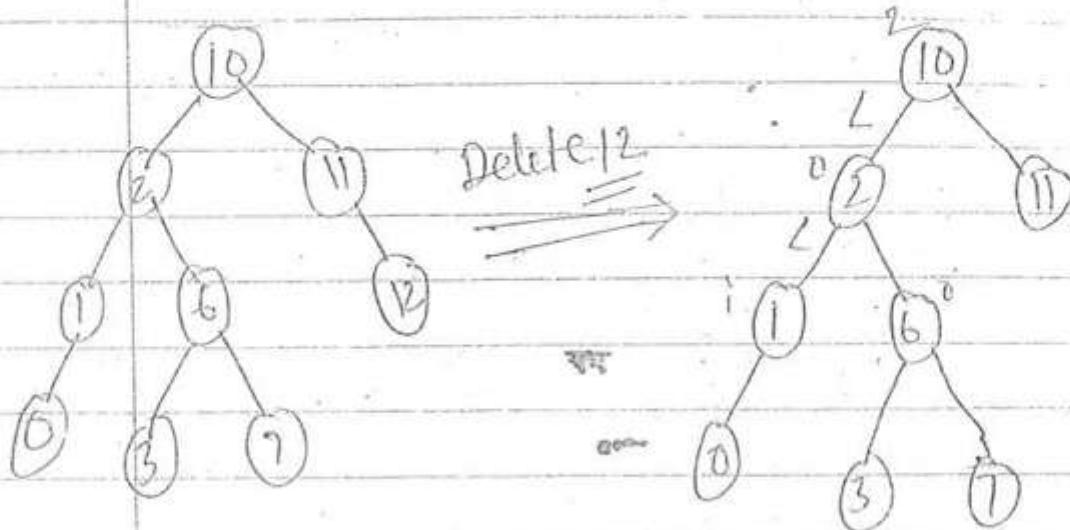


↓ Right

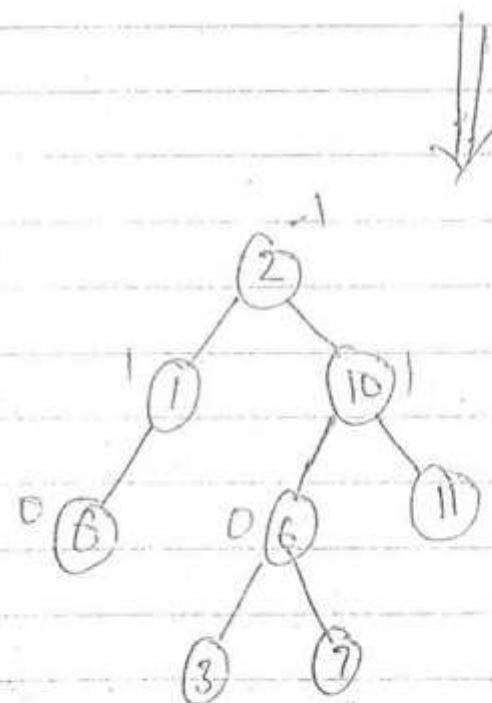


AVL Tree

#1 What will be resulting AVL tree if we delete 12 from above AVL tree.



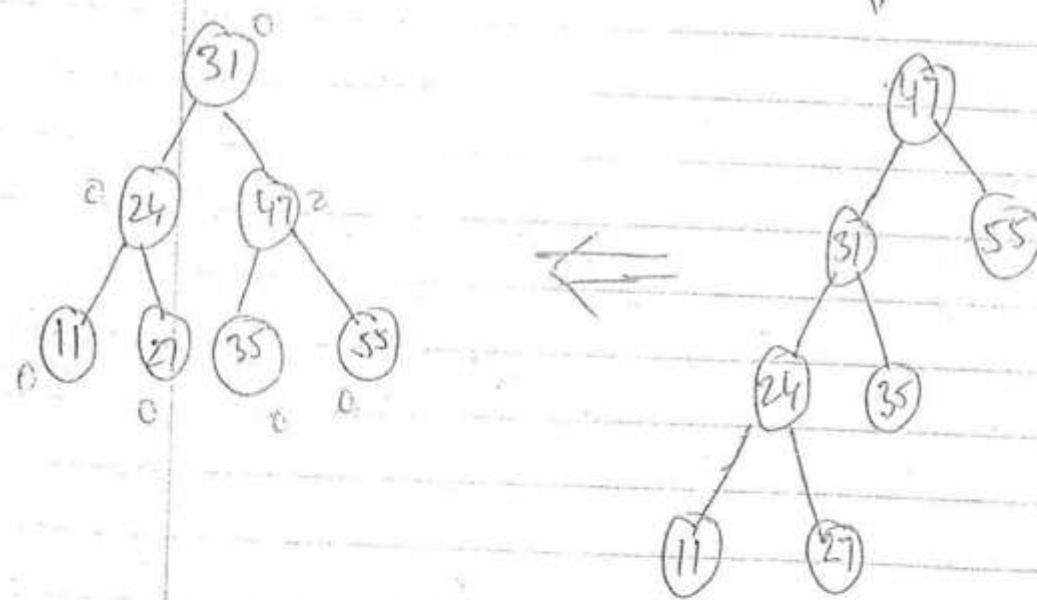
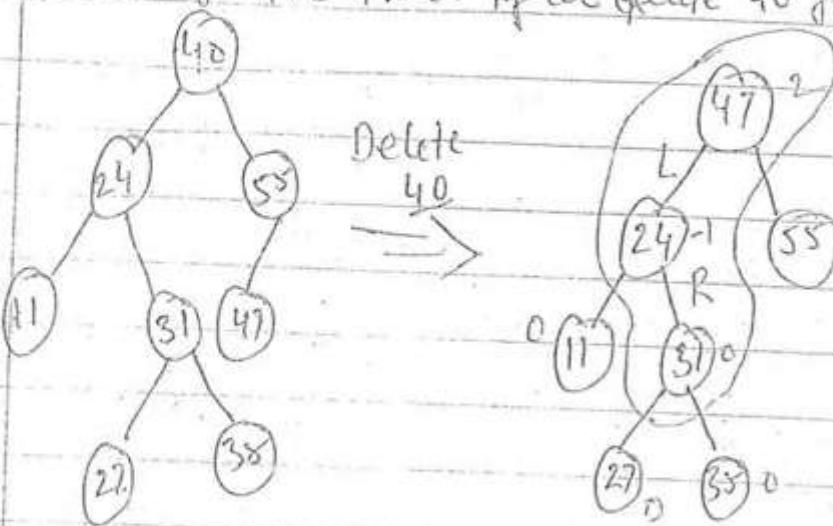
LL and LR got both problem but LL is better becoz LL has less time complexity we do 2 rotate



if we LL & LR problems we should go for LL.

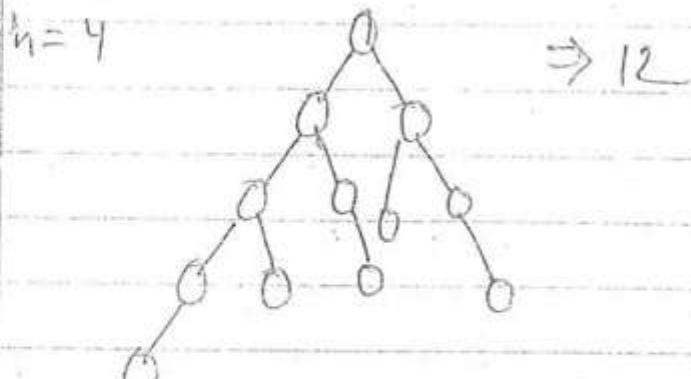
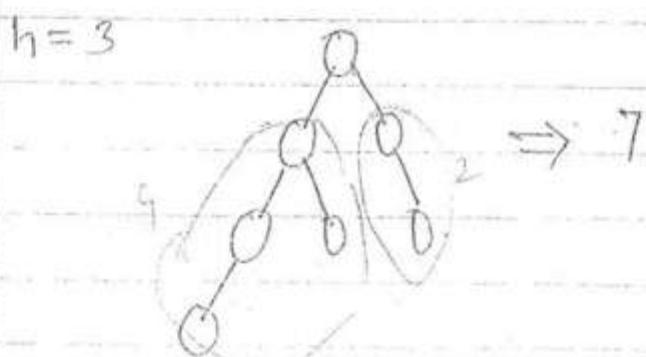
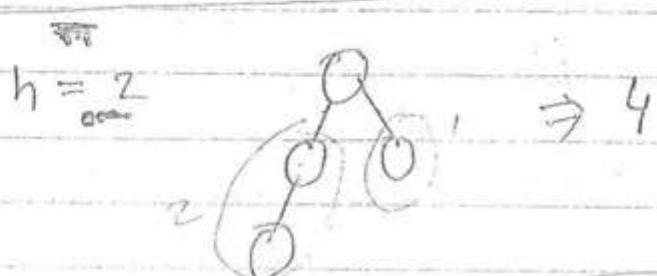
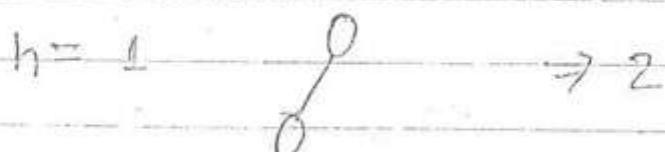
#1

Consider the following AVL tree, what will be the resulting AVL tree if we delete 40 from above tree.



~~#~~ What is the max^{!!} height of any AVL tree with 7 nodes. ?

$$h=0 \quad 0 \rightarrow 1$$



Min^m no of nodes in height(h) AVL tree
is

$$\boxed{\text{Min}(h) = \text{Min}(h-1) + \text{Min}(h-2) + 1}$$

$$h=0 \Rightarrow 1$$

1P

$$h=1 \Rightarrow 2$$

$$h=2 \Rightarrow 4$$

$$h=3 \Rightarrow 7$$

$$= 20$$

$$h=4 \Rightarrow 12$$

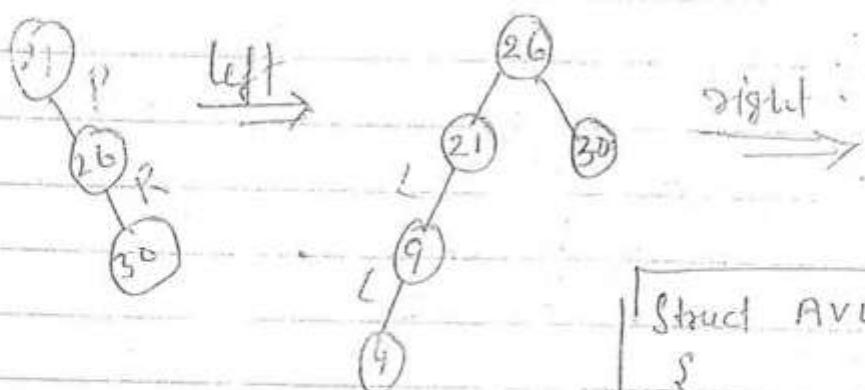
$$= \underline{\underline{20}}$$

* P

Create AVL tree for the following nodes.

21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 8, 7.

from above AVL tree delete - 2, 8, 10
18, 4, 9, 14, 7, 15.



every node have height

Struct AVL

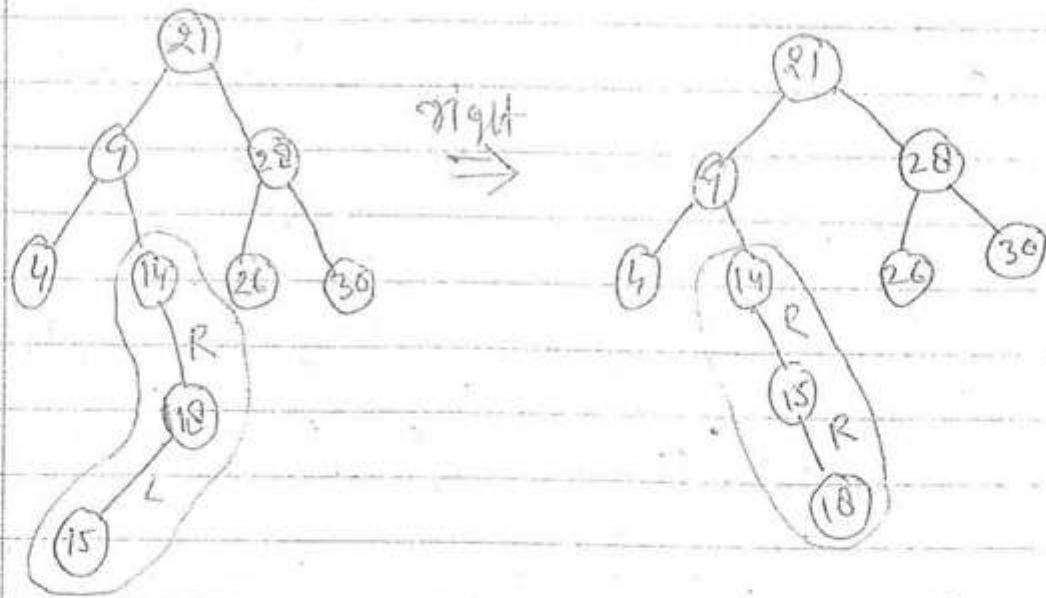
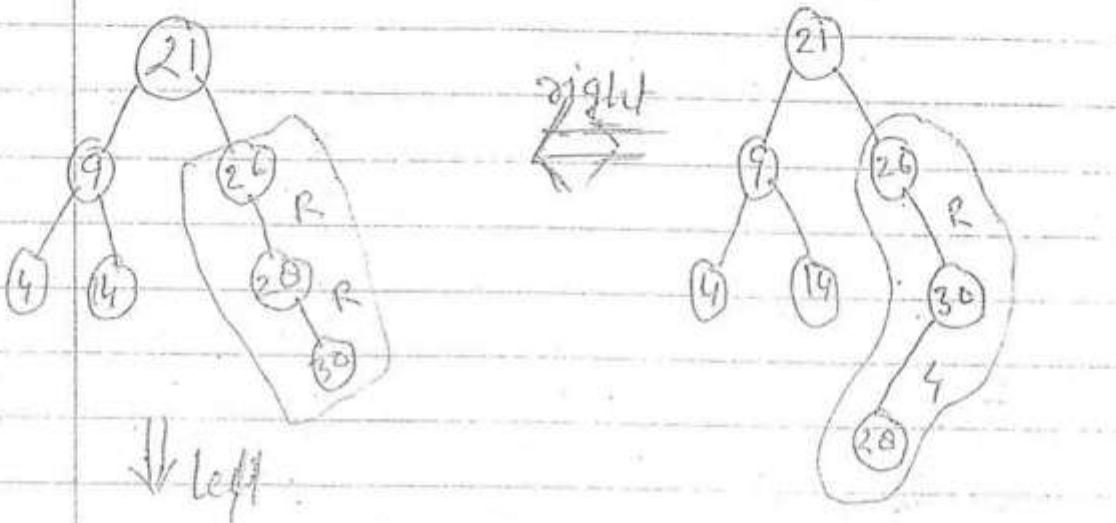
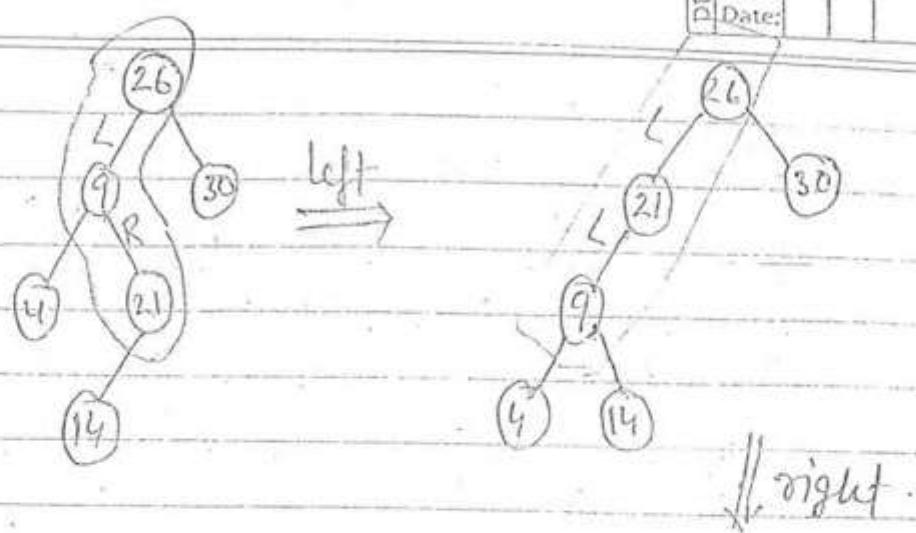
{

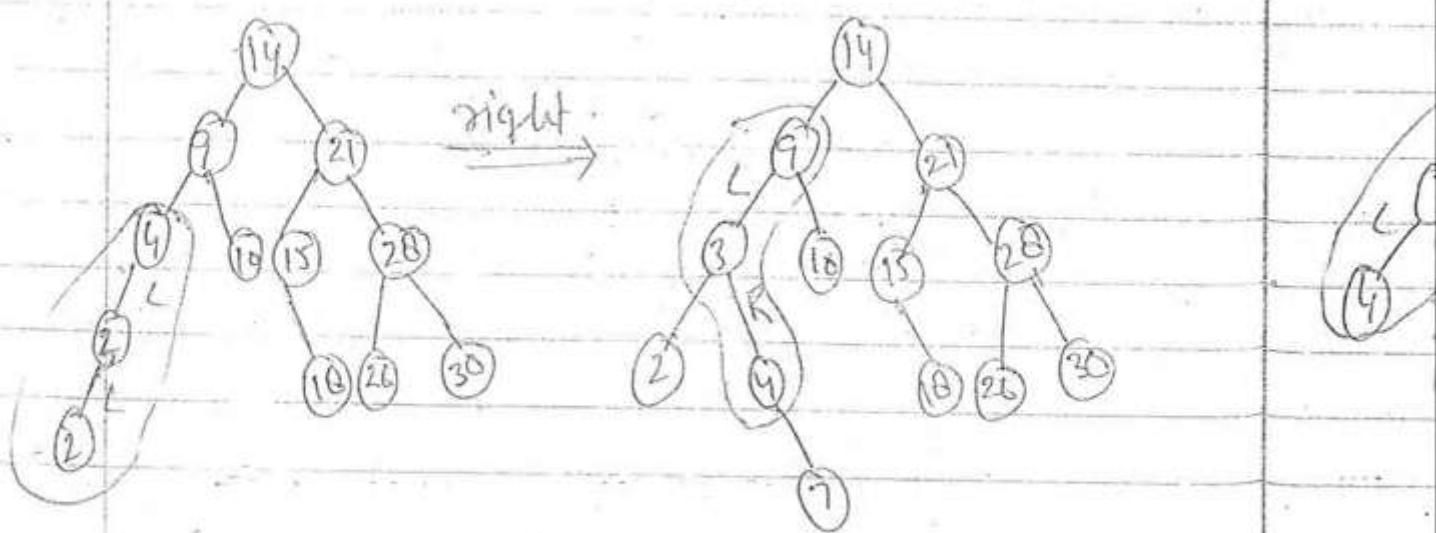
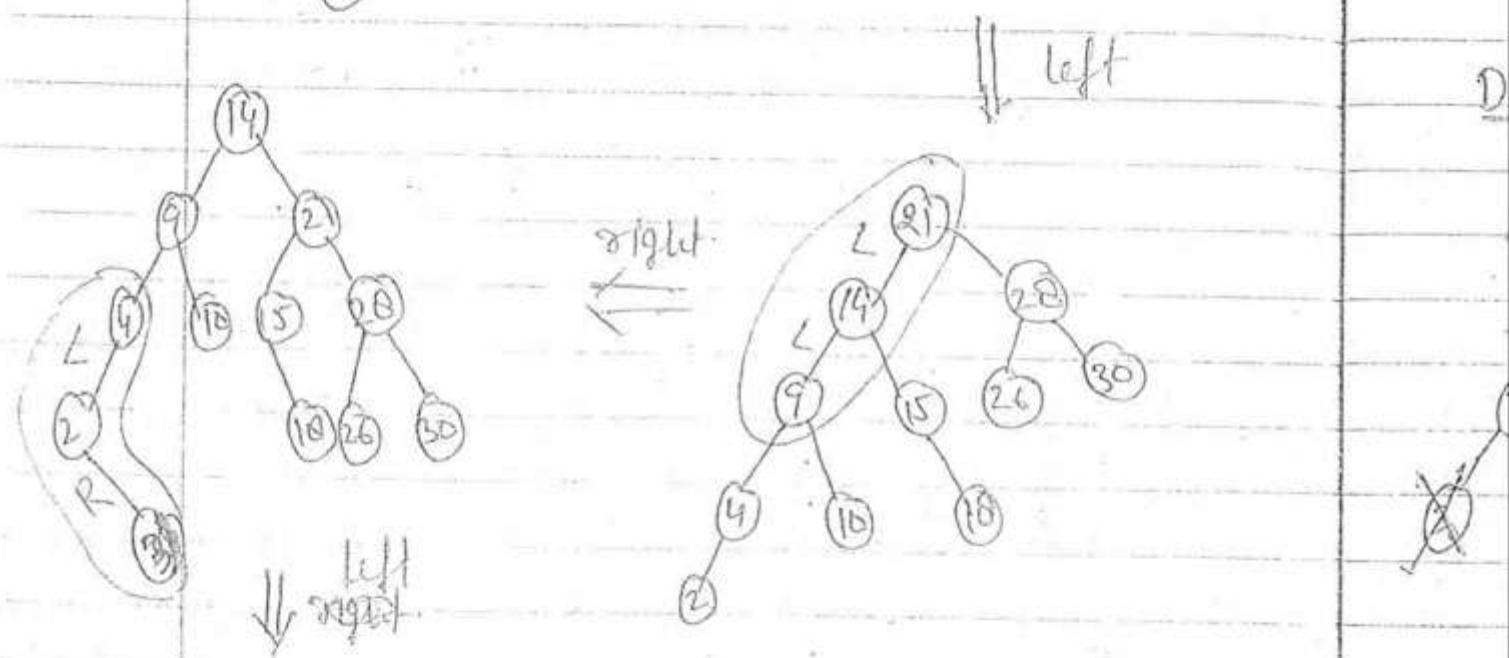
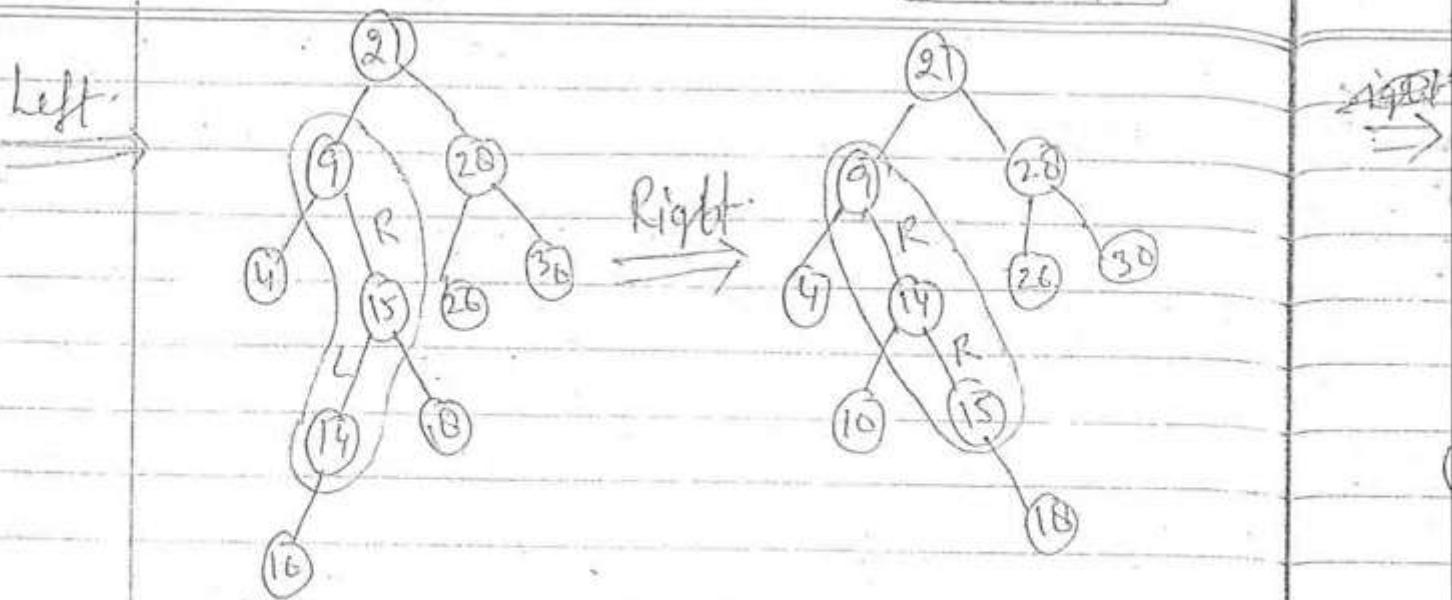
int data;

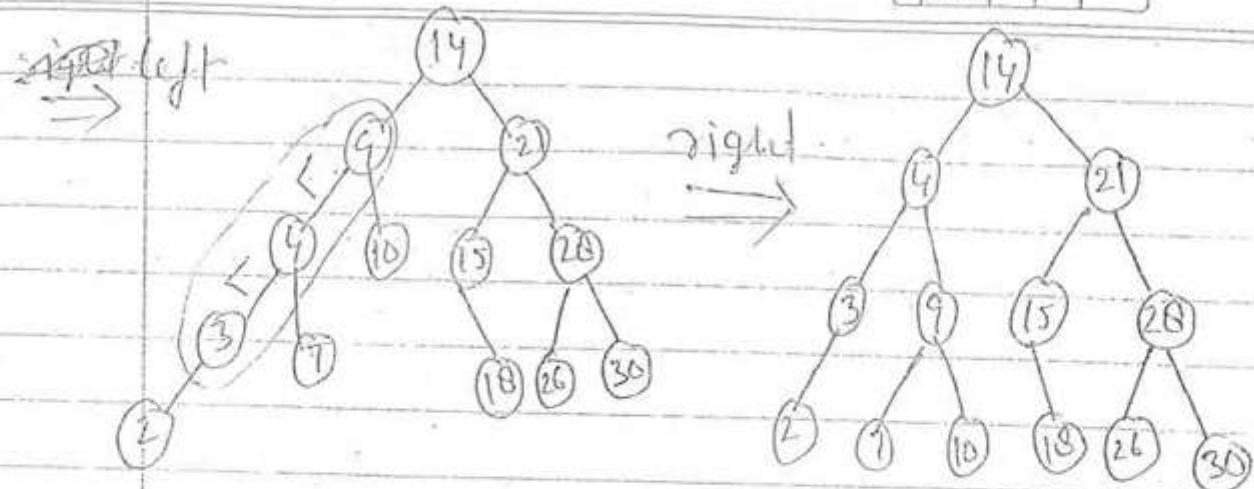
Struct AVL, *dp, **np;

int height;

}

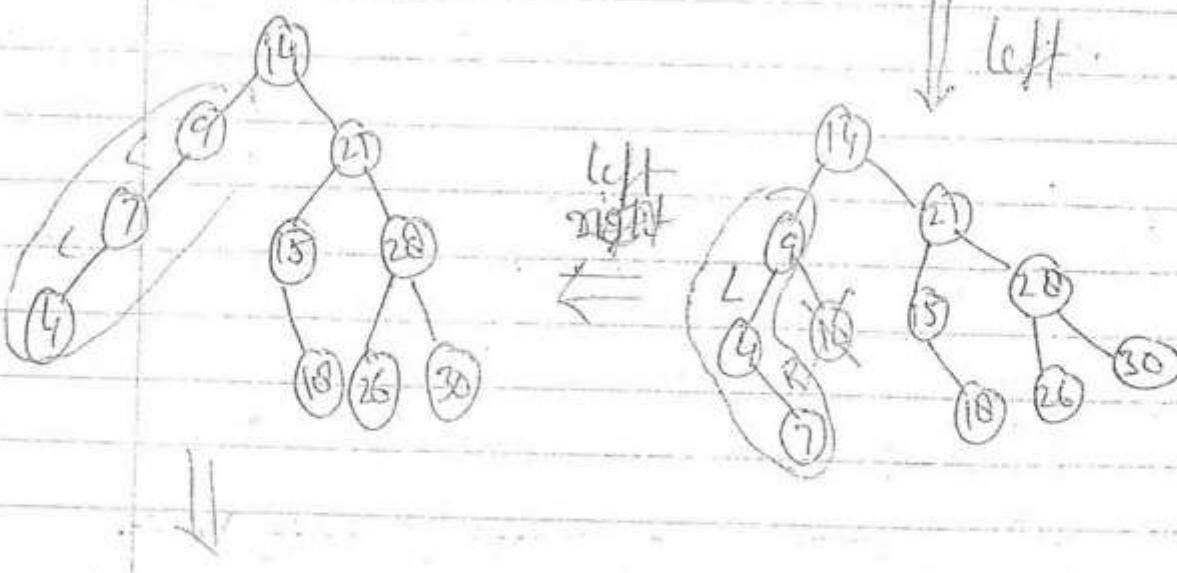
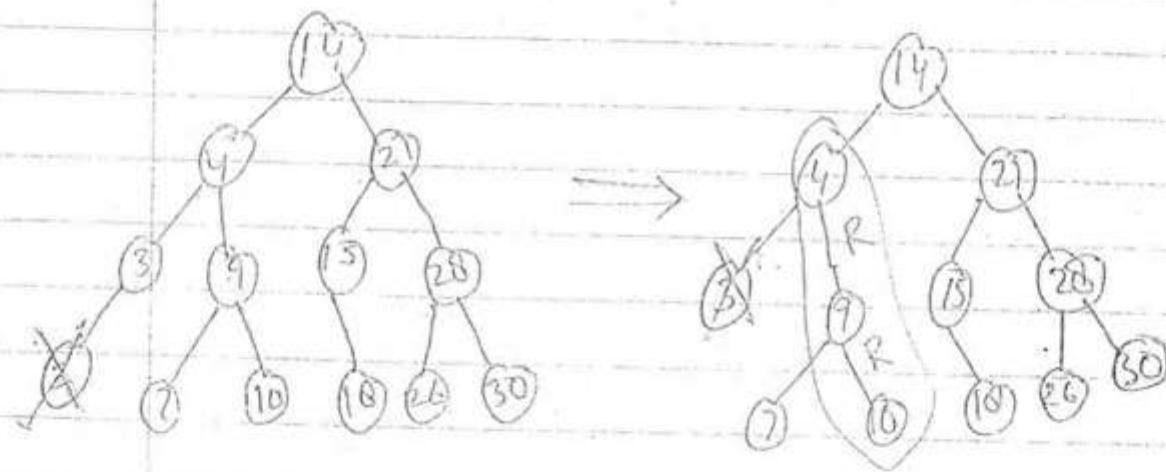


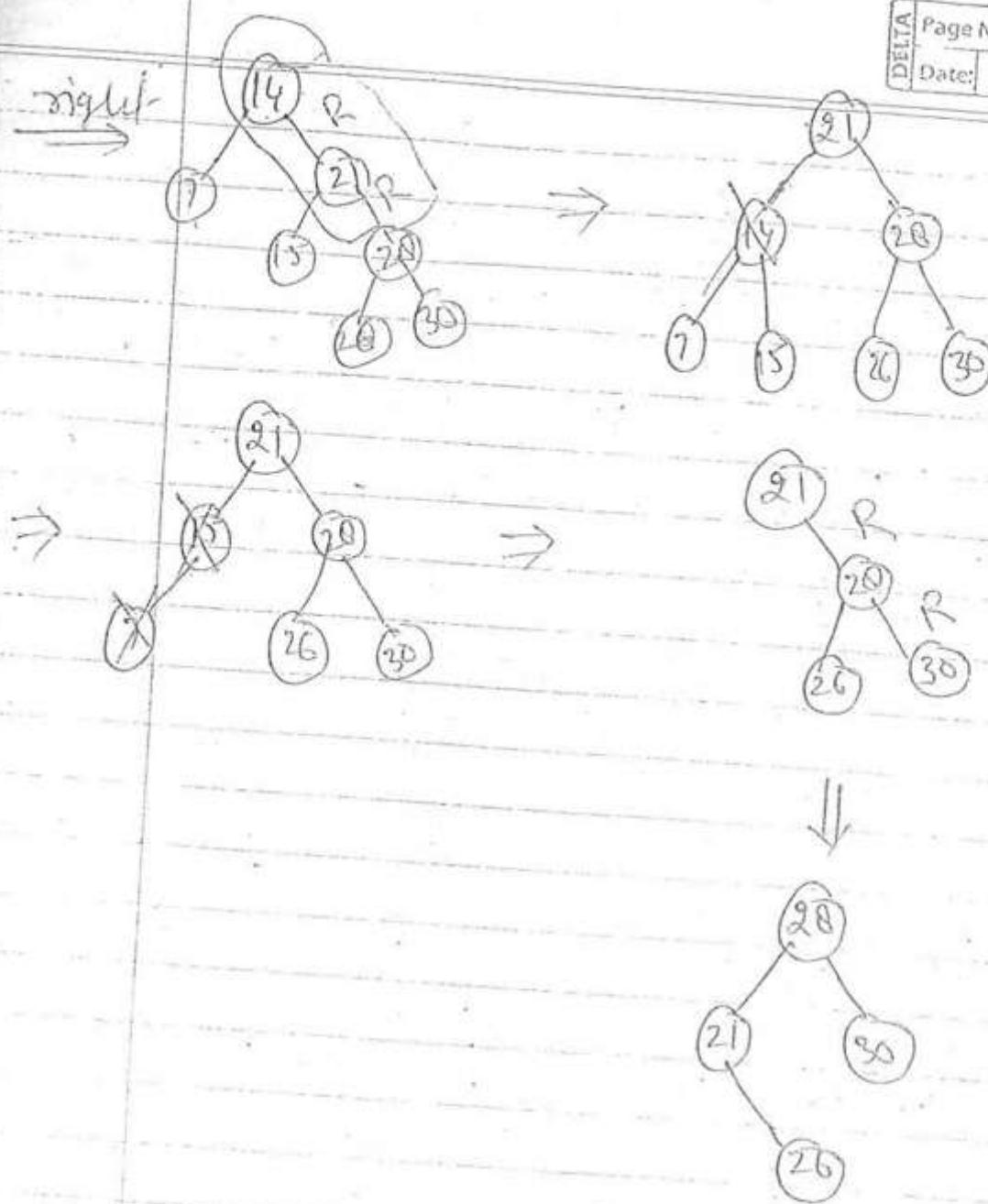




AVL Tree

Delete $\Rightarrow 2, 3, 10, 18, 4, 9, 14, 7, 15$.





Note - The height of AVL tree which containing n nodes = $1.44 \log_2 n$.

$$1.44 \log_2 n < 2 \log n$$

AVL tree height no more than $2 \log n$.

② Inserting an element into an AVL tree which already contain n elements
 $= \Theta(\log n)$.

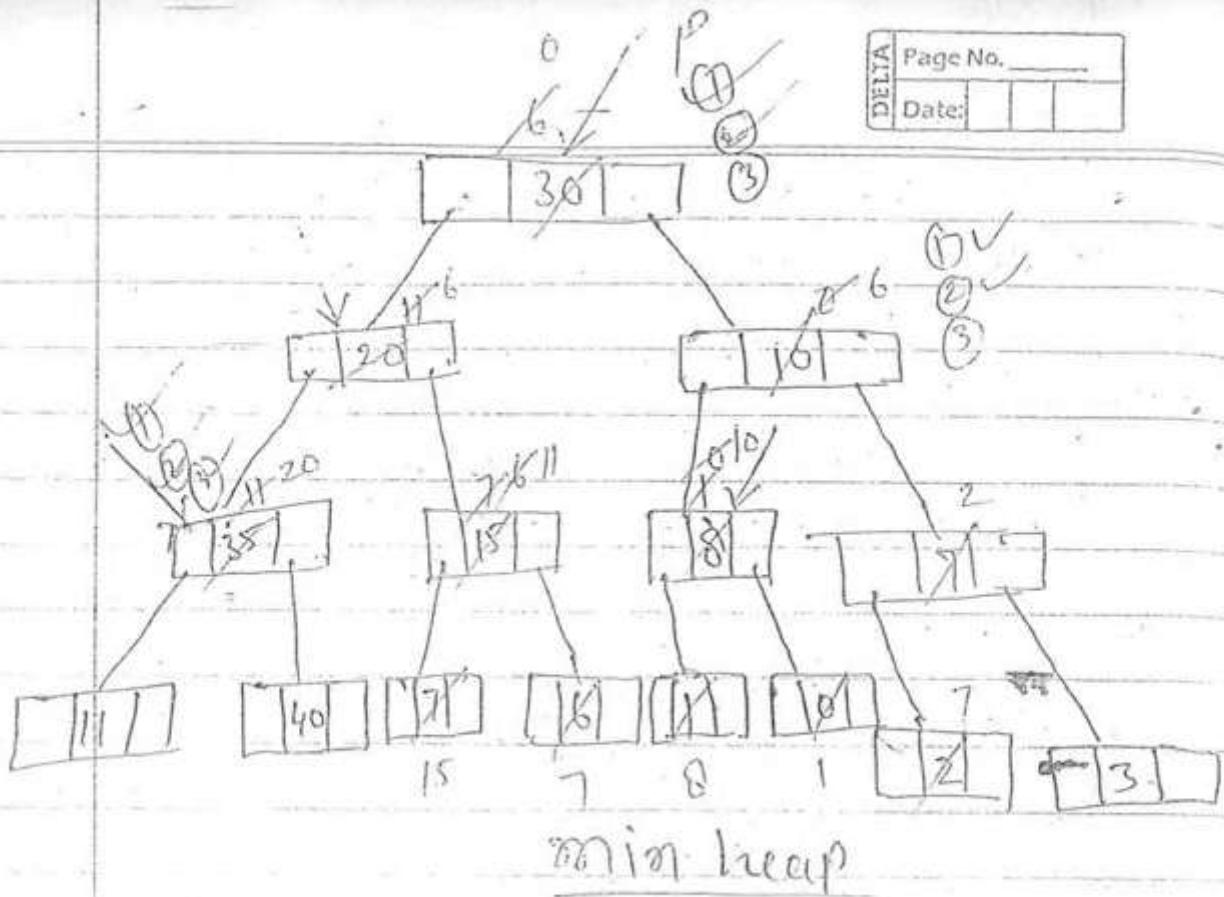
③ deleting an element from AVL tree the which already contain n elements.
 $= \Theta(\log n)$

④ In a AVL tree insertion, deletion takes $O(n)$ time when AVL tree have no height.

```

A(P)
{
    if (P!=Null)
    {
        if (P->data > (P->lP->data))
            swap (P->data, P->lP->data)
        if (P->data > (P->rP->data))
            swap (P->data, P->rP->data)
    }
}

```



B(P)

{
if ($\text{p} \rightarrow \text{lp} \rightarrow \text{lp} \neq \text{Null}$ & & $\text{p} \rightarrow \text{rp} \rightarrow \text{rp} \neq \text{Null}$)

{

① B($\text{p} \rightarrow \text{lp}$)

② B($\text{p} \rightarrow \text{rp}$)

③ if ($\text{p} \rightarrow \text{data} > \text{p} \rightarrow \text{lp} \rightarrow \text{data}$)

swap ($\text{p} \rightarrow \text{data}$, $\text{p} \rightarrow \text{lp} \rightarrow \text{data}$)

if ($\text{p} \rightarrow \text{data} > \text{p} \rightarrow \text{rp} \rightarrow \text{data}$)

swap ($\text{p} \rightarrow \text{data}$, $\text{p} \rightarrow \text{rp} \rightarrow \text{data}$)

}

else

{

If ($p \rightarrow \text{data} > p \rightarrow lp \rightarrow \text{data}$)

swap ($p \rightarrow \text{data}, p \rightarrow lp \rightarrow \text{data}$)

If ($p \rightarrow \text{data} > p \rightarrow rp \rightarrow \text{data}$)

swap ($p \rightarrow \text{data}, p \rightarrow rp \rightarrow \text{data}$)

}

B-Tree (Multi way searching)

DELTA Page No. _____
Date: _____

- ① It is also one of the search tree.
- ② Minimum 2 - children.
- ③ All leaf node in the same level.

Fanout/degree/Order of the B-tree

Order of the B-tree = ~~m~~ (maximum children) \Rightarrow

= m or ~~(Keys)~~ (max^m Keys)

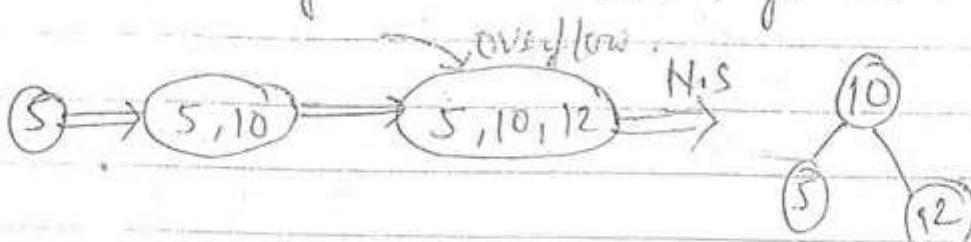
= $\lceil \frac{m}{2} \rceil$ (min^m children) \Rightarrow

= $\frac{m}{2} - 1$ (min^m Keys)

~~Ex~~ Consider the following elements 5, 10, 12, 13, 14,
1, 2, 3, 4, degree = 3

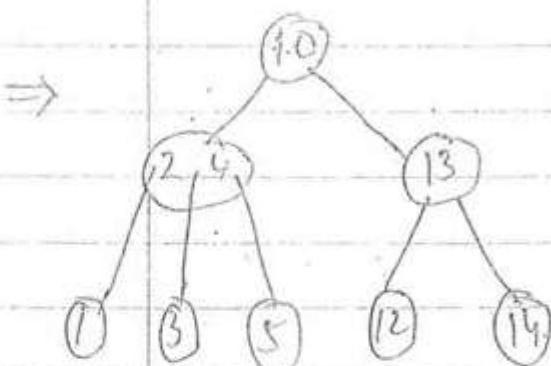
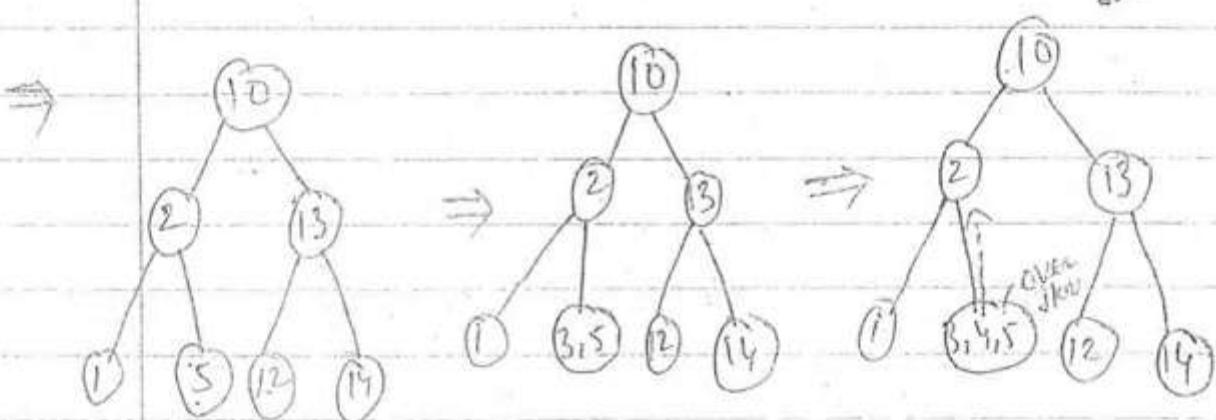
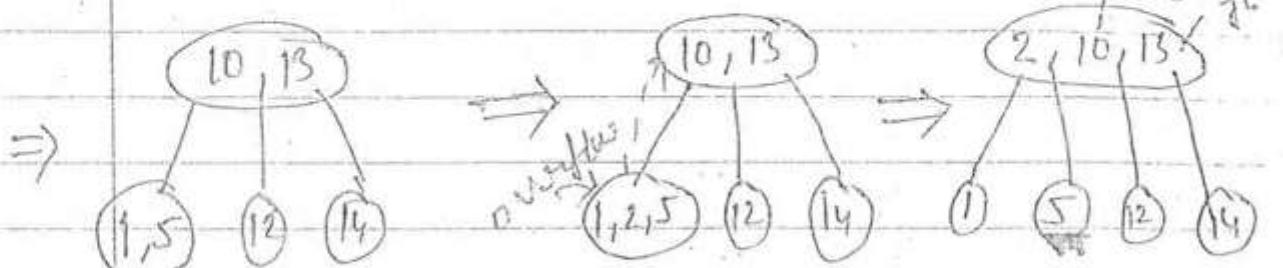
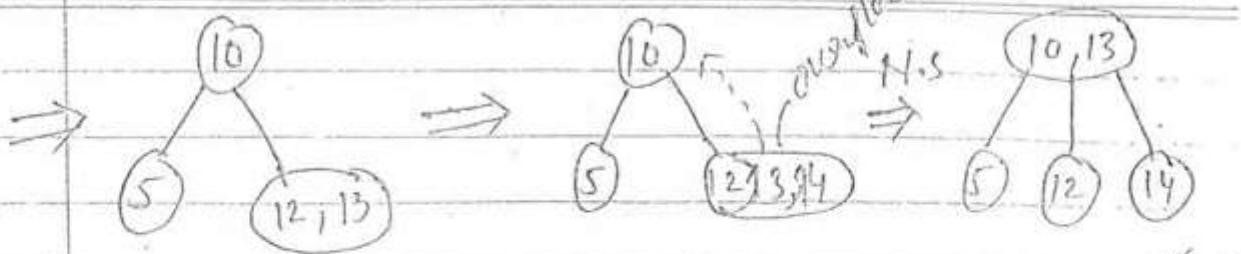
Max^m children = 2 min^m children = $\frac{3}{2} = 1.5 = 2$ \Rightarrow

Max^m keys = 2 min^m keys = $2 - 1 = 1$



When root is 1) overflow merge 2) under flow merge

DETA	Page No.	
	Date:	



$$O(\log n)$$

Degree 5

5, 10, 12, 13, 14, 1, 2, 4, 20, 18, 19, 17, 16, 15

25, 23, 24, 22, 11, 30, 31, 28, 29.

Max^m children = 5

Keys = 4

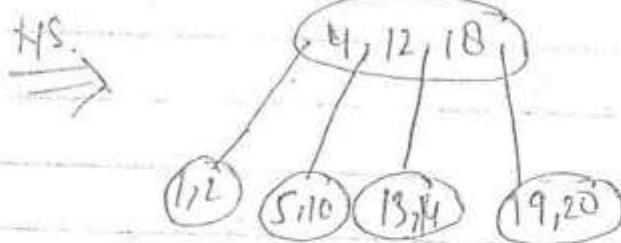
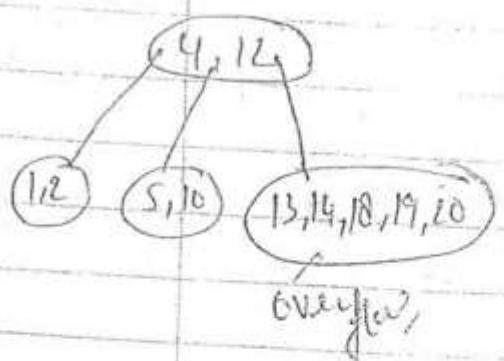
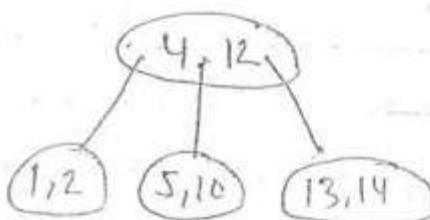
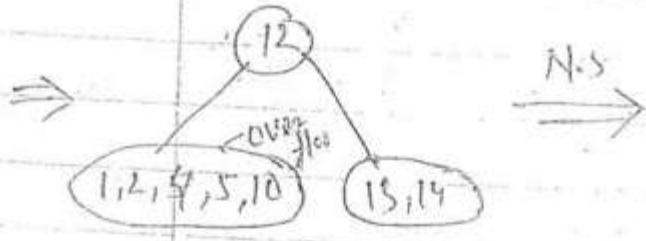
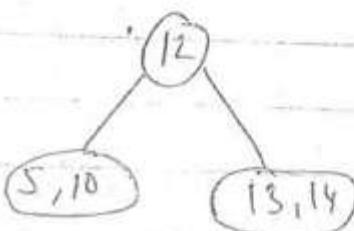
Min^m children = 3

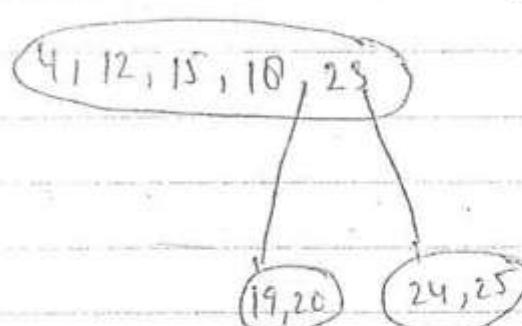
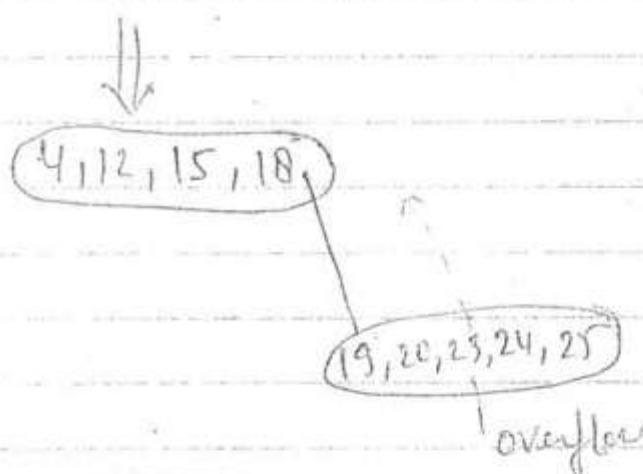
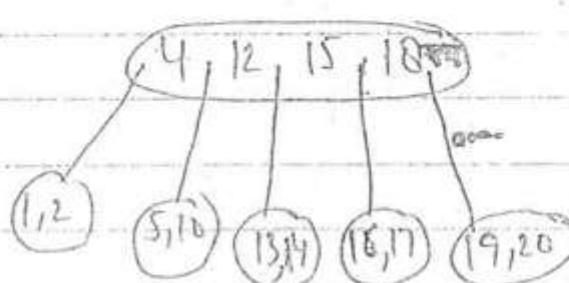
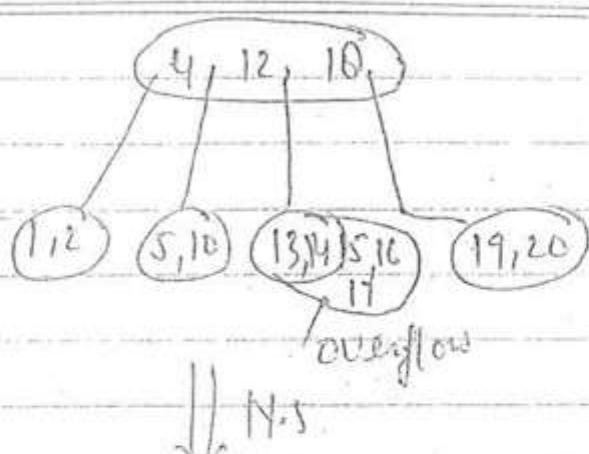
Keys = 2.

overflow

5, 10, 12, 13, 14

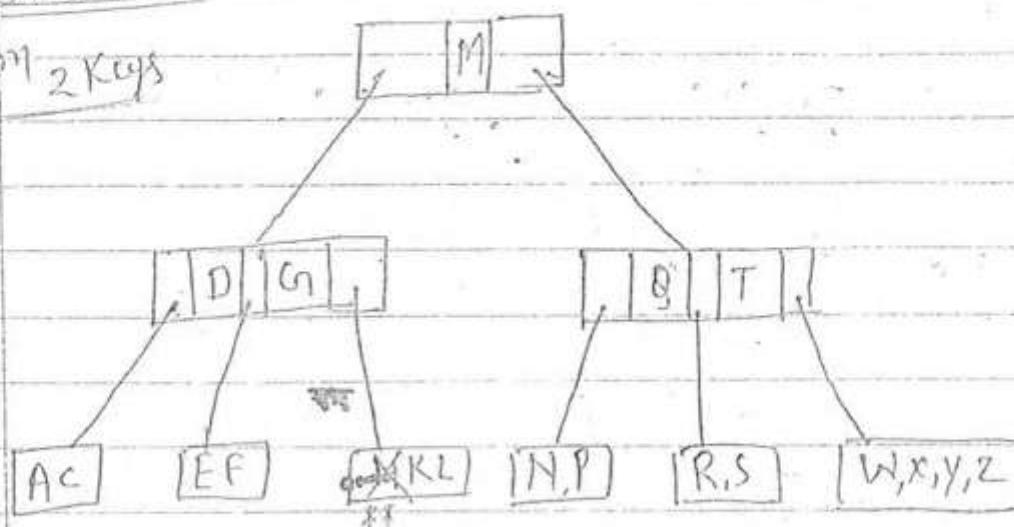
N.S





Consider the following B+ tree.
Deletion

Min 2 Keys



$$D = 6$$

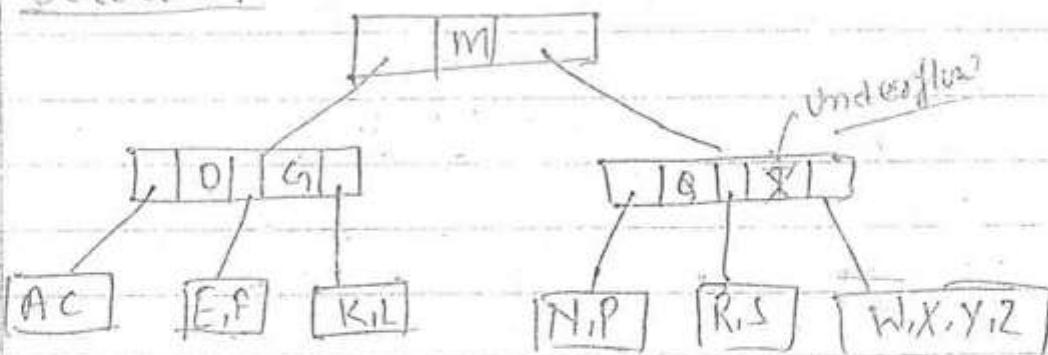
$$\max^{m}_{\text{ch}} = 6$$

$$K = 5$$

$$\min^{m}_{\text{ch}} = 3$$

$$\text{Key} = 2$$

Delete-H

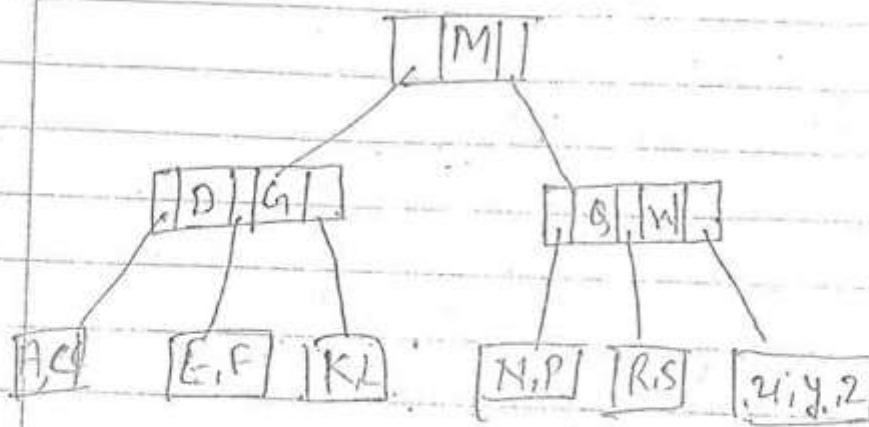


In the above tree after deleting T that node is in underflow but one of the children is having more than the required keys. So it will send

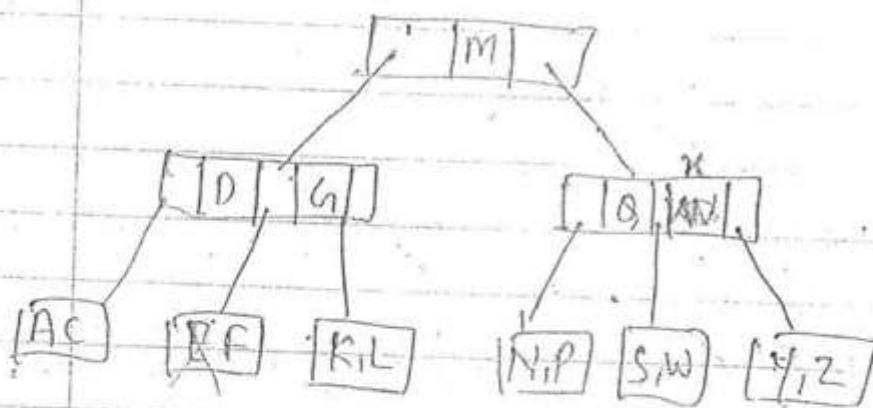
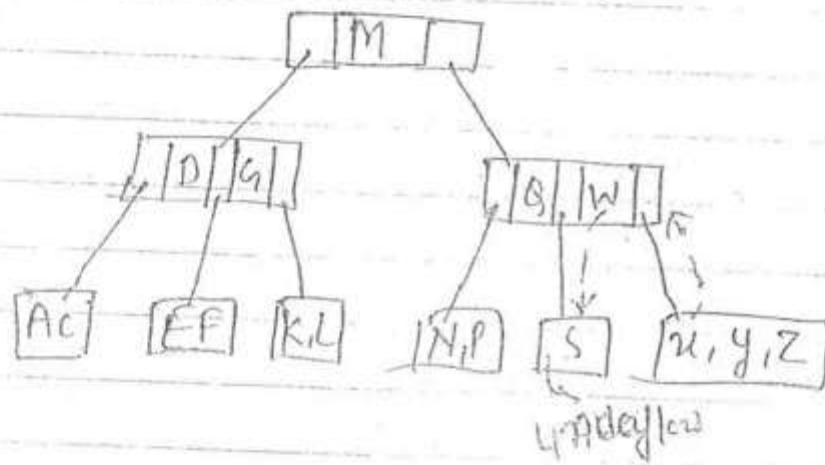
on key to its parent.

DELTA	Page No. _____
	Date: _____

Delete - T

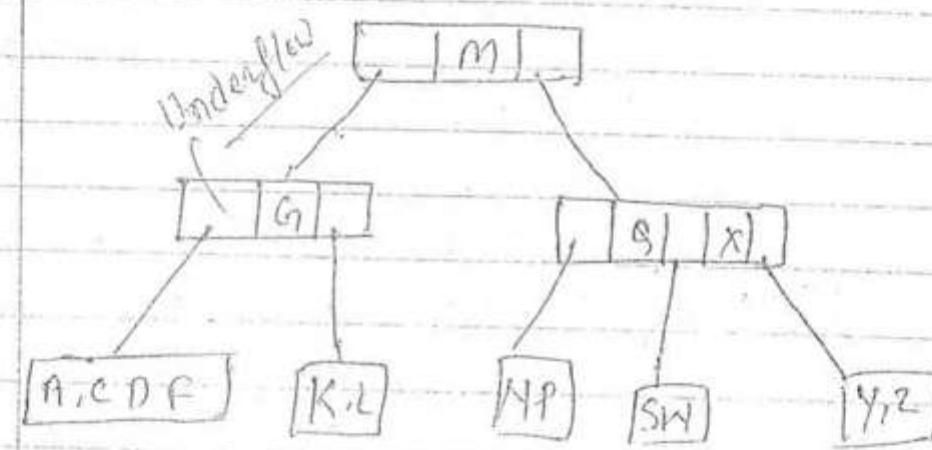
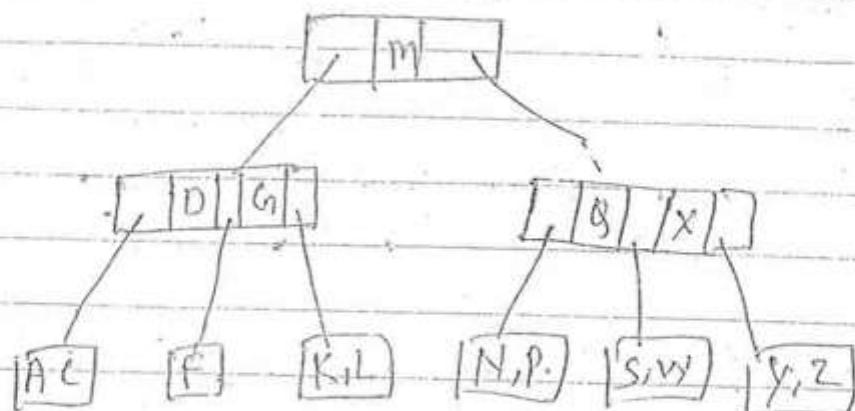


Delete - R.



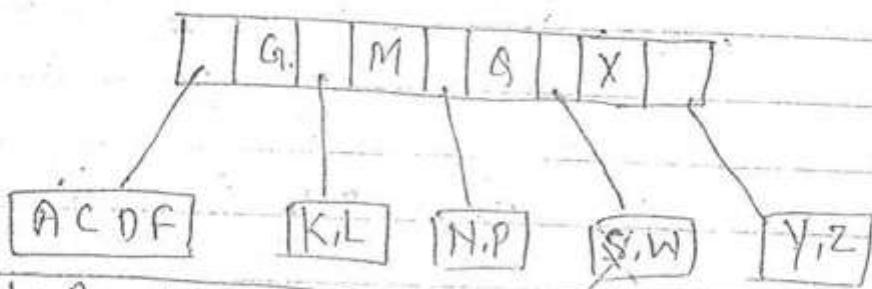
If
replace
of H
so
comb
inc

Delete E



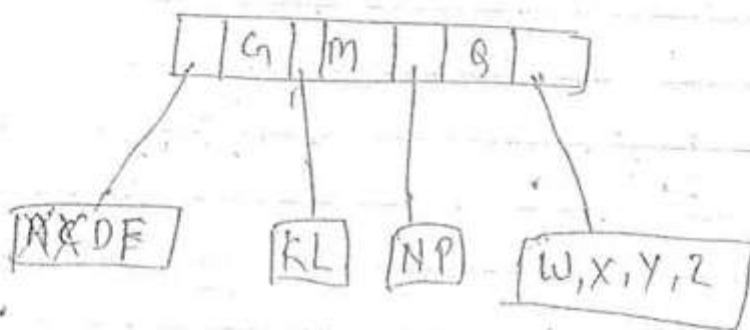
If you delete element E from the above B-tree replacement is one of the keys from any one of the sibling. but if simply it not ready. so final node M is made combining. Node combining nothing but grouping two sibling including parent.

Now the node G is underflow. No sibling is ready to help.



Delete A and C. No problem.

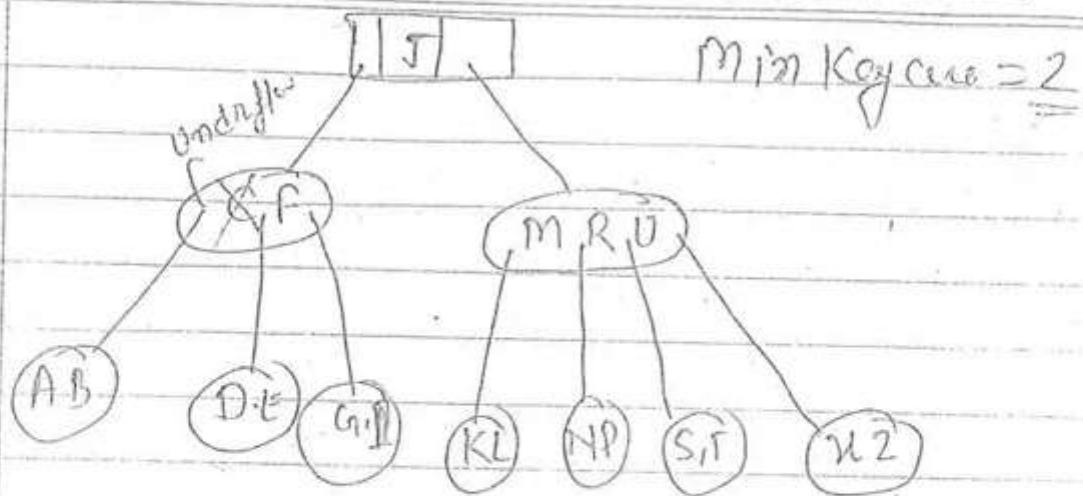
Delete S



Min^m condition should satisfied other than the root entry

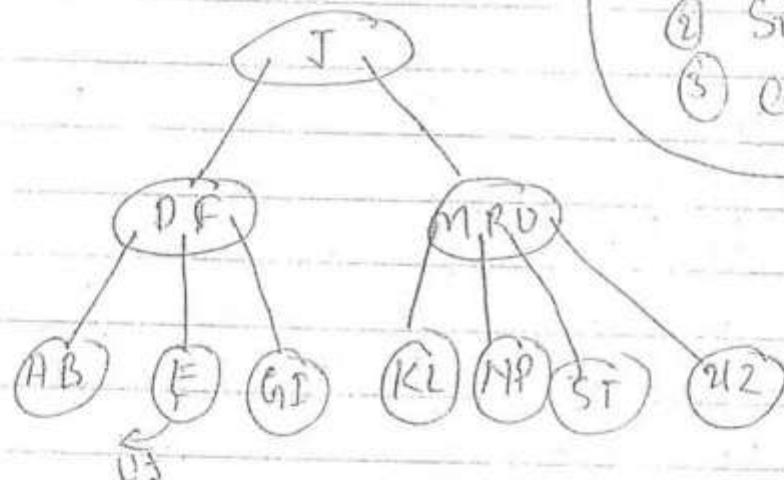
DETA	Page No.
Date:	

#9

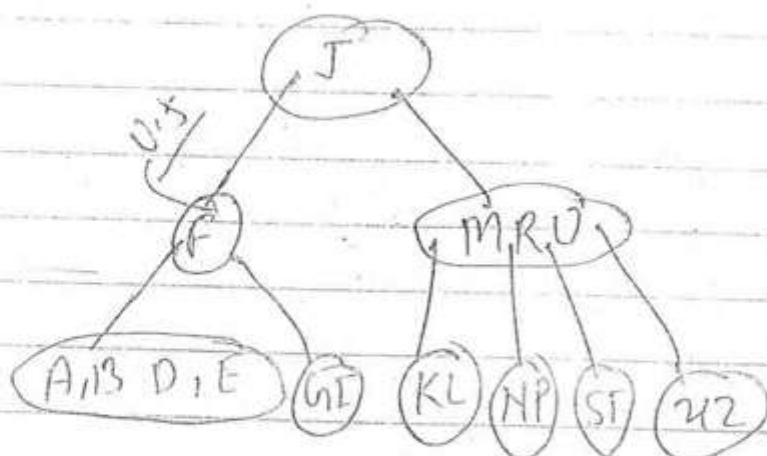


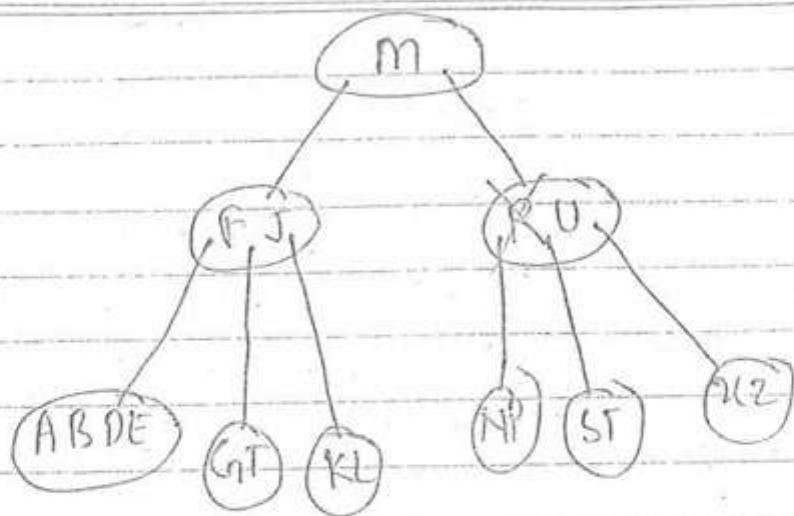
Sq^n

Delete C

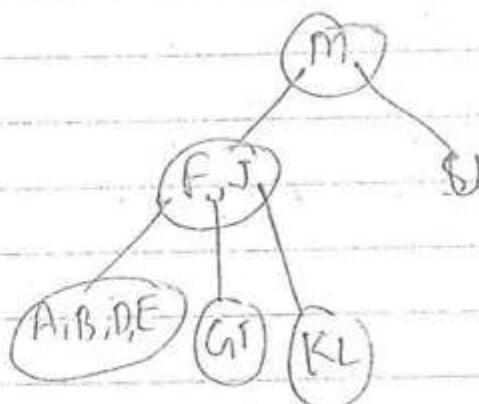
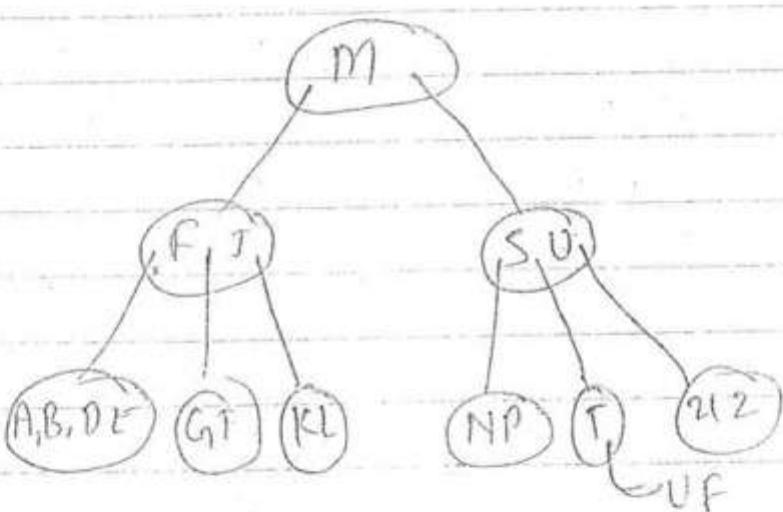


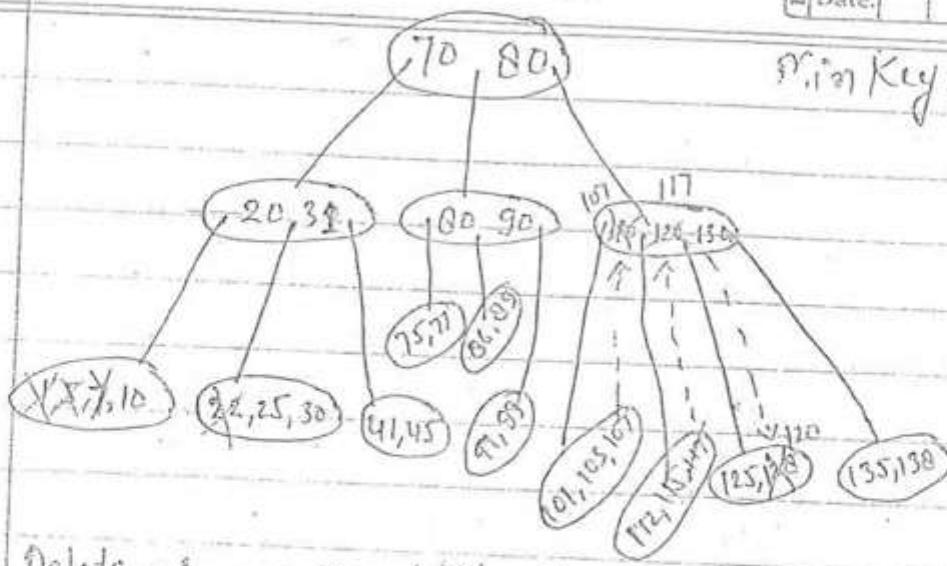
- (1) Children
- (2) Sibling
- (3) Combined





Pellet R





Min Key = 2.

Delete - 1 — ~~can~~ delete directly... no effect

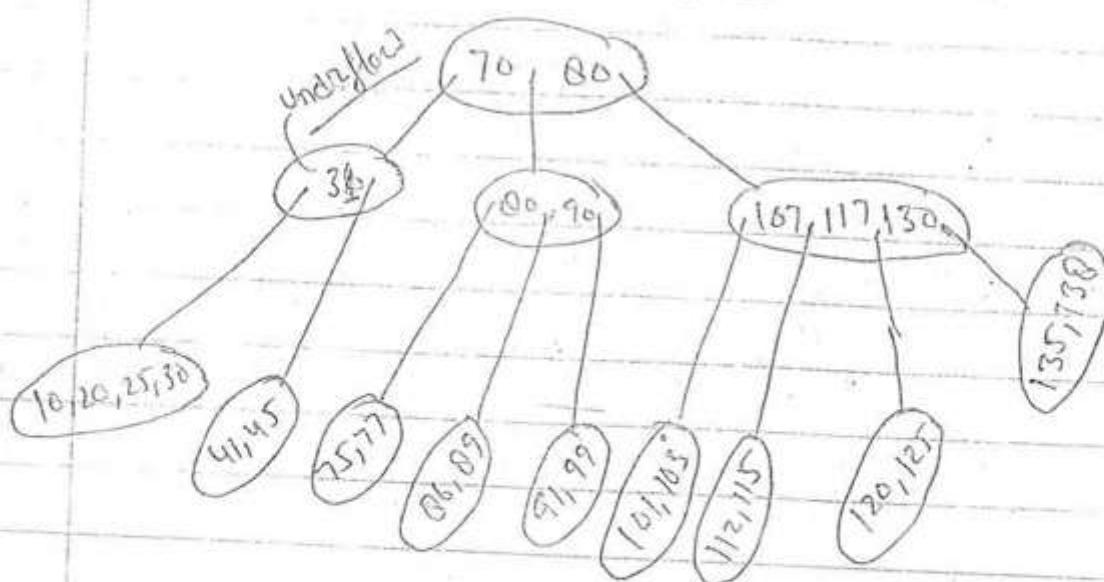
Delete - 22 — " "

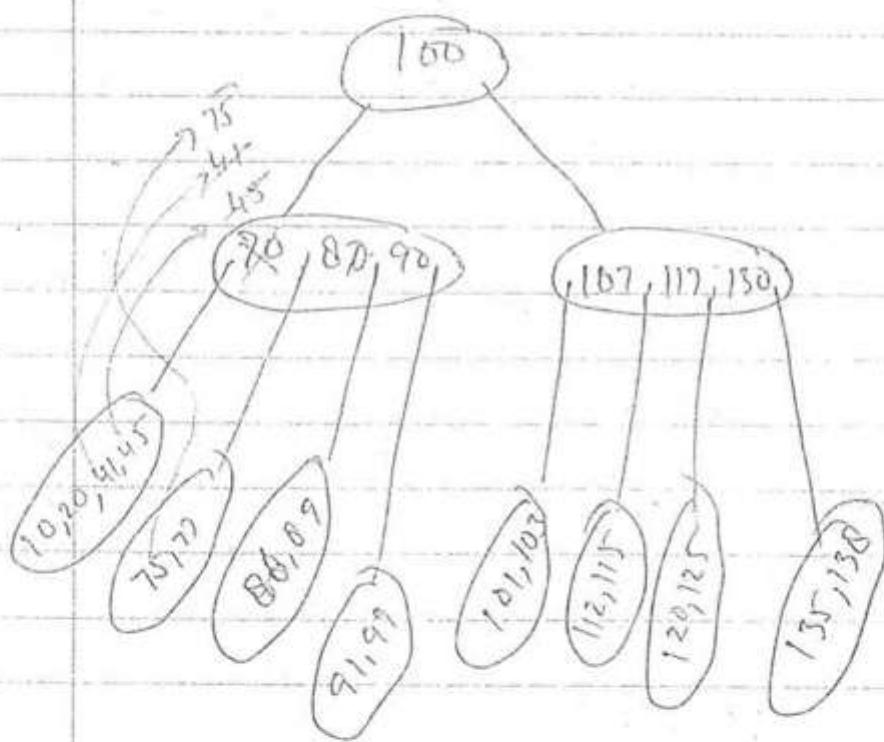
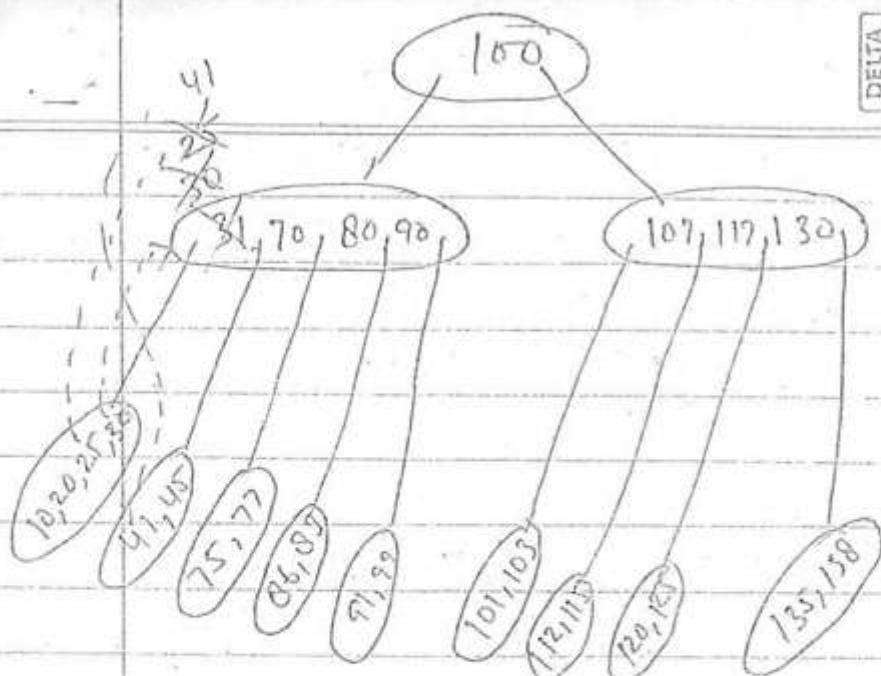
Delete - 5 — " "

Delete 120 -

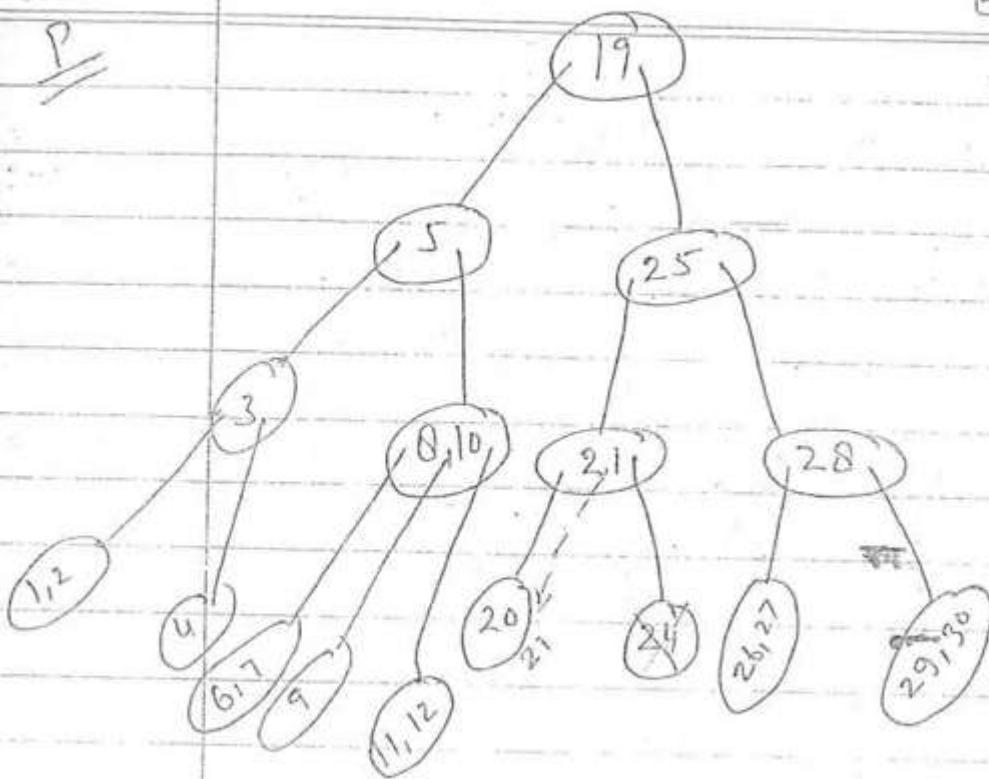
Delete 110

Delete 7 after deleting (7)

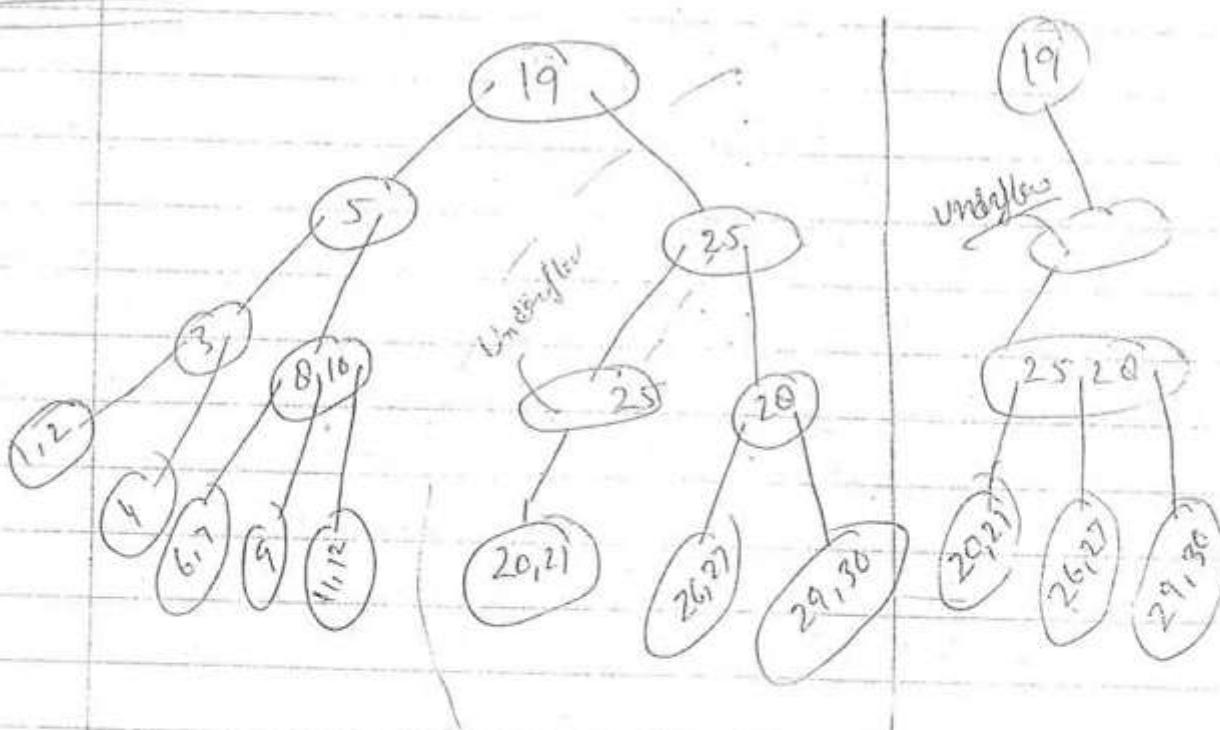


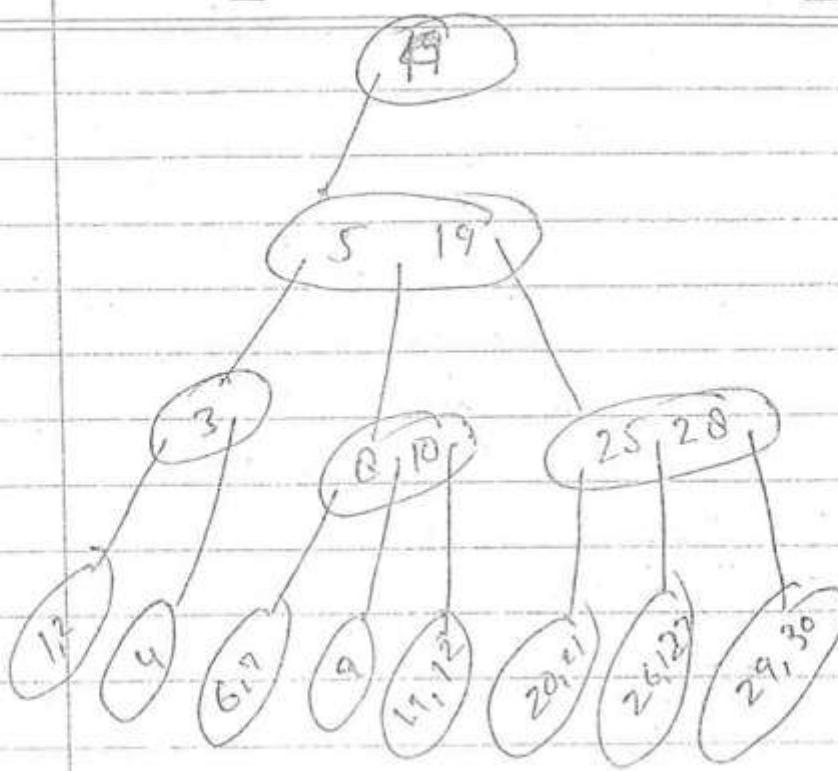


Min Key = 1



Delete 24





\swarrow B-tree of order 3, $m=32$, then.

- (1) Find the max^m key that can accommodate in level 3 B-tree.
- (2) find out max^m no. Key that can accommodate in level 3 B-tree.

~~SD^m~~

$$\text{Max}^m \text{ ch} = 32$$

$$\text{Key}^m = 31$$

$$\text{Min ch} = 16$$

$$\text{key} = 15$$

Max^m

level	nodes	key	children
1	1 (root)	31	32
2	32	32×31	$(32)^2$
3	$(32)^2$	$(32)^2 \times 31$	$(32)^3$

Min^m

level	nodes	key	children
1	1	15	16
2	16	16×15	$(16)^2$
3	(16)	$(16)^2 \times 15$	$(16)^3$

Min^M

level	nodes	key	children
1	1	1	2
2	2	2×15	2×16
3	2×16	$2 \times 16 \times 15$	$2 \times (16)^2$

35 511

Q Let T be a B-tree of order m of height h .
 If no total key = ?
 $n = ?$

Q Let T be a B-tree of order m and height h .
 if n is the number of key elements in
 it then the max^m value of n is:

$$\text{level } 1 \quad m-1 \rightarrow m$$

$$2 \quad m(m-1) \rightarrow m^2$$

$$3 \quad m^2(m-1) \rightarrow m^3$$

$$4 \quad m^3(m-1) \rightarrow m^4.$$

$$m-1 + m(m-1) + m^2(m-1) + m^3(m-1) + \dots + m^h(m-1)$$

$$m-1 [1 + m + m^2 + m^3 + \dots + m^h]$$

$$m-1 [m^0 + m^1 + m^2 + \dots + m^h]$$

$$m-1 \left[1 \left(\frac{m^{h+1} - 1}{m-1} \right) \right] \Rightarrow \boxed{m^{h+1} - 1}$$

Q

Let T be tree of order d and height h . Let $d \left\lceil \frac{m}{2} \right\rceil$ and let n be the no. of elements in T . Then which of the following is True ?

Level nodes Key Children

1	1	1	2
---	---	---	---

2	2	$2 \times (\frac{d-1}{2})$	$2 \times d$
---	---	----------------------------	--------------

3	$2 \times d$	$2 \times d(\frac{d-1}{2})$	$2d^2$
---	--------------	-----------------------------	--------

$$= 1 + 2(d-1) + 2d(d-1) + 2d^2(d-1) + \dots + 2d^{h-1}(d-1)$$

$$= 1 + 2(d-1) \left[d^0 + d^1 + d^2 + \dots + d^{h-1} \right]$$

~~$$= 1 + 2(d-1) \left[1 + \frac{(d^{h-1}-1)}{(d-1)} \right]$$~~

$$= 2d^{h-1} + 1$$

↓

$2d^{h-1}$

B⁺-Tree

DELTA	Page No. _____
	Date: _____

Q

Degree 4

4

max^m children = 4

Key = 3

min ch = 2

Key = 1

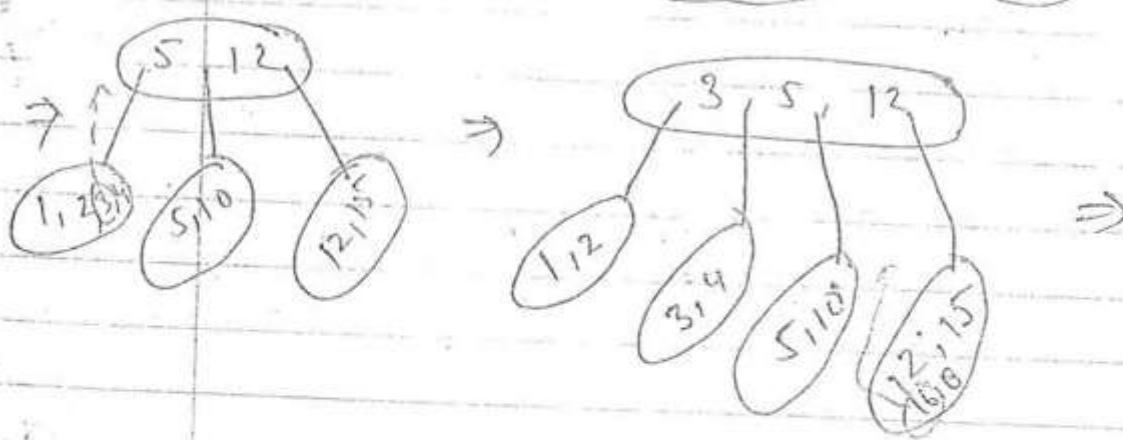
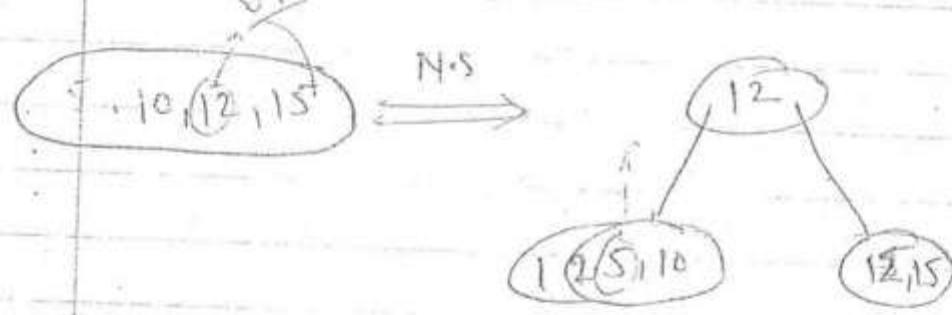
Consider B-Tree for the following

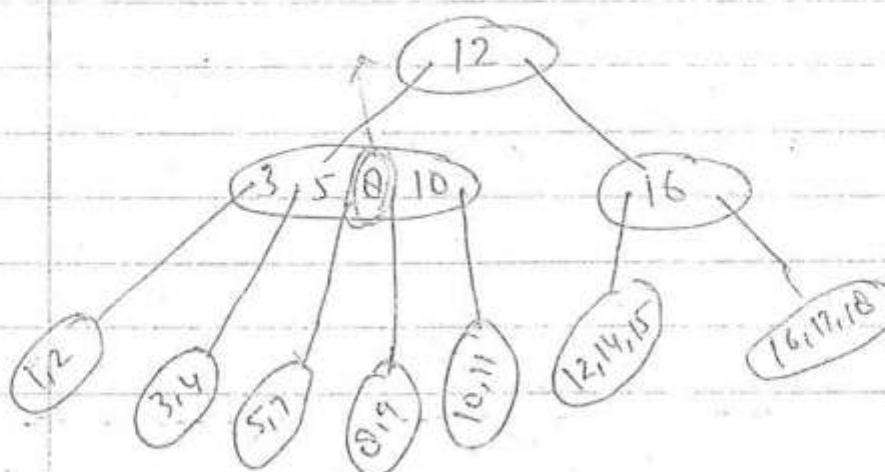
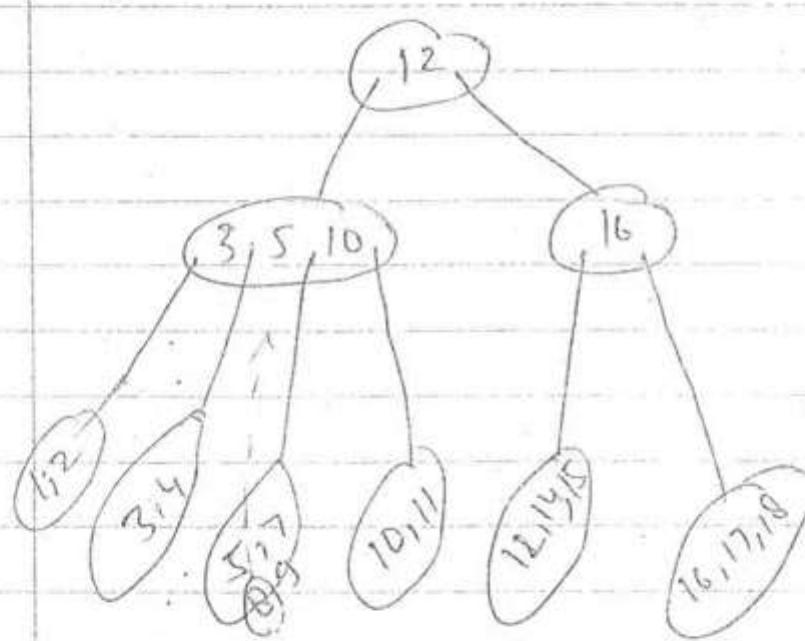
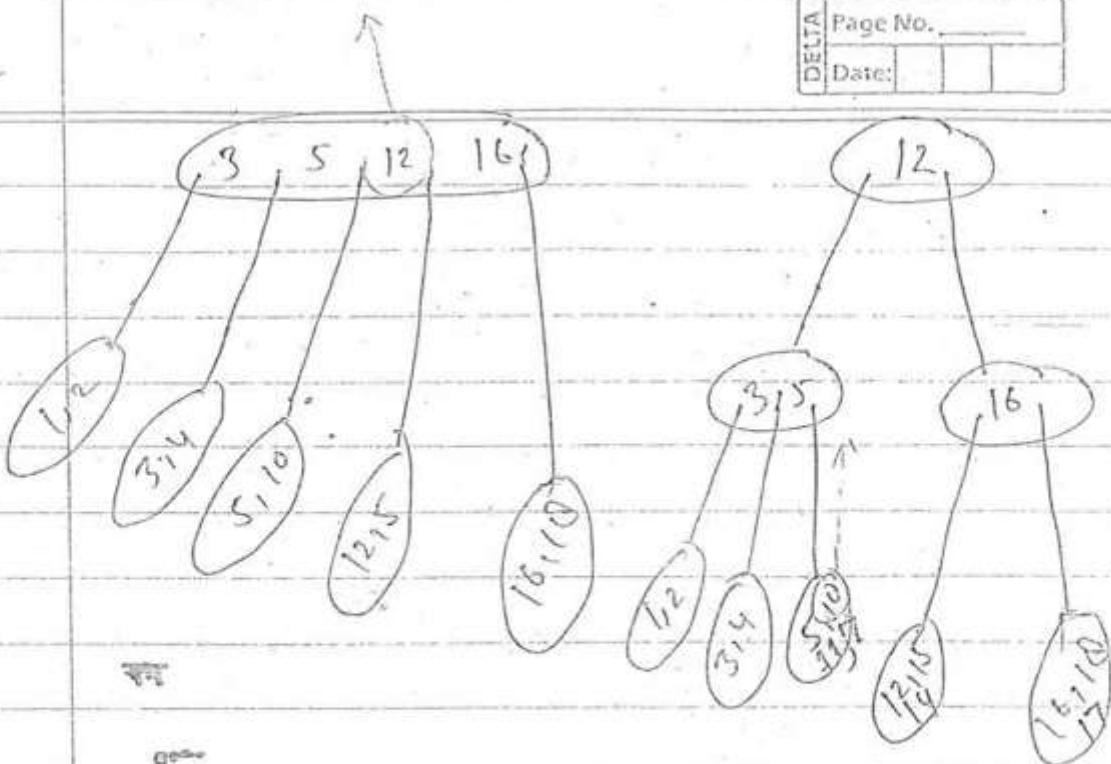
5, 10, 12, 15, 11, 2, 3, 4, 16, 18, 17, 19, 11, 7, 8, 9.

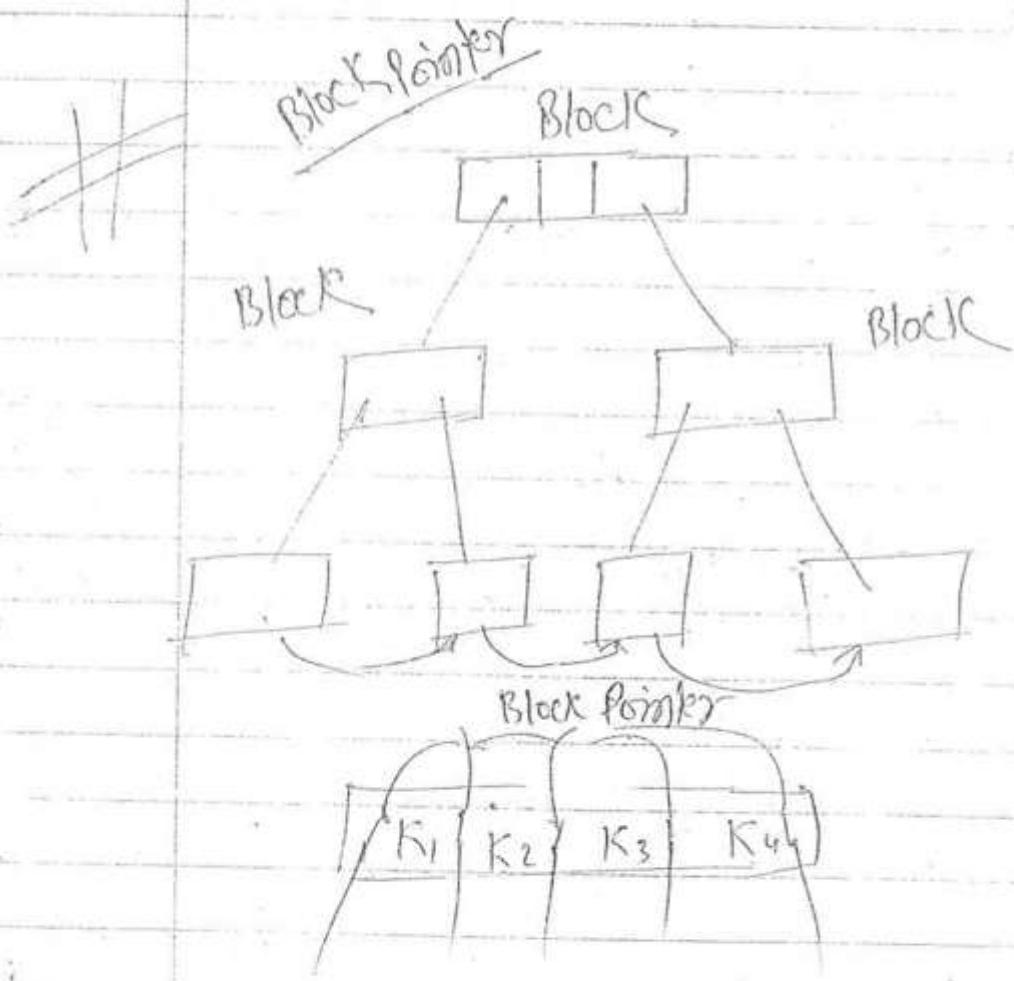
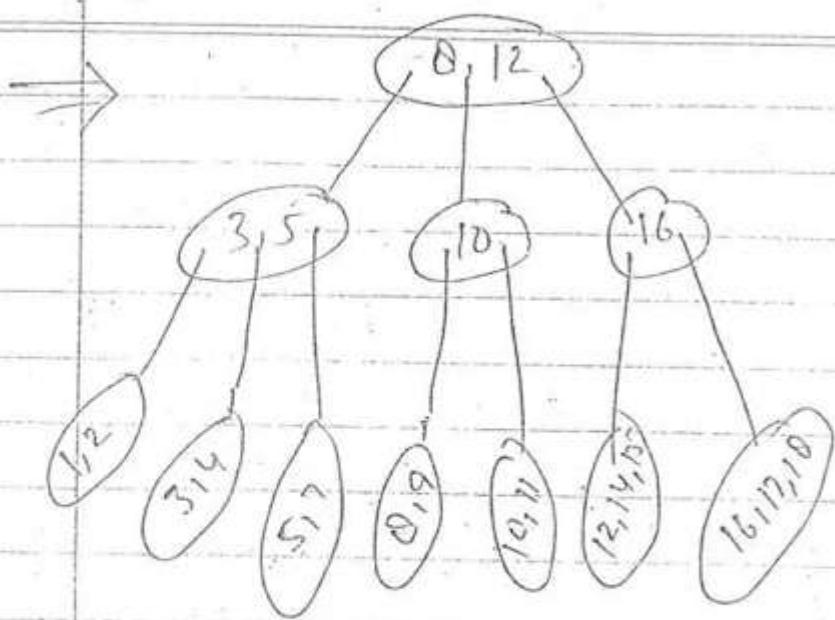
SOP

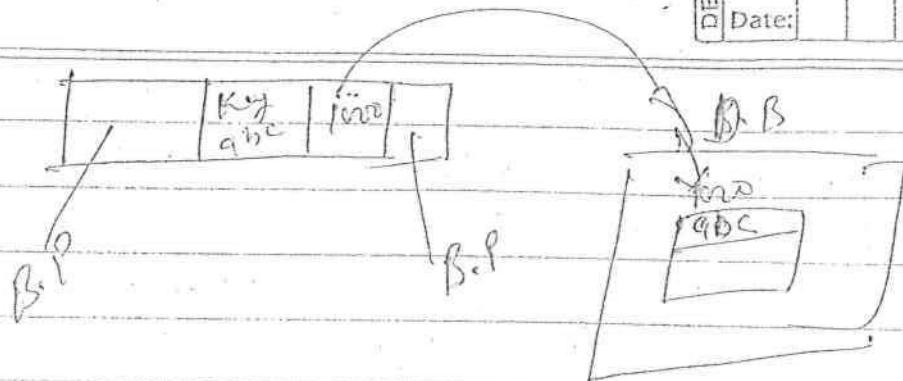
OF

N.S







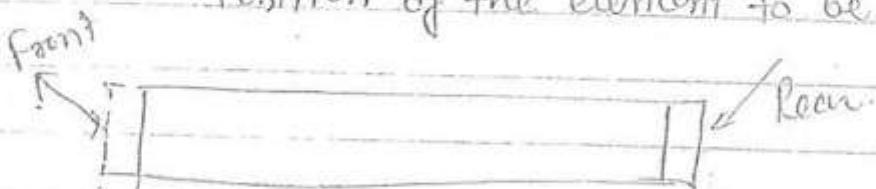


- Queue

Definition :- One side insertion and another side deletion.

: FIFO

: Front is a variable which contain Position of the element to be deleted



: Rear is a variable which contain Position of the newly inserted element.

Abstract data type of Queues.

Enque (Queue , element) = Queue .

Deque (Queue) = element .

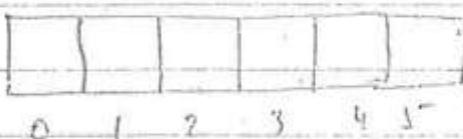
Isfull (Queue) = Boolean .

IsEmpty (Queue) = Boolean .

Enque

$N = 6$

Q.



$F = -1$

$R = -1$

(all time both null^{-1})

Void Enque(Q, N, F, R, u)

```
{
    if (R == N-1)
        Print("Queue is full");
    else
}
```

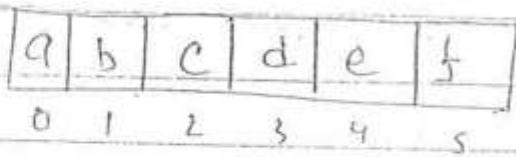
```
{
    if (R == -1)
        F = R = 0;
}
```

```
else
{
    R++;
    Q[R] = u;
}
```

Queue (It is a operation)

$N=6$

Q:



$$F=0 \\ R=5$$

Dequeue (Q, N, F, R)

```
{
    int y;
    if (F == -1)
    {
        cout(" queue is empty");
        exit(1);
    }
}
```

else

```
{
    y = Q[F];
    if (F == R) } Del deletion
    F = R - 1;
```

else

$F++ \rightarrow$ remaining deletion

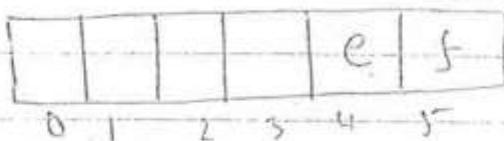
return (y);

In linear queue always

Compare

$$R > F$$

$N = 6$



$f = 4$

$R = 5$

In the above linear Queue, even though four slots are empty, we can not insert before an element because R can not go back. So Linear Queue is not efficient implementation of queue.

So we are going for circular Queue in which R can go back also.

Circular Queue

In circular queue: ($R < F$)

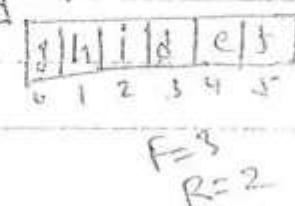
Circular Queue Insertion

Circular Enque(Q, N, F, R, X)

{

 $\Rightarrow n = 6, F = 0, R = N - 1$
 If $((R == N - 1 \text{ } \& \& \text{ } F == 0) || (R == F - 1))$ $\Rightarrow n = 6$

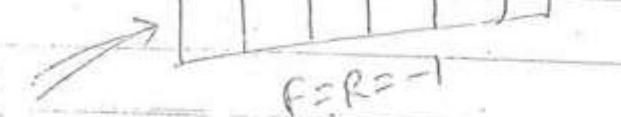
{
 Point("Queue is full");
 exit();
 }



else

{

if ($R == -1$)



$f = R = -1$

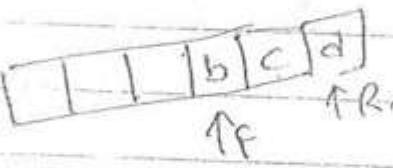
$F = R = 0$;

else

if ($R == N-1$)

$R = 0$

else $R++$



\Rightarrow



}

Circular Queue Deletion

Deletion(Q, N, F, R)

{

int y

if ($F == -1$)

{

printf ("Queue is Empty")

.exit(1);

}

else

{

$y = g[f]$

if ($f == R$)

$f = R = -1$

else

if ($f == n-1$)

$f = 0$;

else

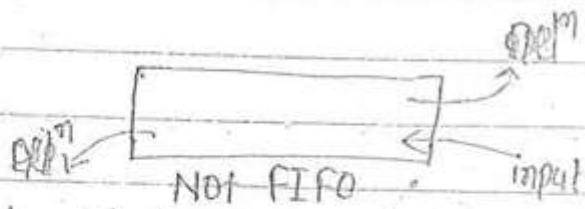
$f++$

return (y);

Types of Queue

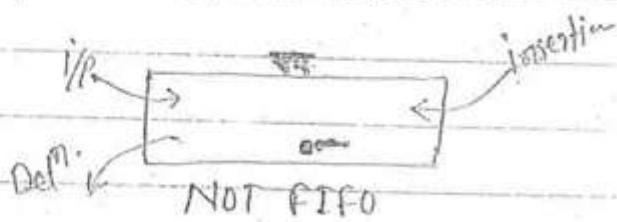
- (1) Input restricted Queue.
- (2) Output restricted Queue.
- (3) Double ended Queue.
- (4) Ascending priority queue.
- (5) Descending priority queue.

✓ Input Restricted Queue :-

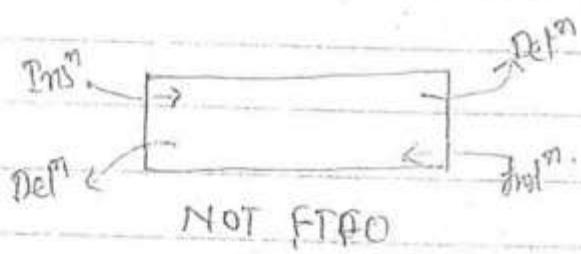


Input will be done only one side but out will be done both sides.

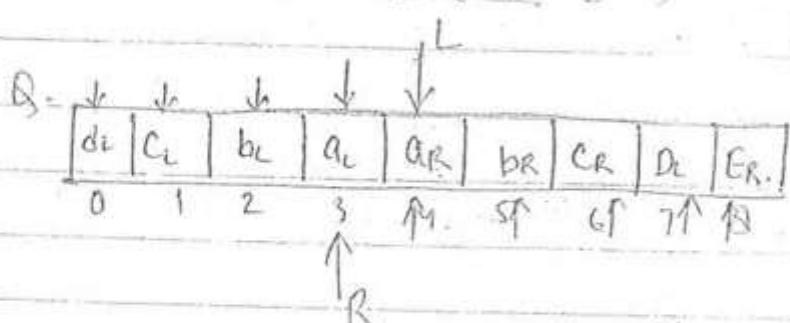
✗ Output Restricted Queue :-



✓ Double Ended Queue (DEQ) \rightarrow It is data structure.



Double Ended Queue (Operation)



$$L = \frac{M}{2}$$

$$R = \frac{M}{2} - 1$$

Insert left

if ($l == 0$) full

else

```
{
    L--
    Q[L] = u
}
```

Deletion left

if ($l == \frac{M}{2}$) empty

else

```
{
    Y = Q[L]
    L++
}
return(Y);
}
```

Insert right

if ($R == M-1$) full

else

```
{
    R++
    Q[R] = u
}
```

Deletion right

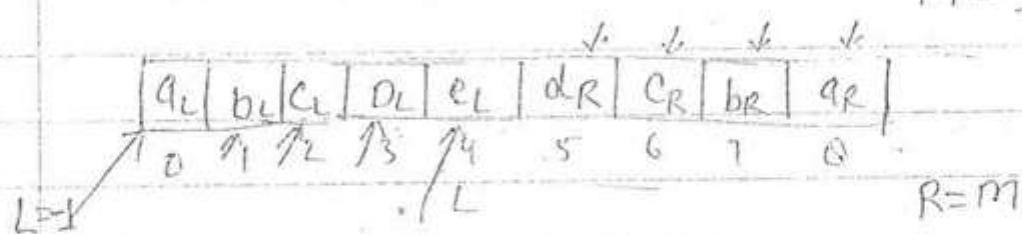
if ($R == \frac{M}{2}-1$) empty

else

```
{
    Y = Q[R]
    R--
}
return(Y);
}
```

Efficient way of implementing Double Ended Queue

$M = 9$



Insert left

if ($R == L+1$) full)

else

{
 $L++$

$Q[L] = u;$

 3

 --

Insert right

if ($R == L+1$) full)

else

{
 $R--$

$Q[R] = u;$

 3

Delete left

if ($L == -1$) empty .

else

{

$y = Q[L];$

$L--;$

 3
 return(y);

Delete right

if ($R == m$) empty .

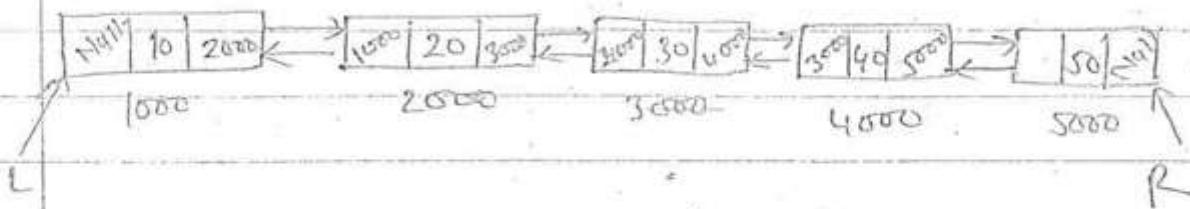
else

{

$y = Q[R];$

$R++;$

 3
 return(y);



Delete Left

① if ($L == \text{NULL}$) empty .

② if ($L == R$) free(L).
 $L = R = \text{NULL}$.

③ $L = L \rightarrow lp$

free($L \rightarrow lp$)

$L = lp = \text{NULL}$

Delete Right

① if ($R == \text{NULL}$) empty.

② if ($L == R$) free(L).
 $L = R = \text{NULL}$.

③ $R = R \rightarrow rp$

free($R \rightarrow rp$)

$R = rp = \text{NULL}$

Insert Left

$p \rightarrow rp = L$

$L = lp = p$

$L = p$

$p = \text{NULL}$

Insert Right

$R \rightarrow rp = p$;

$p \rightarrow lp = R$;

$R = p$;

$p = \text{NULL}$;

Priority Queue

Insertion and Deletion can be done at any place depending upon Priority.

Ascending Priority:- We always delete minth element each time.

50	45	15	10	25	60
0	1	2	3	4	5

each time.



Ascending Priority Queue

10, 15, 25, 45, 50, 60

(min).

Descending Priority:- We always delete maxth element each time.

50	45	15	10	25	60
0	1	2	3	4	5



60, 50, 45, 25, 15, 10

(max).

Implementing Priority Queue

- ① Unordered array { Using Array }
- ② Sorted array.
- ③ Min heap or Max heap [heap properties]
- ④ B+ tree



Unordered Array

60	50	90	5	25	30	1	15
0	1	2	3	4	5	6	7

Inserion

$O(1)$

find min in the given array of n -element and delete

finding \downarrow
min $O(n) + O(n) \Rightarrow$ Sifting

$O(n)$

Sorted Array

60, 50, 90, 5, 25, 30, 1, 15, 90

1	5	15	25	30	50	60	70	90
0	1	2	3	4	5	6	7	8

Inserion

$O(n)$

Deletion

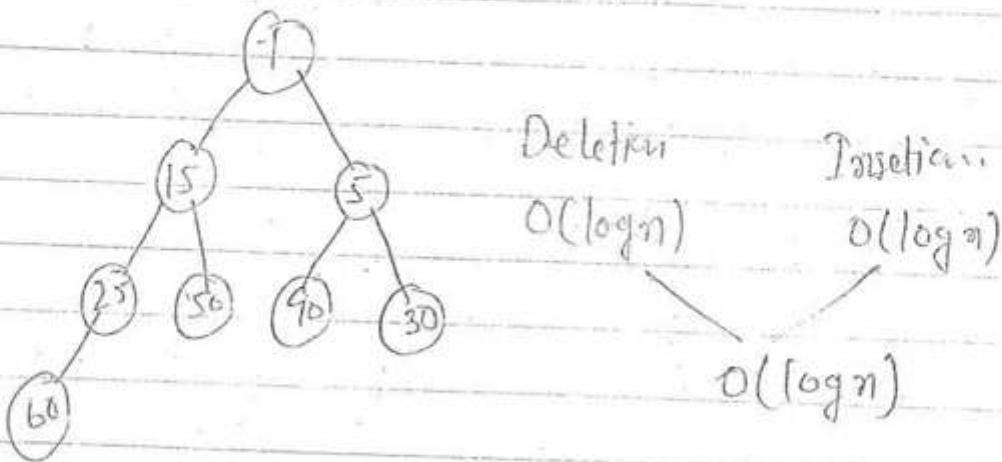
$O(1)$ [using marker]

$$O(n) + O(1) \\ \downarrow \\ O(n)$$

Priority
↑

Min heap

60, 50, 90, 5, 25, 30, 1, 15



Topic 8

B⁺-Tree

Insertion $\Rightarrow O(\log n)$

Deletion $\Rightarrow \underline{O(\log n)}$

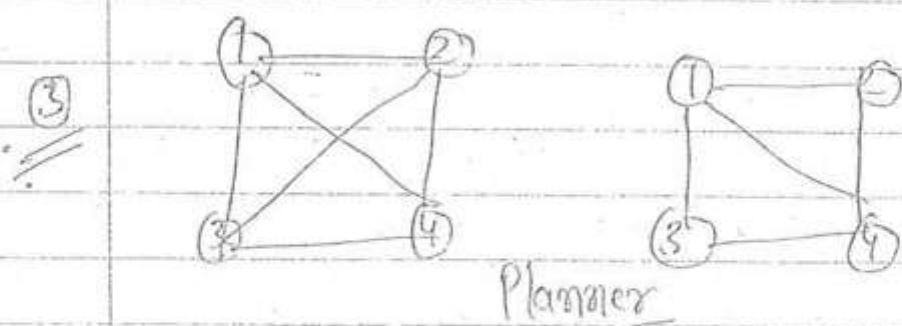
Time

Insertion $O(\log n)$

Deletion = $O(1)$

$O(\log n)$

Work Book



$$A_{4 \times 4} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}$$

$$A_{24 \times 4} = \begin{bmatrix} q_{11} & 0 & 0 & 0 \\ q_{21} & q_{22} & 0 & 0 \\ q_{31} & q_{32} & q_{33} & 0 \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}$$

$$B_{24 \times 4} = \begin{bmatrix} b_{11} & 0 & 0 & 0 \\ b_{21} & b_{22} & 0 & 0 \\ b_{31} & b_{32} & b_{33} & 0 \\ b_{41} & b_{42} & b_{43} & q_{44} \end{bmatrix}$$

$$B^T = \begin{bmatrix} b_{11} & b_{21} & b_{31} & b_{41} \\ 0 & b_{22} & b_{32} & b_{42} \\ 0 & 0 & b_{33} & b_{43} \\ 0 & 0 & 0 & b_{44} \end{bmatrix}$$

$$C = \begin{bmatrix} a_{11} & b_{11} & b_{21} & b_{31} & b_{41} \\ a_{21} & a_{22} & b_{22} & b_{32} & b_{42} \\ a_{31} & a_{32} & a_{33} & b_{33} & b_{43} \\ a_{41} & a_{42} & a_{43} & a_{44} & b_{44} \end{bmatrix}$$

$$b[3,1] = c[1,4]$$

i, 5 j; i+1

(11)

j=4, 5, 6

$$a[4+1] = a[5] \quad] \text{ side effect}$$

$$a[6] = 6$$

(12)

$$n = 1000 \Rightarrow 10 \text{ ms}$$

$$1000 \div \log_{10} 1000 \Rightarrow 10 \text{ ms}$$

$$3000 \div 10 \text{ ms} = 10 \text{ ms}$$

$$C = \frac{10}{3000} \text{ ms}$$

COUNTABLE & UNCOUNTABLE SETS.

Date - 21/07/10

1) Σ^* \rightarrow It is a discrete set,

2) $L \subseteq \Sigma^*$

Lexicographic order

\hookrightarrow every language is a grammar.

\hookrightarrow Recursive enumerable is a Turing enumerable.

\hookrightarrow below RE language in chomsky Hierarchy, all are Turing enumerable.

3) $L \subseteq 2^{\Sigma^*}$ contains every possible language, (Regular, CFL, CSL...) but it is Uncountable.

4) $L_{RE} \xrightarrow{\text{meas}} \text{Set of all RE language and It is "Countable Infinite"}$

5) \hookrightarrow Set of all Turing TMs is Countable Infinite and discrete.

for ex:- $Q = \{q_0, q_1, q_2\}$

$\Gamma \Delta = \{a, b, \square\}$

$\{L, R\}$

6) LRE, L_{fg}, L_{sg}, L_{sc} all are Countable Infinite.

(CANTOR'S DIAGONALISATION THEOREM).

If S is Countable Infinite

Then, 2^S is Uncountable Infinite.

Let $S = \{a, b\}$

→ A real number is uncountable Infinite.

→ All the language are countable.

~~My Out~~ Which one Countable Infinite and Uncountable Infinite

~~only,~~ ~~or~~ 2^{Σ^*} and L not RE is Uncountable Infinite.

→ every Turing M/L is in discrete set. and it can also perform computation.

→ Gödel's Incompleteness Theorem :- famous Mathematician that told math is incomplete

Properties of REC and RE Languages:-

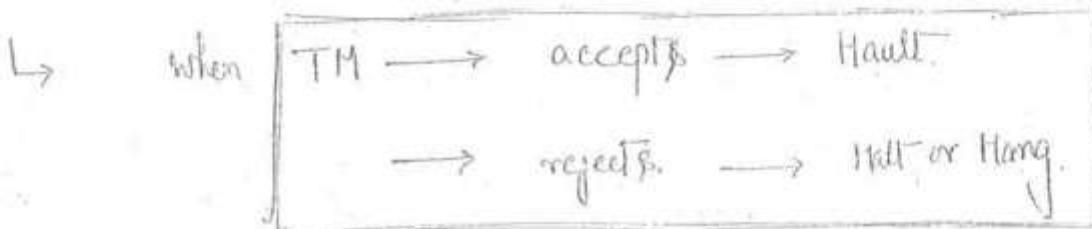
1) RE defn: A language is RE if and only if it accepts or there

exist a Turing machine.

→ It is also called as semidecidable.

2) REC defn: A language is REC if and only if there exist a
Turing machine, i.e. (TM)
which halts on every string 'w' where, $w \in \Sigma^*$.

→ It is also called as decidable.



Limited Version of RE is REC.

- 2) Both RE and REC are enumeration procedure, that run - on TM, or Turing enumerable
- 3) While both RE and REC are not membership algorithms
- 4) Only REC have a membership algorithm.

\rightarrow REC
 \rightarrow RE
 \rightarrow CSL
 \rightarrow CFL
 \rightarrow RL

$\left. \begin{array}{l} \rightarrow \text{REC} \\ \rightarrow \text{RE} \\ \rightarrow \text{CSL} \\ \rightarrow \text{CFL} \\ \rightarrow \text{RL} \end{array} \right\} \rightarrow \text{all are Turing enumerable,}$

Since, RE is enumeration procedure, but not Turing enumerable

→ If both L and L^c are Turing enumerable, then L is REC.

⇒ \hookrightarrow If both L and L^c are RE, then both are REC.

\checkmark If L is RE, then \bar{L} may or may not be RE, because of closure property.

⇒ If L is REC, then \bar{L} is REC.

\hookrightarrow If L and \bar{L} are complementary language,

Case I Both L and \bar{L} is REC i.e. not RE

Case II Both L and \bar{L}

L is RE and not REC, \bar{L} is not RE means, one of them is RE and not REC, and other is not RE.

(DECIDABILITY)

RICE'S Theorem :-

Every non-trivial question regarding RE language is Undecidable.

L_1 is RE

L_2 is RE

$L_1 \cup L_2$ is RE? \Rightarrow decidable

RE language \rightarrow Regular ?

L \rightarrow Using RICE'S theorem, above one is Undecidable.

Such question is nonTrivial question.

Non Trivial \rightarrow Undecidable.

Trivial Question } \rightarrow decidable.

L \rightarrow which RE \rightarrow Regular ?

L₂ \rightarrow which is RE \rightarrow Regular ?

} off is
Trivial question
thus decidable

particular RE \rightarrow decidable
@ arbitrary RE \rightarrow Undecidable

\rightarrow Every finite language is decidable.)

Undecidable Problems for Semidecidable Problems

- 1) Hantling Problem (HP)
- 2) Blank Tape Hantling Problem. (BTHP)
- 3) state entry problem (SEP)
- 4) Post correspondence problem. (PCP)
- 5) Modified post correspondence problem. (MPCP)
- 6) RE membership (REM)

1) HP :- $TM(M, w) \in \Sigma^*$, M will Halt or Hang ?

↳ Undecidable or Semi-decidable

2) BTHP :- Total Halting Problem

Thus, $TM(M, -) \in \Sigma^*$, M will Halt or Hang ?

↳ Undecidable or Semi-decidable

3) SEP :- $TM(M, w)$, M will enter to particular state (q_f) ?

↳ Undecidable or Semi-decidable

~~for~~ $P_1 \leq P_2$

$P_1(D) \Rightarrow P_2(D)$

$P_2(D) \Rightarrow P_1(D)$

\vdash $HP \leq BTHP$

$HP \leq SEP$

$HP \leq PCP$

$REEM \leq MPCP$
(UD) \leq (UD)

4) PCP :- If PCP Solution is U_n decidable,
when $(u_i, u_j, u_k, \dots) = (v_i, v_j, v_k, \dots)$

~~But~~ Undecidable or Semi-decidable,

↳ decidable for only Unary alphabet $\{1\} = \Sigma$

$G_1 = G_2$ i.e. equivalent of grammar is undecidable.

for eg:- $\begin{matrix} u_1 & u_2 & u_3 \\ (10, 00, 110) \end{matrix} \rightarrow \text{Set 1}$
 $\begin{matrix} v_1 & v_2 & v_3 \\ (11, 001, 011, 10) \end{matrix} \rightarrow \text{Set 2}$

PCP solution exist?

$$\begin{array}{l} u_2 \rightarrow 00 \quad u_1 \\ \quad \quad \quad 10 \\ v_2 \rightarrow 001 \quad 11 \\ \quad \quad \quad v_1 \end{array} \quad \left. \begin{array}{c} \\ \\ \end{array} \right\} \text{Therefore} \quad \begin{array}{l} u_2 \quad u_3 \\ \overline{00} \quad \overline{110} \\ 00 \quad 10 \\ v_2 \quad v_3 \end{array} \quad \left. \begin{array}{c} \\ \\ \end{array} \right\}$$

It means there exist
a PCP solution,
thus, e.undecidable,

MPCP :- A solution is said to be MPCP,

$$u_i u_j u_k = v_i v_j v_k$$

eg:- (RE but not REC) universal language binary code

A language is RE but not REC $\Rightarrow L_u = \bigcup T \{ e(T) \in L(T) \}$

eg:- A language is not RE $\Rightarrow L_d = \bigcup T \{ e(T) \notin L(T) \}$

eg of $\left. \begin{array}{c} \\ \end{array} \right\}$ diagonalisation language

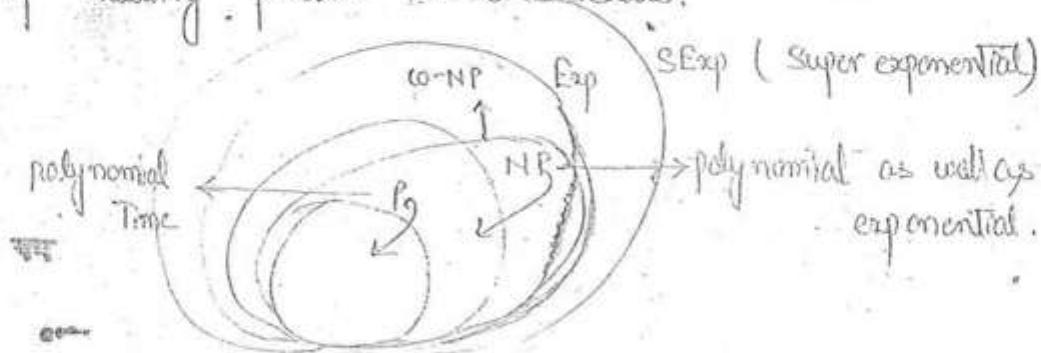
L_d decidable for only unary alphabet, ie $\Sigma = \{1\}$

Computational Complexity

↳ Decidable

⇒ In this topic only Speed matter.

↳ A Hunting problem is Undecidable.



fast speed: $\leftarrow P$ - class problem $\Rightarrow \cup$ Deterministic (n^i)

↳ It is also called as "Tractable" or "Cook-Karp Thesis".

slow speed $\leftarrow NP$ - class problem $\Rightarrow \cup$ nondeterministic (n^i)

↳ It is also called as "In Tractable".

⇒ Mainly NP is said to be exponential.)

$$\Rightarrow P \subseteq NP$$

$$\Rightarrow P \subset NP \quad ? \rightarrow \text{It is not sure | prove till now}$$

$$P = NP? \rightarrow \text{It is also not sure | prove till now}$$

↳ In IIT Bombay, it is proved that,

$$(P \neq NP)$$

~~NP Complete~~

$$\Rightarrow \text{Exp} = \bigcup_{\text{Nondeterministic } (K^n)}.$$

NP Hard

defn: If a problem (L) is NP Hard, iff, for all the languages (L') such that, $L' \in \text{NP}$

$$\& L' \leq_p L^{\text{com}}$$

→ polynomial Time reduction.

NP Complete

defn: If a problem (L) is NP Complete, if and only if, for all the language (L'),

such that $L' \in \text{NP}$

$$L' \leq_p L$$

$$\& L \in \text{NP}$$

~~✓~~ every NP Complete problem is NP Hard but converse is not True.

$$\text{i.e. } L \in \text{NPC} \Rightarrow L \in \text{NPH}$$

↳ If any 'NP' Complete problem belong to 'P' i.e. $\text{NP} \in \text{P}$, Then $P = \text{NP}$.

Closure property

↳ 'P' is closed under ($\cup, \cap, L^c, \cdot, *$)

↳ 'NP' is closed under ($\cup, \cap, \cdot, *$)

for $NP \rightarrow L^c$ is open problem.

Co-NP

$$Co-NP = \{ \bar{L} \mid L \in NP \}$$

Theorem:-2

If NP is closure under L^c then $NP = Co-NP$

Theorem 3

If $NP = Co-NP$ then $P = NP \xrightarrow{Co-NP} \text{False}$

Theorem 4

If $P = NP$ then $NP = Co-NP = P \rightarrow \text{True}$

$P \subseteq NP \cap Co-NP \rightarrow \text{True}$

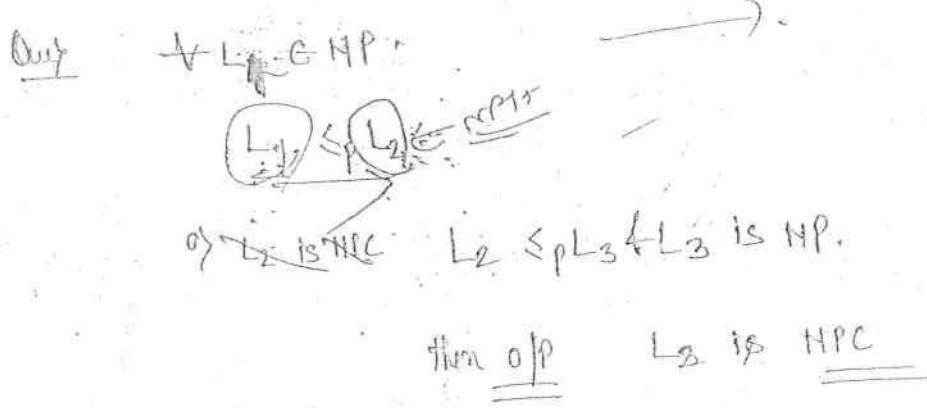
If $L_1 \leq_p L_2 \wedge L_2 \leq_p L_3 \Rightarrow L_1 \leq_p L_3$

~~closed~~

7) for, $L_1 \leq_p L_2$, then,
 if L_2 is Decidable $\xrightarrow{\text{then}}$ L_1 is decidable
 if L_2 is RE $\xrightarrow{\text{then}}$ L_1 is RE
 if L_2 is REC $\xrightarrow{\text{then}}$ L_1 is REC
 if L_2 is P $\xrightarrow{\text{then}}$ L_1 is P
 if L_2 is NP $\xrightarrow{\text{then}}$ L_1 is NP

8) for, $L_1 \leq_p L_2$,

if L_1 is NPH $\xrightarrow{\text{then}}$ L_2 is HPH
 if L_1 is NPC $\xrightarrow{\text{then}}$ L_2 is HPH
 if L_1 is NPC & L_2 is NP $\xrightarrow{\text{then}}$ L_2 is NPC

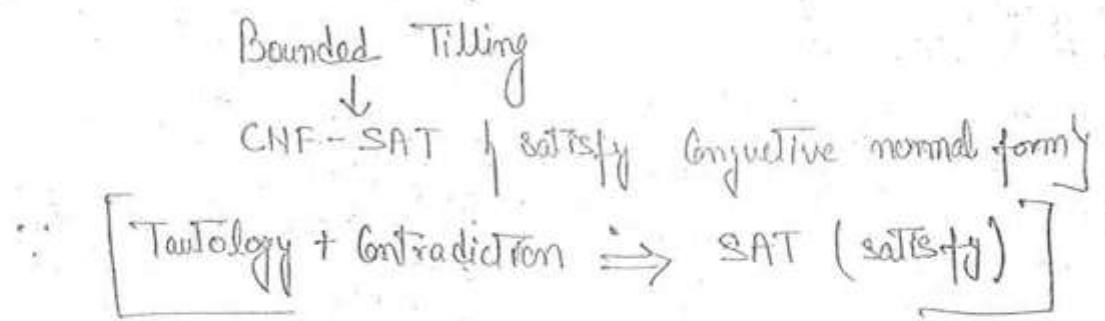


\hookrightarrow every NP Hard problem is reducible to NP Hard \rightarrow True.

But it NPC

\rightarrow false or may or
may not

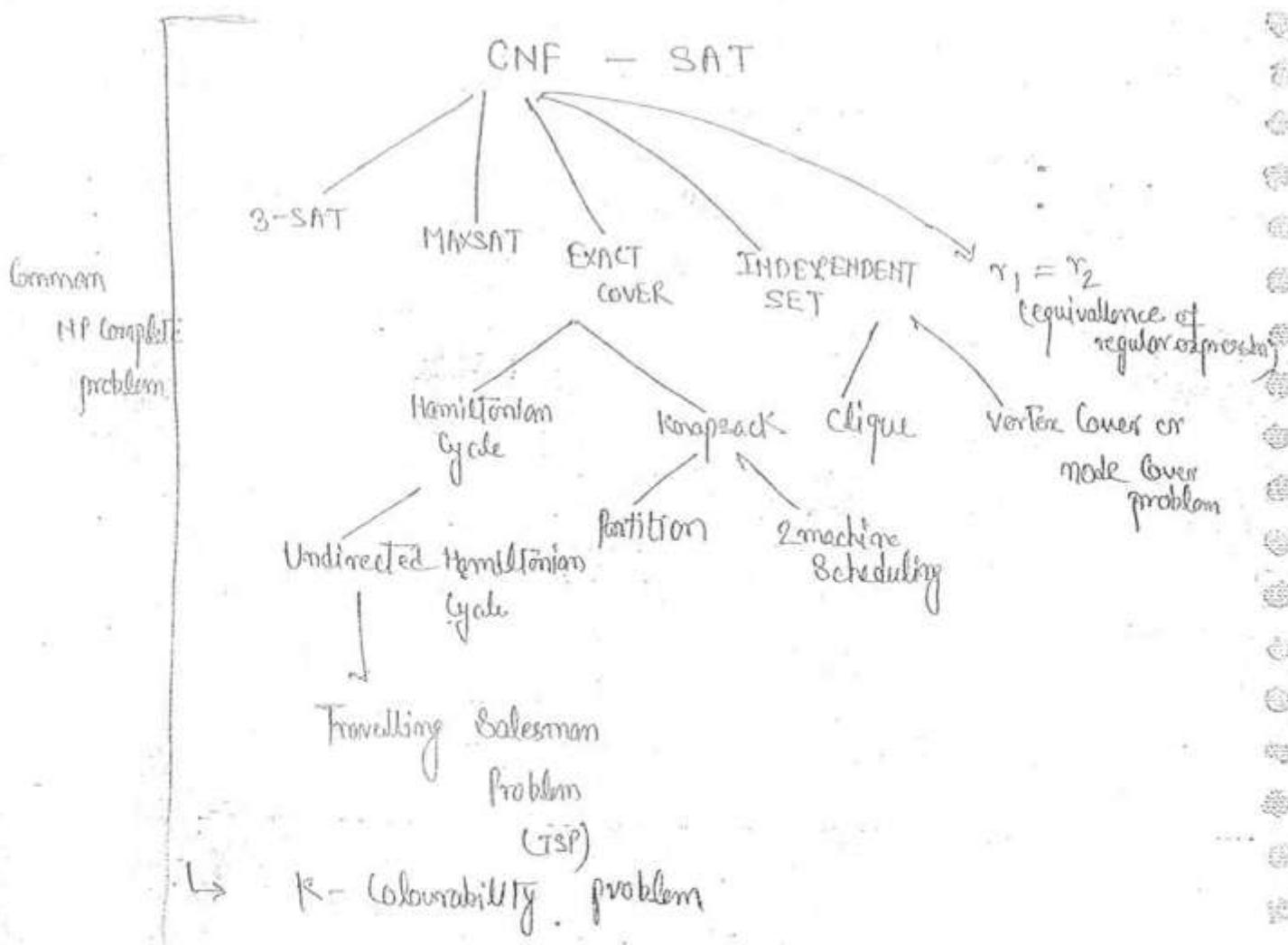
NP-Complete problem :-



0 \leftarrow Contradiction \Rightarrow UNSAT

1 \leftarrow Tautology \Rightarrow SAT

(T+C) \Rightarrow Some time SAT or some time UNSAT.



P - problem

↳ 2-SAT

↳ Unary partition

↳ shortest path.

(Dijkstra's Algo)

↳ 2-colourability

...
↳ equivalence of DFA's

↳ MINCUT

NPC

↳ 3-SAT

↳ partition (general)

↳ longest path,

↳ k-colourability,

↳ equivalence of NFA's &
regular expression,

i.e. $M_1 = M_2$.

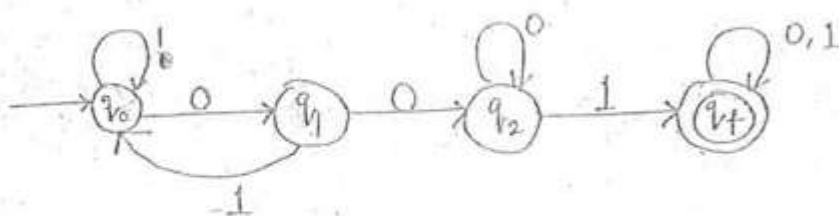
and $\eta = \tau_2$.

↳ MAXCUT.

THE END

Double

i) string that "containing the substring 001".



$$\text{R.E.} \Rightarrow (1 + \underline{001} + \underline{001})^*$$

(True or False)

ii) By Using Arden's Theorem,

$$\text{If } R = Q + RP, \Rightarrow R = QP^*$$

$$q_0 = q_0 1 + q_1 1 + \lambda \quad \text{--- (1)}$$

$$q_1 = q_0 0 \quad \text{--- (2)}$$

$$q_2 = q_1 0 + q_2 0 \quad \text{--- (3)}$$

$$q_f = q_2 1 + q_f (0+1) \quad \text{--- (4)}$$

Sol:

from eq (4)

$$\Rightarrow q_f = q_2 1 + q_f (0+1)$$

$$R = Q + RP \Rightarrow R = QP^*$$

$$\boxed{q_f \geq q_2 1 (0+1)^*}$$

from eq (3)

$$\Rightarrow q_2 = q_1 0 + q_2 0$$

$$\rightarrow \boxed{q_2 = q_1 0 0^*}$$

$$\begin{cases} q_f = q_1 0 0^* 1 (0+1)^* \\ q_2 = q_0 1 + q_1 1 + \lambda \\ \text{Since, } q_1 = q_0 0 \end{cases}$$

Application of Nfa & Dfa.

Date:- 13 Sep. 2010

- 1) Lexical Analysis h scanner plays an important role in it { }
- 2) Text editor { find, replace }
- 3) Units graph { pattern search }
- 4) Spell checker
- 5) String Match is not possible,
↳ pattern matching is done by finite automata while,
finite automata cannot done string Matching.

Application of Moore & Mealy Mc.

- ↳ Sequential Circuit design.
or
digital Circuit design.

Variation of dfa | nfa.

$$\delta(q_0, a) = \{ (q_1, b), (q_2, a) \}$$

L → left
R → Right

(Variation of dfa, nfa)

1) 2 dfa, nfa \rightarrow dfa, nfa + R or L

2) dfa, nfa + R \leftrightarrow L + R/w \equiv TM.

3) dfa, nfa + stack \rightarrow Pda

\hookrightarrow nfa + stack $\xrightarrow{\text{equivalent}}$ NPDA.

\hookrightarrow dfa + stack $\xrightarrow{\text{equivalent}}$ DPDA.

4) dfa | nfa + 2 stack $\xrightarrow{\text{equivalent}}$ TM

5) dfa | nfa + 2 counter \equiv TM

Counter \rightarrow "It is nothing but a stack."

i.e.: $y = \{ z, \}^*$ stack symbol
 \downarrow counter

6) $\underbrace{\text{dfa/nfa + 1 counter}}_{\text{less powerful.}} < \underbrace{\text{dfa/nfa + 1 stack}}_{\text{more powerful.}}$

(nfa + 1 counter is better than nfa.)

Note:

while,

nfa + 1 counter $<$ (nfa + 1 stack) \rightarrow CFL

nfa + 2 counter \equiv nfa + 2 stack

~~nfa + 1 stack~~

Note:

nfa tdfa < nfa + counter

CFL

nfa + 1 stack

TM

(counter
nfa + 2 stacks = nfa +
2 stack)

(Chapter - 2)

{Context free Languages} (INDEX)

- 1} Standard CFL's, grammar, $CFG \rightarrow L$
 $L \rightarrow CFG$
- 2} Derivation, L.M. Derivation & R.M. Derivation.
- 3} Derivation Tree, Ambiguity of Grammar &
Languages, Partial derivation tree, Sentential form.
- 4} PARSING : Membership algorithm, \xrightarrow{BPP}
 $\xrightarrow{CYK \text{ algo}}$
Complexity, Programming language & TOC, $O(n)$
Linear time Compiler, S grammar, LL grammar,
LR grammar.
- 5} Algorithm of CFG's, 1) algorithm, 2) Removal of λ production
2) Removal of unit production;
3) " " useless production;
4) " " LR $\xrightarrow{\text{Remove ambiguity}}$
5) " " L factoring $\xrightarrow{\text{Remove ambiguity}}$

6) $\text{cfg} \rightarrow \text{Cnf}$

7) $\text{cfg} \rightarrow \text{gnt} \rightarrow \text{pda.}$

In greibach normal form, length is of "m"

while, In chomsky normal form, length is of " $2n - 1$ ".

5) PDA:- Programming of DPDA & NPDA

$\boxed{\text{cfg} \rightarrow \text{Pda}}$ \Rightarrow Algorithm is used to.
Convert context free language
into pda.

6) Closure & decidability of CFL's, pumping Lemma for
CFL & Linear.

\hookrightarrow (Context free language).

\checkmark Every regular language is context free language.

while context free language is not regular language

for eg:-

$a^n b^n$, n ≥ 0

→ Grammer: $S \rightarrow aSb \mid \lambda$
↳ very restricted in nature

$w | n_a(w) = n_b(w)$
called balanced brackets
 $S \rightarrow (aSb | bSa | SS | \lambda)$

$S \rightarrow (aSb | SS | \lambda)$
properly balanced parentheses

$w w^R$
 $\rightarrow w(atb)w^R$
odd palindrome
 $S \rightarrow aSa | bSb | a\lambda$
even palindrome
 $S \rightarrow aSa | bSb | \lambda$

$\{w \mid n_a(w) = n_b(w) \text{ & } n_a(v) \geq n_b(v), \forall v \text{ which is a prefix of } w\}$

Total no. 'a' = total no. of 'b'
 prefix of 'a' must greater or equals to prefix of 'b'
 both condition must check.

~~Note:-~~ (properly balanced ~~is~~ is always a proper subset i.e. c of balanced parenthesis.)

~~✓~~ → A finite automata can not do counting. Thus its regular expression always comes under either $V^* T$ or $T^* V$.

But, $a^* b^*$ or $a^* s b^*$ is only come under Cfl because it can do counting.

$$V \rightarrow T^* V + V T^* + \overbrace{T^* V T^*}^{CFL}$$

~~✓~~ $w w^k$ is always in the form of ~~w~~ a palindrome;

$$L_1 = \{ww^R \mid w \in (a,b)^*\} \rightarrow \text{Even palindrome}$$

$$L_2 = \{w(a+b)w^R \mid w \in (a,b)^*\} \rightarrow \text{odd palindrome}$$

$$L_3 = \{w \mid w = w^R\} \rightarrow \text{All palindrome.} \\ (\text{even or odd})$$

Thus,

$$L_3 = L_1 + L_2 \text{ or } L_1 \cup L_2.$$

e.g. $a^n b^m ; n \geq 1$

grammar: $S \rightarrow aabb \mid \lambda$

e.g. $a^n b^{2n} ; n \geq 1$

grammar: $S \rightarrow aabb \mid \lambda$

e.g. $a^n b^n ; n \geq 0$

grammar: $S \rightarrow aasb \mid \lambda$

we always take condition by default $n, m \geq 0$
if not given.

e.g. $\{a^m b^n ; m \geq n \text{ and } n = 5n+1\}$

$\{a^m b^{5n+1} ; m \geq 0\}$

for e.g:- $a^n b^m ; n > m ; n, m \geq 0$

$$S \rightarrow AS_1$$

$$S_1 \rightarrow aS_1 b \mid \lambda$$

$$A \rightarrow aA \mid \lambda \rightarrow a^*$$

$S \rightarrow a^* a^n b^n \rightarrow$ This is nothing but,

$$\text{or } a^{n+m} b^n \text{ or } a^{n+m} b^m$$

Thus, $\left(a^{n+m} b^n ; n > m ; n, m \geq 0 \right)$ This is equivalent

$$(a^n b^m ; n > m ; n, m \geq 0)$$

Ques

$$S \rightarrow AS_1 \mid S_1 B$$

$$S_1 \rightarrow aS_1 b \mid \lambda$$

$$A \rightarrow aA \mid a \quad X \quad X \quad a$$

$$B \rightarrow bB \mid b \quad X \quad b \quad \lambda$$

Condition to check

Sol

$$S \rightarrow AS_1 \rightarrow A a S_1 b$$

$$\rightarrow a A a S_1 b$$

$$\rightarrow \underline{\underline{aaaaab}}$$

$$\rightarrow \underline{\underline{a^n b^m}}$$

Thus, $[a^n b^m]$ $\Downarrow n > m$ or $n < m$
 $\Downarrow n \neq m$

Ques $S \rightarrow AS_1 \rightarrow n-m = 2k$

$S_1 \rightarrow aS_1b \mid \lambda \rightarrow n=m$

$A \rightarrow aaA \mid aa$

Soln

$S \rightarrow AS_1 \rightarrow aaAS_1 \rightarrow AS_1$

$\rightarrow aaaaAS_1b \rightarrow AaAS_1b \rightarrow AaaaAS_1bb \rightarrow aaaaabbb$

$\rightarrow \underline{aaaaab}$

Final O/P $\Rightarrow (a^n b^m; n-m=2k); k \geq 1$

Ques $n_a(w) = 2n_b(w)$

$S \rightarrow aasb \mid asbb \mid ss \mid \lambda$

Possible
Combination
or order

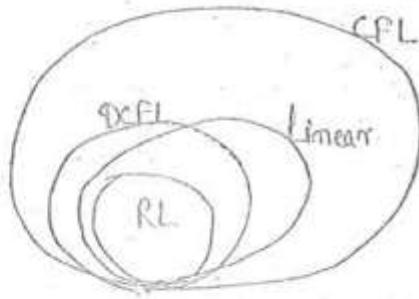
$S \rightarrow \overbrace{as} \overbrace{as} b$
as bs a
b S as a $\mid \lambda$

s can be put here (previous or after)

for eg:- $\boxed{w \# w^R} \mid w \in \{a, b\}^* \& \# \notin \{a, b\}^*$

Grammars

$S \rightarrow aSa \mid bSb \mid \#$



~~✓~~ from the above fig., every DCFL and Linear grammar contains Regular grammar and all three are under Context free language,

Linear grammar.

$$(N \rightarrow T^* + VT^* + TV + T^*VT^*)$$

- ↳ Every DPDA can do linear comparison.
- ↳ Push and Pop is not possible in DPA.
- ↳ while, Push and Pop is possible in HPDA.

for eg:- $aab \xrightarrow{\text{push}} baq \xrightarrow{\text{pop}}$

$\frac{a}{\text{push}} \rightarrow a$

$\frac{b}{\text{push}} \rightarrow b$

$\frac{a}{\text{pop}} \rightarrow b$

$\frac{b}{\text{pop}} \rightarrow a$

[If $a \rightarrow a$ comes push]
 [If $a \rightarrow b$ comes pop]

$\Rightarrow S \rightarrow aSb \mid \lambda \Rightarrow$ Linear grammar.

$\Rightarrow S \rightarrow aSb \mid bSa \mid SS \mid \lambda \Rightarrow$ Non linear grammar.
but
DCFL.

e.g.: i) wwR .

$S \rightarrow aSb \mid bSa \mid \lambda$ → It is CFL but not DCFL.

ii) $w \mid m_L(w) = n_F(w) \rightarrow$ It is not linear, but it is DCFL.

iii) $a^n b^n : n \geq 0 \rightarrow$ It is linear as well as DCFL.

e.g.: $a^m b^n : m \leq n \leq 3m$

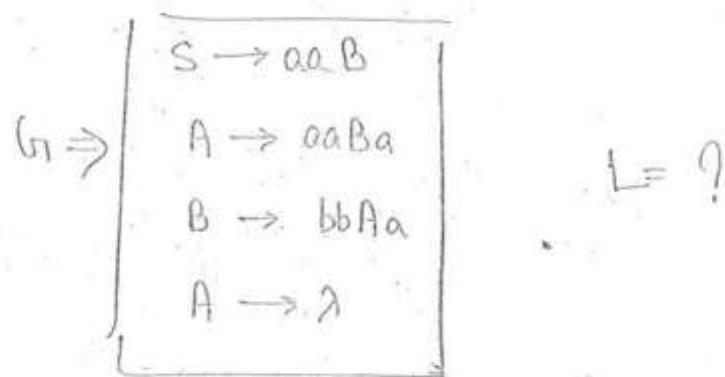
or linear CFL
It is CFL, but not DCFL.

$S \rightarrow aSb \mid aSbb \mid aSbbb \mid \lambda$

$\rightarrow S \rightarrow aSb \rightarrow aaSbb \rightarrow \underline{\underline{aab}}$

$\rightarrow S \rightarrow aSb \rightarrow aaSbbb \rightarrow \underline{\underline{aabb}}$

Ques. find a language correspond to the grammar:-



$$\rightarrow S \rightarrow aaB \rightarrow aabbAa \rightarrow aabbaaBa$$

$$\begin{array}{ll} \rightarrow aabb \\ \rightarrow aabbAa \\ \rightarrow aabbaba \\ \rightarrow aabbbaBba \\ \rightarrow aabbbaabbaba \\ \rightarrow aabbbaabbbaa \end{array} \rightarrow aabbaabbAaa$$

$$S \rightarrow xSy \mid z$$

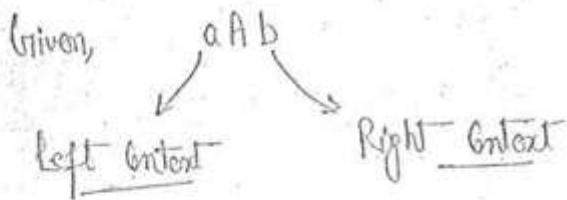
$$\rightarrow x^n z y^n \quad [S \xrightarrow{n} \text{remove by } z]$$

$$S \rightarrow ab \underbrace{(bbao)}_P^n \underbrace{bb}_Q \underbrace{a(ba)}_{QP}^n ; n \geq 0$$

$$\therefore \Rightarrow B \rightarrow (bbao)^n bba (ba)^n \quad \xrightarrow{\text{Taking a common}}$$

$$(PQ)^n P = P \cdot (QP)^n$$

$$\Rightarrow \left\{ abbb(aabb)^n (ab)^n a ; n \geq 0 \right\}$$



→ If the grammar is Context Sensitive, then we can write

The grammar as :-

$$xAy \rightarrow xA'y$$

ex:-

$$\text{for e.g.: } i) aAb \rightarrow aAab \quad : A \rightarrow Aa$$

$$ii) bAc \rightarrow bBc$$

$$iii) aA \lambda \rightarrow aAa$$

\downarrow \swarrow

left context right context

$$iv) aA \rightarrow \lambda A \lambda$$

Notes:-

↳ Every Context free Grammar is Context Sensitive grammar.

Derivation

defn: $w \in L(G)$ if and only if at least one derivation for it using production of G.

LMD \downarrow left most Derivation :- left most choice.

for e.g:- $S \rightarrow A B$ ($a a^* b^*$) $S \rightarrow AB$

$$\begin{array}{l} A \rightarrow aaA \mid \lambda \rightarrow aa^* \\ B \rightarrow bbB \mid \lambda \rightarrow b^* \end{array} \quad \begin{array}{l} \rightarrow aaAB \\ \rightarrow aaAbB \\ \rightarrow \underline{aab} \end{array}$$

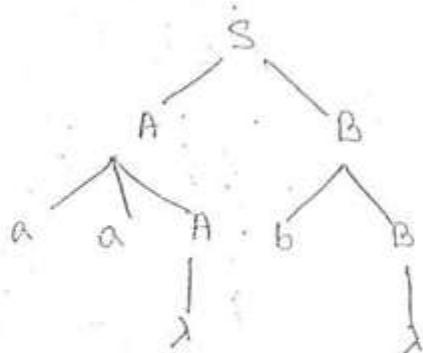
$aab \in L(G)$

L.M.D. $\Rightarrow S \Rightarrow A B \xrightarrow{\text{(left most)}} A aaAB \rightarrow aaB \rightarrow aabB \rightarrow \underline{aab}$

R.M.D. $\Rightarrow S \Rightarrow AB \xrightarrow{\text{(right most)}} Abb \rightarrow Ab \Rightarrow aaAb \rightarrow \underline{aab}$

Derivation $\Rightarrow S \Rightarrow AB \rightarrow aaAB \rightarrow aaAbB \rightarrow aabB \rightarrow \underline{aab}$
 (neither LMD over RMD)

(Derivation Tree)

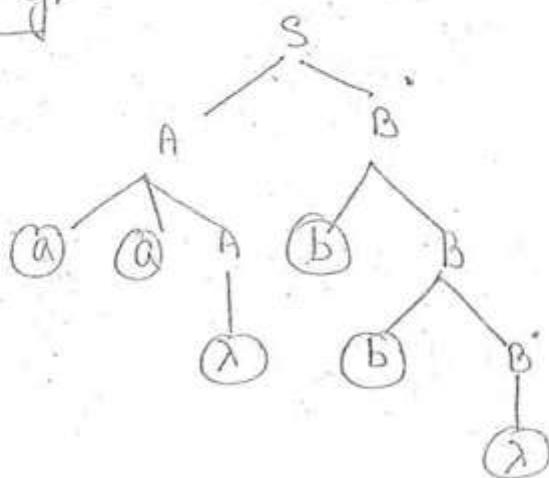


↳ A derivation tree must always be unique.

↳ In derivation tree, ambiguity is not allowed.
 ↳ It has 1 LMD and 1 RMD,

↳ In derivation tree, from left to right move, towards leaf nodes it always gives the string.

for e.g:-



String $\Rightarrow a.a \underline{bb}$

Note:

YIELD of a derivation tree always gives Sentence.

YIELD of a Partial derivation Tree always gives Sentential form.

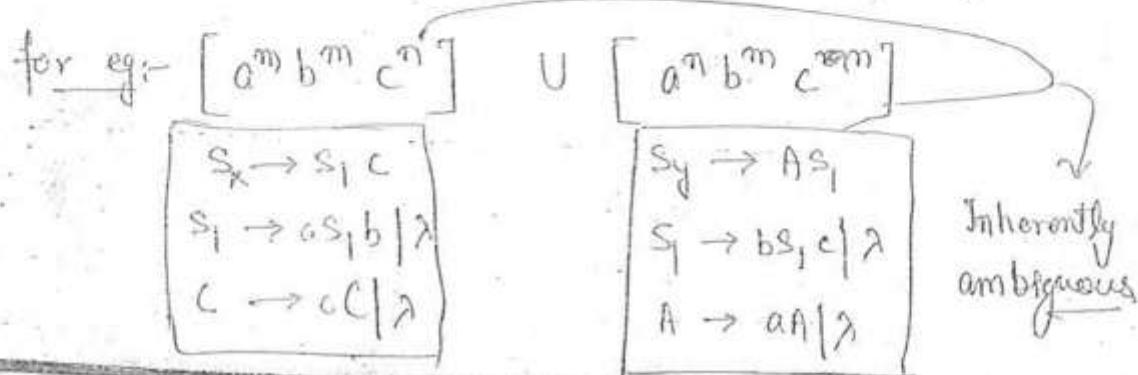
↳ Starting of derivation tree is always started from 'S'
(Root node)

↳ While Root of partial derivation tree can be any one in the tree, not necessary of 'S'
(Root node)
i.e. (Root must be anywhere in the tree)

- ↪ A grammar ' G ' is unambiguous if and only if
 $\forall w \in L(G)$, w must have exactly one derivation tree.
- ↪ A grammar ' G ' is ambiguous, if and only if
~~(at least)~~ $\exists w \in L(G)$, w must have at least two or more derivation tree.



- A grammar ' G ' is unambiguous, if and only if
 $\forall w, w \in L(G)$, w must have exactly one L.M.D. or one R.M.D.
- ↪ A language ' L ' is inherently ambiguous if and only if every grammar that generates ' L ' is ambiguous.



i.e. $\boxed{\text{Inherently ambiguous}} \rightarrow \{\text{ambiguous}\} \cup \{\text{Ambiguous}\}$

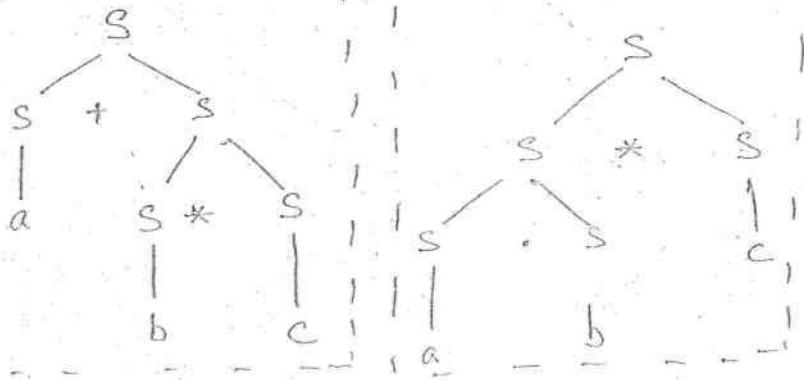
~~for eg:-~~

Date :- 14/09/10.

⇒ Ambiguous grammar :-

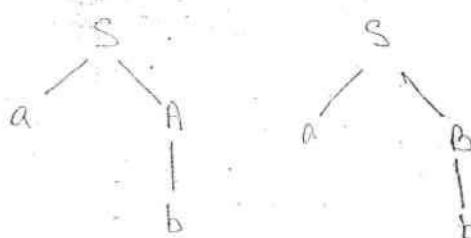
$$\text{eg:- } S \rightarrow S+S \mid S*S \mid a \mid b \mid c.$$

$$a+(b*c) = (a+b)*c$$

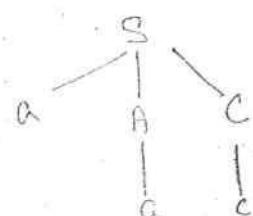
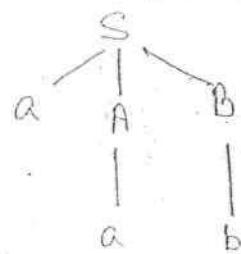


⇒ Left factoring Ambiguity :-

$$\text{eg:- } S \rightarrow aA \mid aB \quad \text{or, } S \rightarrow aAB \mid aAC \\ A \rightarrow b \quad \text{or, } S \rightarrow Aa \mid Ab. \\ B \rightarrow b$$



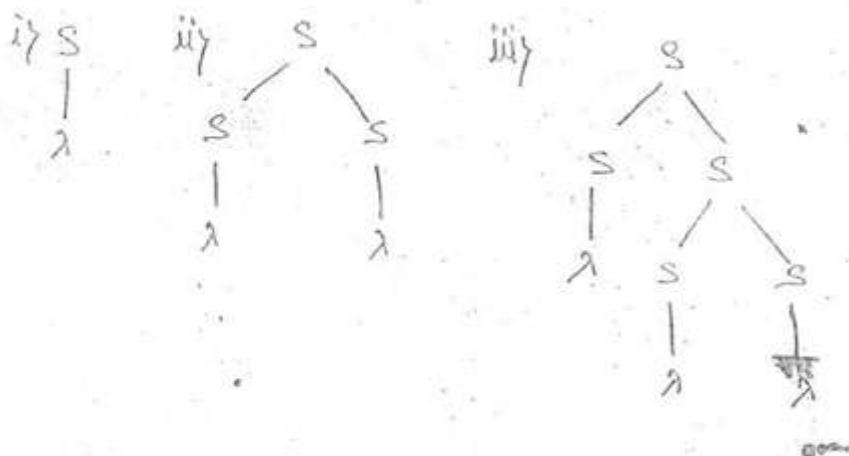
$$\begin{array}{l} A \rightarrow a \\ B \rightarrow b \\ C \rightarrow c \end{array}$$



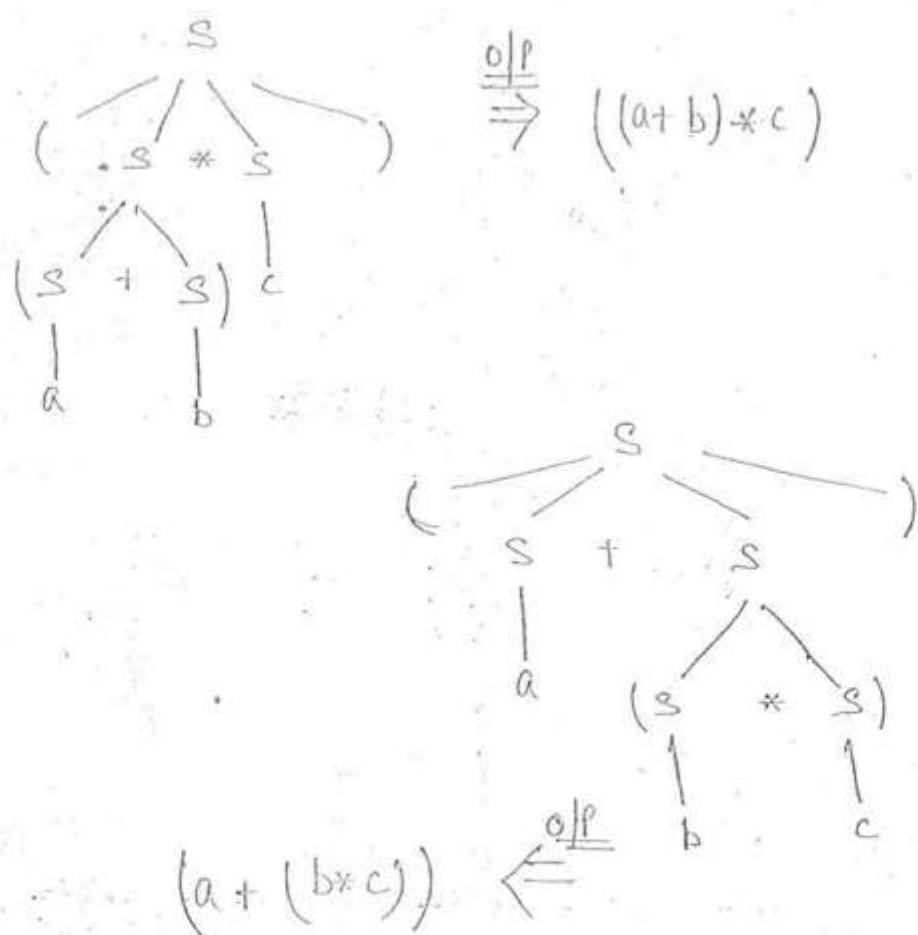
↳ If two SS is given, then there is more chance of Ambiguity.

for e.g:- $S \rightarrow a \mid b \mid ss \mid \lambda$ $\text{O/P} \Rightarrow \text{Ambiguous}$

derivation Tree



e.g ii) $S \rightarrow (S+S) \mid (S*S) \mid a \mid b \mid c$



Since $((a+b)*c) \neq (a+(b*c))$
It is not ambiguous

PARSING & Membership Algo.

↳ [Another name of derivation Tree is called "Parse Tree".]

↳ → membership algo. represents,

$$\nexists w ; w \in \Sigma^* \text{ & } w \in L(G) : - \text{ Yes or No}$$

$w \rightarrow \text{Word}$

→ [If we remove these two production from any set of production then grammar will generate expandable string.]

$$\begin{cases} S \rightarrow \lambda & \{ \text{Null production} \} \\ S \rightarrow A & \{ \text{Unit production} \} \end{cases}$$

Removal of λ .

$$L(G) = L(G')$$

$$L(G') = L(G) - \lambda$$

→ [those CFL of $L(G)$ in which for every λ -free CFL, that generates $(L(G'))$ which does contain λ or λ -free grammar, then it is called as set of production without λ (certain)]

BFP :- Brute force Parsing.

↳ It will only work after removing ' λ ' & "Unit" production.

→ Complexity of Brute force algorithm. is

$$\boxed{O(K^n)} \Rightarrow \text{In worst case.}$$

$$|P| + |P|^2 + |P|^3 + \dots + |P|^{2n}$$

It mean, it hold $\boxed{2n \text{ rounds}}$

Thus, the no. of step required :-

$$\boxed{\text{no. of steps} = |P| \frac{(|P|^{2n} - 1)}{|P| - 1}}$$

↳ (BFP is a membership algo.) $\Rightarrow |P|^{2n}$

↳ It comes under NP problem $\Rightarrow [|P|^2]^n$

$\boxed{\text{NP problem} \xrightarrow{\text{implies}} \text{exponential} + \text{polynomial} \quad K^n}$

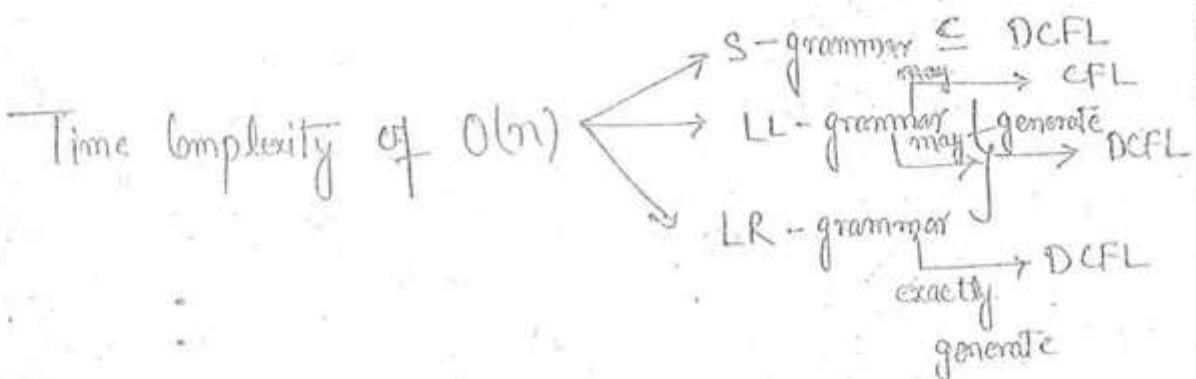
CYK Algorithm.

⇒ The best membership algorithm for CFL is O(n³)

$$\boxed{\text{Time Complexity} = O(n^3)}$$

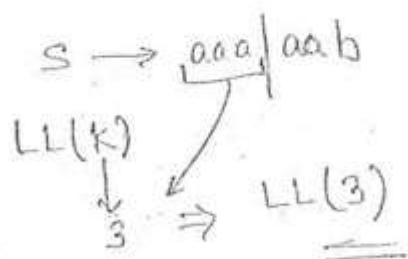
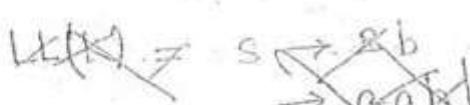
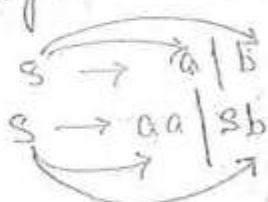
Complexity of CYK Algo $\Rightarrow O(n^3)$

(proper subset)



(S-grammar is the Super Set of Regular grammar)

(super set)
Regular grammar \subseteq S-grammar \subseteq DCFL,



$LL(k) = ?$ $LR(k) = ?$ \Rightarrow Int \Rightarrow (Read in compiler subject)

- If a grammar is ambiguous, then we do not have any LL(K) for any value of ' K '. However, a grammar is unambiguous, then it is not not mean that grammar is LL(K) for any value of ' K '.
- for LL(K) we just have to check by using
 ↘ Predictive parse table,

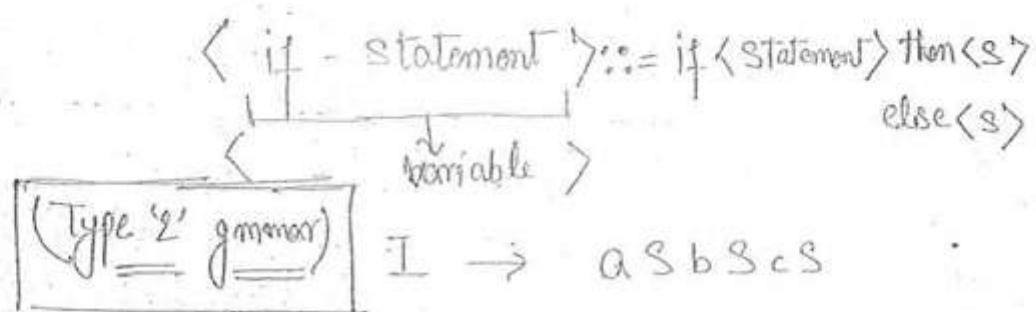
→ Similarly for LR(K),

Every LL(K) and LR(K) is unambiguous for one any some value of ' K '.

for e.g.: $S \rightarrow aSb \mid ss \mid a \mid b$

O.P.: - Neither, LL(K) nor LR(K).

(Backup - NAFR form)



$\langle \text{Term} \rangle ::= \langle \text{Term} \rangle * \langle \text{factor} \rangle$

$$\begin{array}{l} T \rightarrow T * F \\ F \rightarrow F + T \end{array} \quad \left[\begin{array}{l} T \rightarrow \text{Term} \\ F \rightarrow \text{factor} \end{array} \right]$$

↳ There is no grammar in "Semantics"

In Semantics \Rightarrow problem is only "Run Time error"

↳ Properties of LL and LR grammar

1) Every "LL" and "LR" grammar is unambiguous.

2) for LL(K) and LR(K)

Where, $K \rightarrow$ look ahead symbol,

There, LL(0) exist,

LR(K) \rightarrow "look ahead Symbol"

LL(K) \rightarrow LL(K') $K' > K$

LR(K) \rightarrow LR(K') $K' > K$

But Converse is not True.

i.e. LL(1) \rightarrow LL(2) \Rightarrow [True]

But LL(2) \rightarrow LL(1) \Rightarrow [False]

3) Every DCFL is unambiguous,

i.e. $LR(K) \xleftrightarrow{\text{given}} DCFL$,

Since, every regular language is a DCFL,

Thus, Every regular language is unambiguous,

But, $\boxed{\text{Every } LR(K) \xrightarrow{\text{can't generate}} \text{Regular language}}$

↳ Some Regular grammar is ambiguous, and some are unambiguous.]

4) for any ~~eff~~ fixed value of $K \exists$ algo. to find out if given CFG G_1 , is LL(K) or LR(K).

5) If a grammar ' G_1 ' is $LR(K)$, $K \geq 1$, \exists algo to equivalent $LR(1)$.

6) $LR(0) \neq LR(1)$

7) for any DCFL $\xrightarrow{\text{with prefix}} DEFL$ $LR(0)$
 $\xrightarrow{\text{without prefix}} LR(1)$

8) $LR(0)$ grammar exist only if and only if for DCFL with prefix property.

→ $\boxed{\text{language } L \text{ is said to be prefix property if and only if no proper prefix of } w \in L \text{ must be in } L.}$

$$L = \{aab, ab\} \\ \{a, \underline{aa}, \lambda, \underline{a}, \underline{\lambda}\} \rightarrow LR(0)$$

$L = \{aab, \lambda, a\} \rightarrow$ only $LR(1)$ not $LR(0)$.

~~✓~~ $\boxed{\lambda \text{ is a proper prefix of any word, except } \lambda}$

i). $a^n b^n; n > 0 \Rightarrow LR(0) \text{ not exist.}$

ii) $a^n b^n; n > 1 \Rightarrow LR(0) \text{ exist.}$

Q If a language (L) has no prefix property, then
 $L^{\$}$ will have prefix property.

$$L^{\$} = \{ w\$ \mid w \in L, w \in \Sigma^*, \$ \notin \Sigma \}$$

e.g.: $L = \{aab, aab\} \Rightarrow$ not proper prefix.

$$L^{\$} = \{\underline{a} \underline{a} b \$, \underline{a} \underline{a} \$\} \Rightarrow LR(0)$$

[\rightarrow If L is not a DFLR(0) then in order to make L LR(0)
make it $L\$$.]

\hookrightarrow If proper prefix, does not consist $\$$ (sign) but in order to
make it LR(0) we have to add $\$$.

[\rightarrow for every DFLR, L must produce $LR(0)$ or
 $L\$$ must produce $LR(0)$.]

ALGORITHM.

\hookrightarrow 1. Removal of Null production:

$A \rightarrow \lambda \Rightarrow$ called null (λ) production

e.g:-

$$\begin{array}{l} A \rightarrow BC \\ B \rightarrow \epsilon \lambda \\ C \rightarrow \lambda \end{array} \left\{ \begin{array}{c} A \xrightarrow{*} \lambda \\ \text{Nullable} \end{array} \right.$$

Steps:-

1) $N =$ all Nullable Variable.

2) put all nullable variable to null in
every possible permutation.

for eg:-

$$S \rightarrow ABaC$$

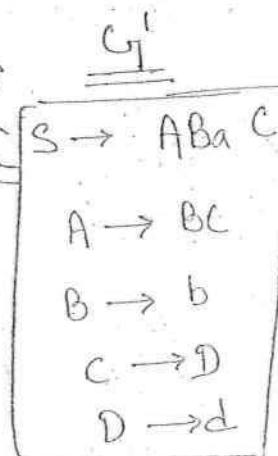
$$A \rightarrow BC$$

$$B \rightarrow b | \lambda$$

$$C \rightarrow D | \lambda$$

$$D \rightarrow d$$

$\Rightarrow G$



Step 1}

$$N = \{B, C\}$$

$$S \rightarrow BaC | AaC | ABa | aC | aB | a$$

Ba | a

Round ~~step 2~~

$$N = \{B, C, A\}$$

$$A \rightarrow c | B$$

Round 3}

$$N = \{B, C, A\}$$

(After combining we get).

$$S \rightarrow ABaC | BaC | AaC | ABa | aC | Aa | Ba | a$$

$$A \rightarrow BC | C | B$$

$$B \rightarrow b$$

$$c \rightarrow D$$

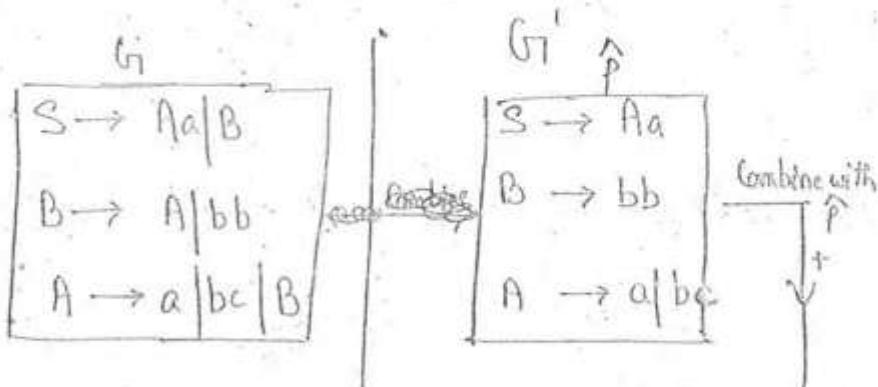
$$D \rightarrow d$$

\Rightarrow final O/P.

2. (Removal of Unit production).

↳ Always give first preference to null production then unit production, then Use less prod.

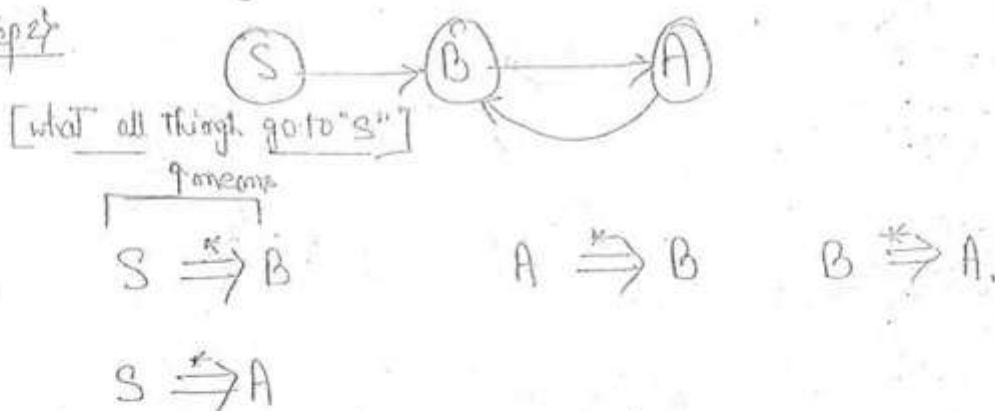
for eg:-



Step 1 make a unit production Dependency graph.

↳ make a node for every Variable,

Step 2



G_1

~~$S \rightarrow B | Aa$~~

$S \rightarrow bb$

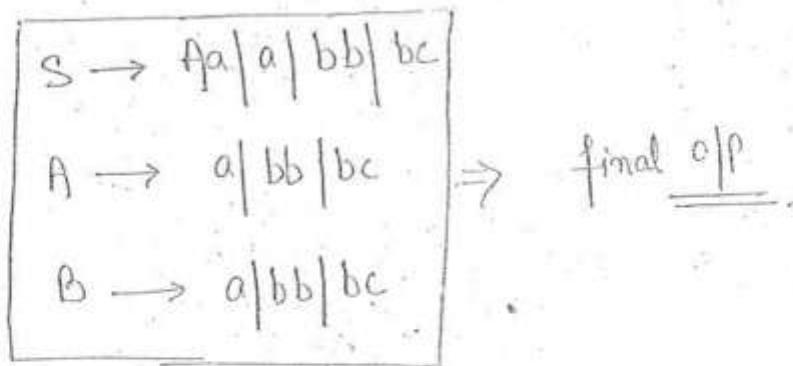
$S \rightarrow a | bc$

$A \rightarrow bb$

$B \rightarrow a | bc$

Thus,

$G^1 \Rightarrow$



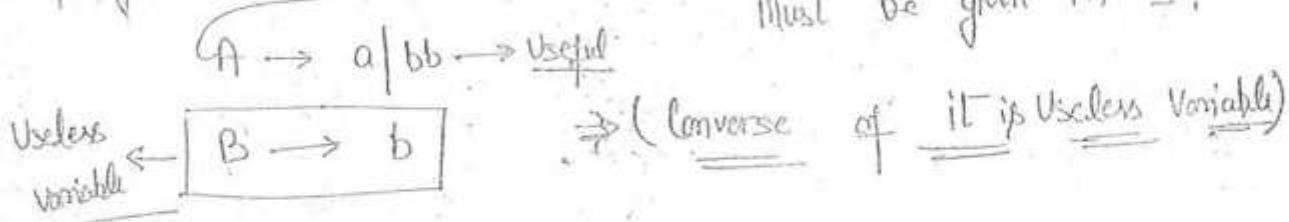
3) (Removal of Useless production)

↳ A production containing useless variable either on left or right called useless product.

\Rightarrow Any problem solve by Algorithm is "decidable"

Useful Variable :- \Rightarrow It must generate Terminal symbol.

for e.g. $S \rightarrow aA | ab \quad \Rightarrow$ If my variable specified, Variable / Terminal must be given in 'S'.



def.n : (Usefull V = { X | $s \xrightarrow{*} uxv \xrightarrow{*} w$
Variable })
 $wv \in (TUV)^*$;
 $w \in L(G)$ }

for eg:- $S \rightarrow aS | A | C$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow acb$

Step 1 We go through with ^{all} Usefull Variable and delete Variable that not generate any Terminal

Step 2 Contains Terminal (T^*) and Usefull Variable

1)

$$U = \{ A, B \}$$

$$U = \{ A, B, S \}$$

$$U = \{ A, B, S \}$$

2)

$S \rightarrow aS | A$

$A \rightarrow a$

$B \rightarrow aa$



$$U = \{ S, A \}$$

Now, $S \rightarrow aS | A$

$A \rightarrow a$

Thus, $S \rightarrow aS | a$

$\rightarrow [aa^*]_{\text{final}}$

Date:- 15 Sep, 2010

Removal of Left Recurssion.

for eg:- $G = A \rightarrow A\alpha_1 | A\alpha_2 | A\alpha_3 | \dots | A\alpha_m | B_1 | B_2 | B_3 | \dots | B_n$

$$\text{In this step, } G_1^1 = A^1 \rightarrow \alpha_1 A^1 \left| \alpha_2 A^1 \left| \alpha_3 A^1 \right| \dots \right| \alpha_n A^1 \left| \lambda \right.$$

In this we not stop with β , we stop with by ' λ '

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \cdots \mid \beta_n A'$$

$$\text{for eg:- } S \rightarrow \frac{S+S}{\alpha_1} \mid \frac{S \cdot S}{\alpha_2} \mid \frac{a}{\beta_1} \mid \frac{b}{\beta_2} \mid \frac{c}{\beta_3} \quad \left. \right\} \rightarrow G$$

$$G_1 \Rightarrow S' \rightarrow +SS' \left| *SS' \right| \lambda$$

$$S \rightarrow aS' \left| bS' \right| cS'$$

Removal of Left factoring

$$A \rightarrow \alpha_x | \alpha_y | \alpha_z$$

$$\alpha, \alpha' \in (\vee \cup \top)^*$$

$$\left. \begin{array}{c} A \rightarrow abc \mid abd \mid aba' \\ \quad \quad \quad \rightarrow aAC \mid aAB \mid aAa' \end{array} \right\}$$

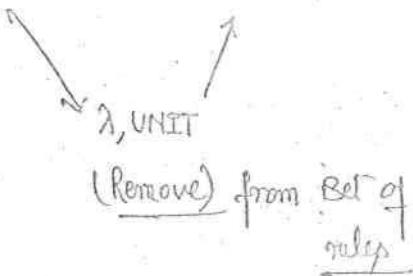
Thus, it can be written as:-

$$A \rightarrow ab A^n$$

$$A'' \rightarrow C|D|A'$$

Algorithm.

CFG \rightarrow CNF {Chomsky Normal form}



e.g:- Remove the null, Unit & ~~redundant~~ production;

$$S \rightarrow aA \mid aBB$$

$$A \rightarrow aaA \mid \lambda$$

$$B \rightarrow bB \mid bbC$$

$$C \rightarrow B$$

Step 1:

Removal of Null,

Here, only A is nullable,

$$S \rightarrow aA \mid aBB \mid a$$

$$A \rightarrow aaA \mid aa$$

$$B \rightarrow bB \mid bbC$$

$$C \rightarrow B$$

Step 2

Removal of Unit production,

$$C \rightarrow B \mid \text{Unit production}$$

$$S \rightarrow aA | aBB | a$$

$$A \rightarrow aaA | aa$$

$$B \rightarrow bB | bbC$$

$$C \rightarrow bb | BBC$$

Step 3} Removal of Useless :-

B & C are Useless,

thus, $S \rightarrow aA | a$
 $A \rightarrow aaA | aa.$

$a|aa|^*$,

thus, $S \rightarrow aaa(aa)^* | a$

final off $[a^{2n+1}; n \geq 0]$

CNF

Thus, CNF of LHS is:-

let, $D_a \rightarrow a$

thus

$S \rightarrow DaA | Da$
 $A \rightarrow DaDaA | DaDa.$

$D^1 \rightarrow DaA.$

Ques $S \rightarrow ABA$
 $A \rightarrow aa\ b$
 $B \rightarrow Ac$

$\left. \begin{array}{l} \\ \\ \end{array} \right\} \Rightarrow \text{Convert it into CNF.}$

Soln Let $D_a \rightarrow a$
 $D_b \rightarrow b$
 $D_c \rightarrow c$

Thus, $\left. \begin{array}{l} S \rightarrow ABD_a \\ A \rightarrow D_a D_a D_b \\ B \rightarrow AD_c \end{array} \right\}$

$S \rightarrow AD'$ $D' \rightarrow BD_a$ $A \rightarrow D_a D''$ $D'' \rightarrow D_a D_b$ $B \rightarrow AD_c$ $D_a \rightarrow a$ $D_b \rightarrow b$ $D_c \rightarrow c$	$\Rightarrow \text{final off.}$
---	---------------------------------

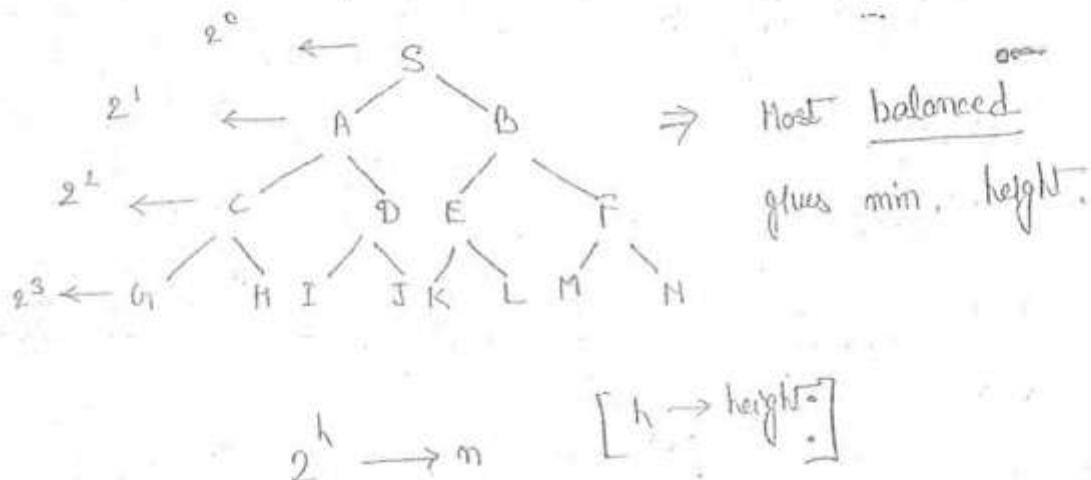
(Theorem of Chomsky Normal form)

Derivation of any string of length n , when G is
 CNF has $(2n-1)$ steps

2. Minimum height of derivation tree for a string of length 'n';
 i.e. $|w|=n$, G is CNF is $\lceil \log_2 n \rceil + 1$

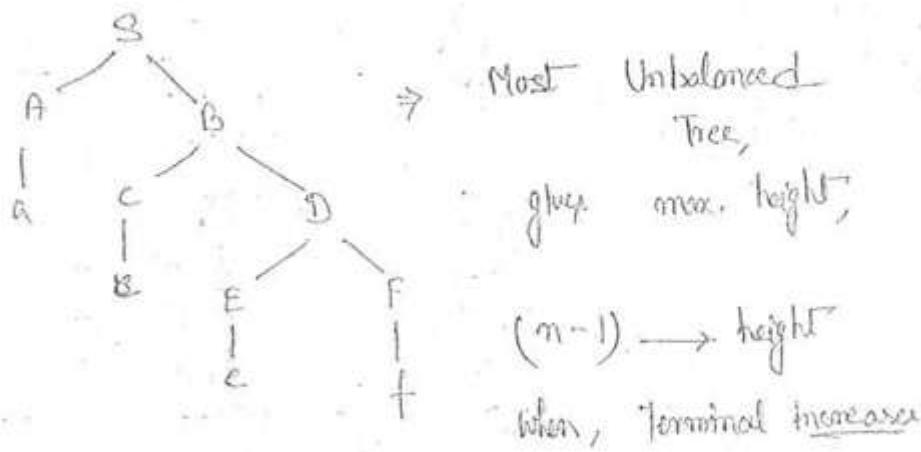
3. Maximum height of derivation tree for a string of length 'n'
 i.e. $|w|=n$, G is CNF is "n".

\Rightarrow for min. height \Rightarrow



$$2^h \rightarrow n \quad [h \rightarrow \text{height}]$$

Thus, $(h = \log_2 n)$ Thus, min. height is $\lceil \log_2 n + 1 \rceil$



$$(n-1) + 1 \Rightarrow n$$

thus, max. height is (n) .

Notes

- S-Grammar is always in Greibach normal form, while GNF not GNF.
- Every S-Grammar is LL(1) grammar, but converse is not true.

(Conversion of CFG to GNF)

GNF form :-
$$\boxed{N \rightarrow TV^* \\ \rightarrow T}$$

for eg:- $S \rightarrow aSb \mid bSa \mid \lambda$

thus, $S \rightarrow aSb \mid bSa \mid ab \mid ba$.

Note :- If the front part of set of production is Terminal
we can convert it into Variable.

for eg:- $S \rightarrow D_a S D_b \mid D_b S D_a \mid D_a D_b \mid D_b D_a$

for eg:- $S \rightarrow ABo \mid A\bar{a}$
 $A \rightarrow aB \mid bBC \mid b$ Substituting A

thus $S \rightarrow ABBo \mid BBCBo \mid bBa \mid aba \mid bBca \mid ba$

$\left[\begin{array}{l} D_a \rightarrow a \\ D_b \rightarrow b \end{array} \right]$ $\rightarrow aBbD_a \mid bBCBD_a \mid bBDD_a \mid aBD_a \mid bBCDD_a \mid bD_a$.

eg: $S \rightarrow Aa|Bb$
 $A \rightarrow aAb|aB$
 $B \rightarrow b.$

$S \rightarrow aAb|aB|b|ab|aAa$
 $A \rightarrow aAb|aB$
 $B \rightarrow b ; D_a \rightarrow a ; D_b \rightarrow b$

thus, $S \rightarrow aAD_bD_a | aBD_a | D_bD_b | aBA\underline{D_a}$

No. of steps in derivation of $|w|=m$ - length of string } in CNF
 is "m"

Complexity of CNF is $O(n)$

Machine

(Push Down Automata)

PDA \rightarrow NPDA

\hookrightarrow Every DPDA is NPDA but converse is not true.

\hookrightarrow general method to represent PDA.

$\{Q, \Sigma, \Gamma, \delta, q_0, Z, F\}$

\hookrightarrow start symbol,

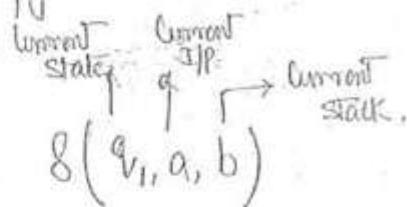
$Z \in \Gamma$ and Z can not push into stack.

$\Sigma = \{a, b\}$

[Instead 'a' there may be '0'
, " of 'b' .. . "]

$\Gamma = \{Z, a\}$

\hookrightarrow Dead Configuration is allowed in both DPDA and NPDA.



\hookrightarrow Non-deterministic is always "Search & back track"
(NPDA, NFA)
for e.g. chess game,

\Rightarrow four operation of stack :-
 1) push $\rightarrow \delta(q_0, a, b) \Rightarrow q_1(a, (q_1, ab))$

2) pop $\rightarrow \delta(q_0, a, b) = (q_1, \lambda)$

3) replace $\rightarrow \delta(q_0, a, b) = (q_1, c)$

4) do nothing $\rightarrow \delta(q_0, a, b) = (q_1, b)$

PDA program.

$a^n b^n; n > 0$

$$m_a(w) = m_b(w)$$

wwR

Y

abab|b|bb|z|

push



Here a will pop the b,
b will pop the a,

$$\delta(q_0, a, z) = (q_1, az)$$

$$\delta(q_0, b, z) = (q_0, bz)$$

$$\delta(q_0, a, a) = (q_0, aa)$$

$$\delta(q_0, b, a) = (q_0, ba)$$

$$\delta(q_0, a, b) = (q_0, ab)$$

$$\delta(q_0, b, b) = (q_0, bb)$$

$$\delta(q_0, \lambda, a) = (q_1, a)$$

$$\delta(q_0, \lambda, b) = (q_1, b)$$

$$\delta(q_0, a, z) = (q_1, az)$$

$$\delta(q_1, a, a) = (q_1, aa)$$

$$\delta(q_1, b, a) = (q_2, \lambda)$$

$$\delta(q_2, b, a) = (q_2, \lambda)$$

$$\delta(q_1, \lambda, z) = (q_0, z)$$

$$\delta(q_0, a, z) = (q_1, az)$$

$$\delta(q_0, b, z) = (q_1, bz)$$

$$\delta(q_1, a, a) = (q_1, aa)$$

$$\delta(q_1, a, b) = (q_1, \lambda)$$

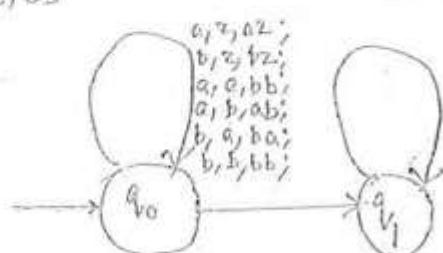
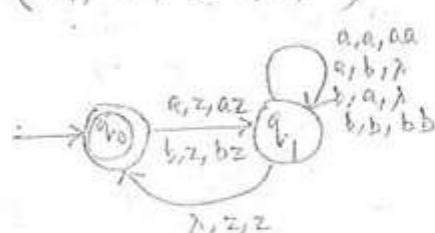
$$\delta(q_1, b, a) = (q_1, \lambda)$$

$$\delta(q_1, b, b) = (q_1, bb)$$

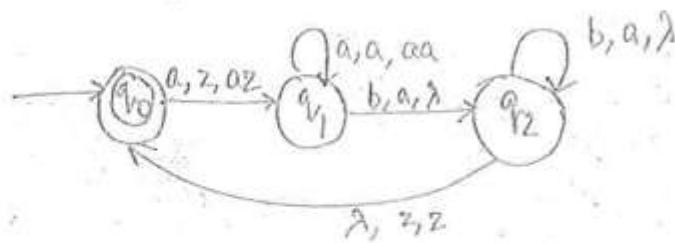
$a^n b^{2n}; n > 0$

$$\delta(q_0, a, z) = (q_1, aa.z)$$

$$\delta(q_1, a, a) = (q_1, aa.aa)$$



e.g.i) state diagram of $a^n b^n; n \geq 0$

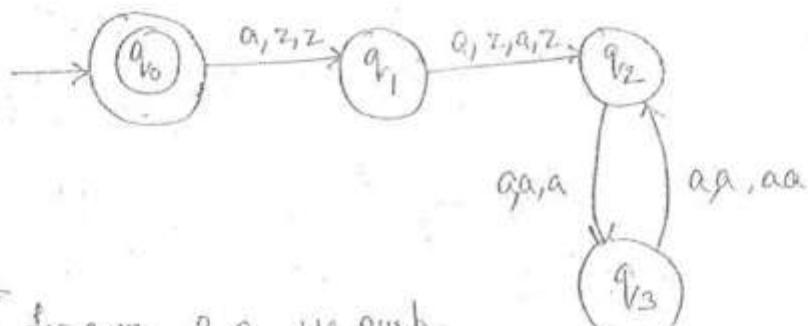


L is not regular

$L(L) \rightarrow$

$L \neq \bar{L}$ \bar{L} is regular
 $L \neq \bar{L}$ is P.E
 $L = ?$

e.g.ii) $L = \{a^{2n}, b^n; n \geq 0\}$



[for every 2a we push 1'b]

e.g.iii) $L = \{a^m b^{n+1}\} m \geq 0$

{ dec by own }

g.vi) $L = \{a^n b^n c\} n \geq 0$

vii) $L = a^m b^n; m \geq n$ { dec by own }

$m \geq n$

$m \neq n$

$m > n$

$m < n$

$\Rightarrow \{ \text{CFL} \cap \text{CFL} \} \Rightarrow \text{Not } \underline{\text{CFL}}$.

for e.g. $\stackrel{(w)}{\boxed{a^n b^m c^m}} \cap \boxed{a^n b^n} c^m = \boxed{a^n b^n c^n} \hookrightarrow \underline{\text{CSL}}$,

$$\Rightarrow \boxed{A \cap B = (A^c \cup B^c)^c} \Rightarrow \text{De Morgan's Law}$$

	CFL
→ Membership Algo.	✓
→ finite / Infinite	✓
→ Emptiness	✓

If L is CFL \rightarrow L satisfied
Pumping lemma.
while converse is not true

$\cancel{\forall} s$ is useful, the language is not empty.

$\cancel{\exists} s$ is useless, the language is empty.

↳ equivalence of PDA \rightarrow undecidable

" of DFA \rightarrow decidable

" of NFA \rightarrow decidable

" of ambiguity \rightarrow Undecidable.

$L = \Sigma^*$ \rightarrow Undecidable,

$L_1 \cap L_2 = \emptyset$ \rightarrow Undecidable.

True

false

$$\text{Def} \quad P \rightarrow q$$

$$\neg q \rightarrow \neg P$$

{ Contrapositive }

$$q \rightarrow P \quad \{ \text{Converse} \}$$

$$\neg P \rightarrow \neg q \quad \{ \text{Inverse} \}$$

ww \rightarrow Linear \rightarrow Surely Satisfy the Pumping lemma for
Linear

$S \rightarrow \lambda$ (only in context sensitive language) $\rightarrow S$ does not appear on RHS

$A \rightarrow \lambda$ (never in type 1 language) (may or may not)

\Rightarrow Two different grammar may generate same languages.

$$G_1' \Rightarrow S \rightarrow SS \mid aS \mid bS \mid a \mid b$$

$$G_1 \Rightarrow S \rightarrow aS \mid bS \mid a \mid b$$

$$\boxed{L(G_1') = L(G_1)}$$



$$\Rightarrow L(G_1) \text{ is defined}, \quad w \in \Sigma^* \quad \boxed{S \xrightarrow{*} w}$$

$\Rightarrow S \xrightarrow{*} \alpha$ α is called as Sentential form

Out ($a^m b^n$; $m \leq n \leq 3n$)

Grammer $\Rightarrow S \rightarrow aSb \mid aSbb \mid aSbbb \mid \lambda$

$$\delta(q_0, 0, z) = (q_1, az) \quad (q_1, aa z), (q_1, aaa z)$$

$$\delta(q_1, a, a) = (q_1, aa) \quad (q_1, aaa), (q_1, aaag)$$

(Conversion of CFG to Machine).

for eg:- $CFG \Rightarrow NPDA$



eg:-
 $S \rightarrow aA$
 $A \rightarrow aABC \mid bB \mid a$
 $B \rightarrow b$
 $C \rightarrow c$

∴ The above grammer is already in the form of
Greibach Normal form,
$$[V \rightarrow TV^*]$$
$$[\rightarrow T^*]$$

Here, Variable portion kept in the stack.

and Terminal kept in the Tape.

$S \Rightarrow aA \Rightarrow abb \Rightarrow \underline{\underline{abb}}$

| a | b | b | λ |

Tape

Stack

$$\delta(q_0, \lambda, z) = \{(q_1, S_2)\}$$

$$\delta(q_1, \lambda, z) = \{(q_1, z)\}$$

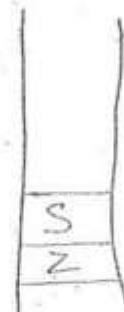
$$\delta(q_1, a, S) = \{(q_1, A)\}$$

$$\delta(q_1, b, A) = \{(q_1, nS), (q_1, \lambda)\} \rightarrow \text{represent NPDA}$$

$$\delta(q_1, b, A) = \{(q_1, B)\}$$

$$\delta(q_1, b, B) = \{(q_1, \lambda)\}$$

$$\delta(q_1, c, C) = \{(q_1, \lambda)\}$$



→ If the grammar has in production maximum Commands
one $(n+2)$.

↳ Any CFG, accepted by machine (NDPA | DPDA) is
of at most 3 states.

ANSWER

(Question to be ask):

→ a) Type '0' grammar / RE grammar can generate Regular grammar

 i) \rightarrow True / false, ii) Pumping Lemma \Rightarrow Ans?

→ Every CFG generate CSL (that are under CFL)

→ every regular grammar is a linear grammar $\left\{ \begin{array}{l} (\text{subset}) \\ (\text{total}) \end{array} \right\}$

then, RE grammar or Type '0' grammar can also generate Regular grammar
(True / False)

→ length of string is $|\Sigma^n|$ and the no. of string of length Σ^n
 i.e. $|\Sigma|^n$ True / False

→ $a^n b^n c^n$

Set of productions :-

$S \rightarrow aSBC \mid aBC$

$cB \rightarrow BC$,

$0B \rightarrow ab$,

$bb \rightarrow bb$,

$bC \rightarrow bc$,

$cc \rightarrow cc$

$\left. \begin{array}{l} \Rightarrow \text{equivalent} \\ \text{to CSL} \\ \text{but not CSL} \\ \text{due, to } CB \rightarrow BC \end{array} \right\}$

True / False

Date: 16/09/10

↳ at least 1 as and 2 and 2 b's.

$$\begin{aligned} & (a+b)^* a (a+b)^* \oplus b (a+b)^* \oplus b (a+b)^* \\ & \quad + \\ & (a+b)^* b (a+b)^* \oplus a (a+b)^* \oplus b (a+b)^* \\ & \quad + \\ & (a+b)^* b (a+b)^* \oplus b (a+b)^* \oplus a (a+b)^* \end{aligned} \quad \left\{ \begin{array}{l} \\ \\ \end{array} \right.$$

$B \xrightarrow{a}$

$B \xrightarrow{a} C$
 $B \xrightarrow{b} -$

$C \xrightarrow{a} -$
 $B \xrightarrow{a} C$
 $C \xrightarrow{b} B$
 $B \xrightarrow{b} -$

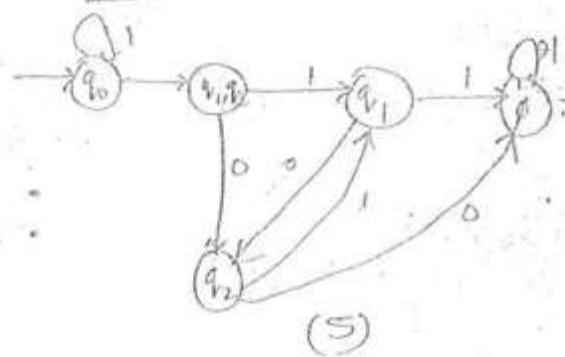
$(00)^* 0 \rightarrow \underline{\text{odd group}}$

$00^* 0 \rightarrow \underline{\text{even group}}$

$(00)^* 0 \neq 00^* 0$

$(00)^* 0 = \underline{\underline{000}}^*$

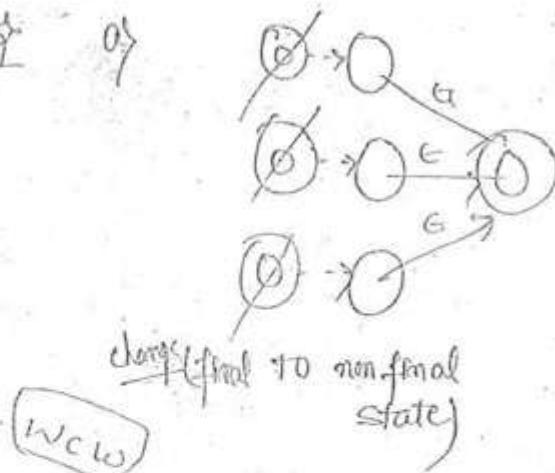
Solution



$\Rightarrow 0^* 1^* \Rightarrow 0^* + 1^* + 0^* 1^*$

$\Rightarrow 0^* 1^* 2^* \Rightarrow 0^* + 1^* + 2^* + 0^* 1^* + 0^* 2^* + 1^* 2^* + 0^* 1^* 2^*$

Ques 8) Q



But in dfa it is
not possible,
because, In dfa, null
move is not possible,

Ques 8b)

$$L = (\overline{aa} + ba)^* (e + b)$$

$$h(L) = (h(a) + h(b) \cdot N!)^* (e + h(b))$$

e.g:-

$$L = \{01\}$$

$$h(0) = 00$$

$$\therefore h(1) = 11$$

$$h^{-1}(L) = \emptyset$$

$$h(h^{-1}(L)) = h(\emptyset) = \emptyset.$$

$$\begin{array}{l} S \rightarrow aA | aB \\ A \rightarrow Ab | \alpha \end{array}$$

$$\begin{aligned} \Rightarrow S &\rightarrow aA \\ &\rightarrow \underline{aAb} \\ &\rightarrow aaAb \\ &\rightarrow aaab \\ &\rightarrow \boxed{a^n b^n} \end{aligned}$$

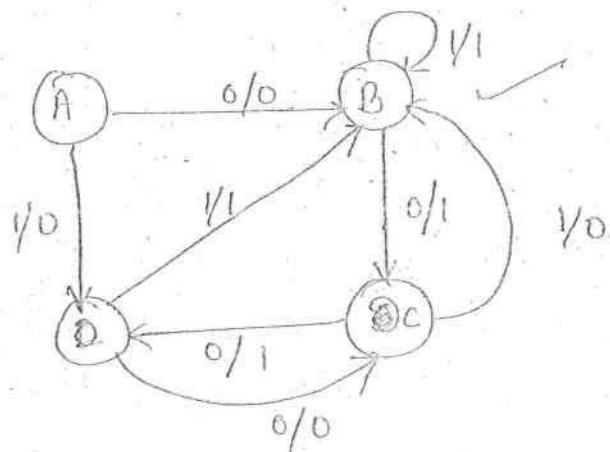
$$\begin{aligned} S &\rightarrow aCa \\ C &\rightarrow \underline{aCa} | b \end{aligned}$$

$$\begin{aligned} S &\rightarrow aCa \\ &\rightarrow aCaCa \\ &\rightarrow aaaa \\ &\rightarrow aaabaaa \\ &\rightarrow \boxed{a^n b^n} \end{aligned}$$

$$U, L^c \Rightarrow \oplus$$

$$\begin{aligned} A \oplus B &\Rightarrow (A - B) \cup (B - A) \\ &\Rightarrow (A \cap B^c) \cup (B \cap A^c) \end{aligned}$$

Sol 132



Sol

a) 01

b) 10

⇒ 101

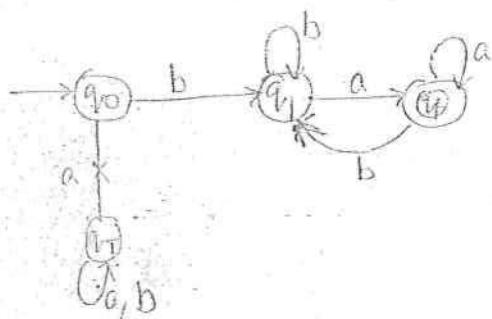
⇒ 110

⇒ last bit condition also asked?

①

~~Only~~ Not neither starting with 'a' nor ending with 'b' ^{Not} opposite

~~Sol~~ either starting with 'b' or ending with 'a'.



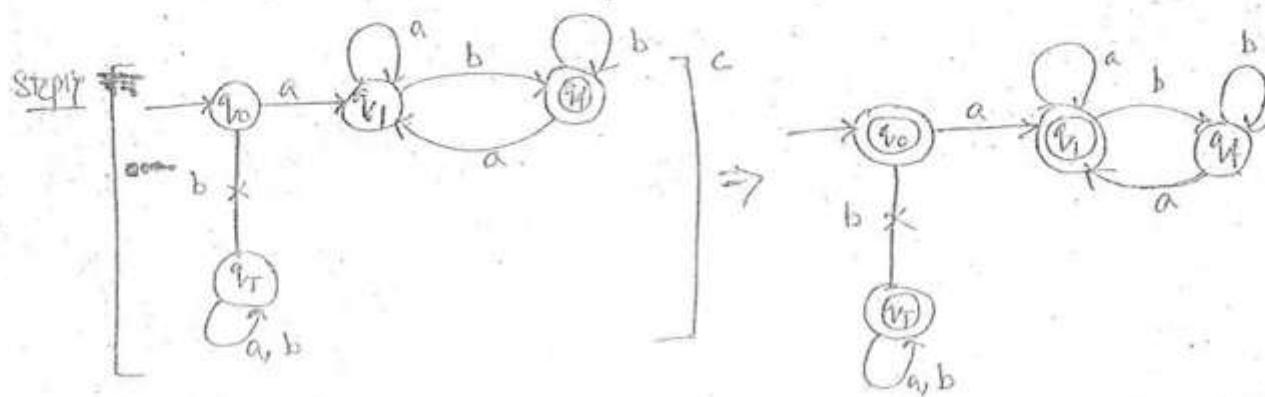
20/09/10

Context free grammar

Ques Not starting with 'a' or not ending with 'b'.

↳ [Not starting with 'a' or not ending with 'b']

→ [Starting with 'a' and ending with 'b']



n.e. $\Rightarrow \boxed{\lambda + a(a+b)^* + (a+b)^*b}$

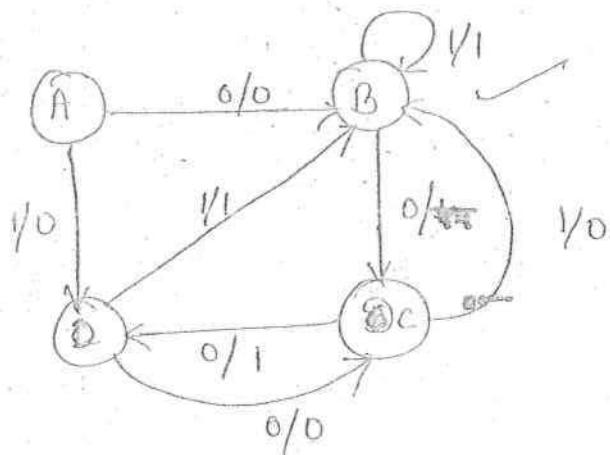
or

n.e. $\Rightarrow \boxed{\lambda + a(a+bb^*a)^* + b(a+b)^*}$

$$U, L^c \Rightarrow \oplus$$

$$\begin{aligned} A \oplus B &\Rightarrow (A - B) \cup (B - A) \\ &\Rightarrow (A \cap B^c) \cup (B \cap A^c) \end{aligned}$$

Sol} 134



Sol}

a) 01

b) 10

$\Rightarrow 101$

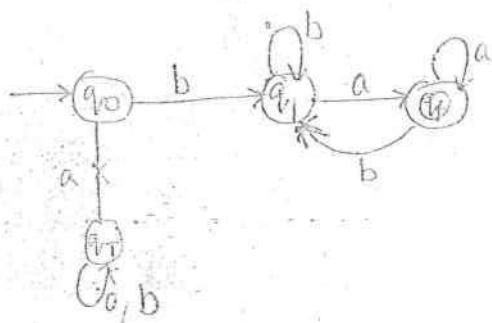
c) 110

\Rightarrow But bit condition also asked?

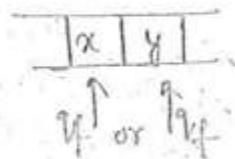
①

~~Ques~~ Not Neither Starting with 'a' nor ending with 'b' {
Opposite}

~~Ques~~ either starting with 'b' or ending with 'a'.



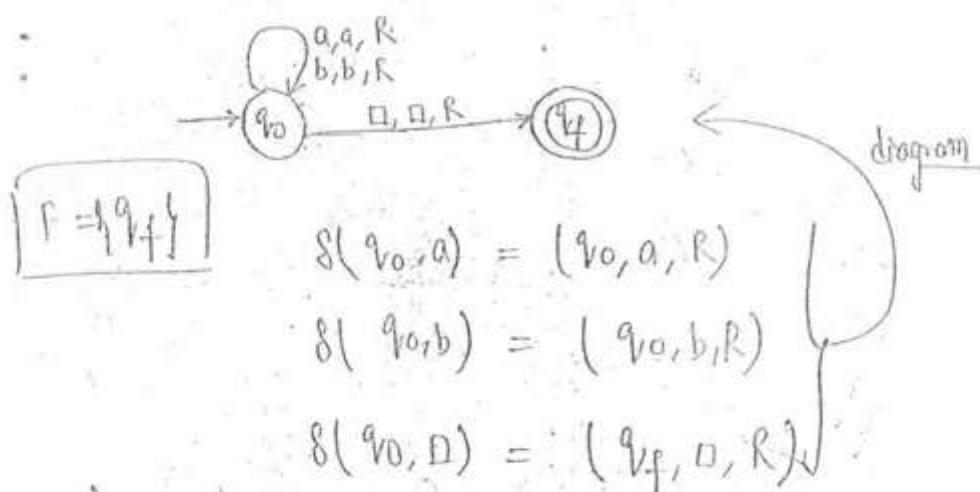
↳ $q_f \in Q$ → Tells about the final state whether on left or right.



↳ $x, y \in \Gamma^*$ → It shows Input Tape.

Ques Converting a machine into language.

$$M \rightarrow L(M)$$



Ques What language accepting by TM.

Soln

$$(a+b)^*$$

$|a|b|a|b|0|\square|$

Ques What is set of string (w) it will halt.

Soln

$$(a+b)^*$$

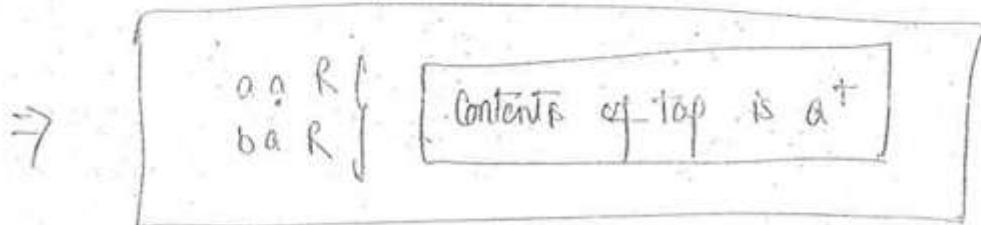
Ques $w \rightarrow \text{hang}$.

Solt \emptyset (empty)

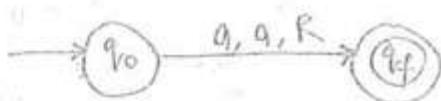
Ques Contents of Top of Set,

Solt Unchanged (By default always)

If it goes 'L', it will hang

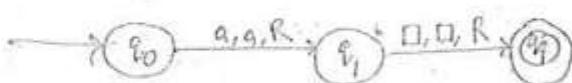


eg ii

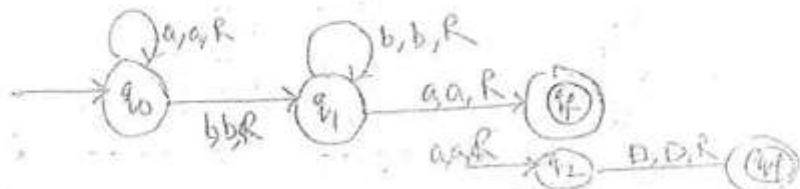


$\boxed{\square a | a | b | \square} \xrightarrow{?} 0 | ? \Rightarrow a(a+b)^*$

iii ii $\boxed{\square a | \square} \xrightarrow{?} 0 | ? \Rightarrow (\square) \text{ or (empty) or only } a$



iv

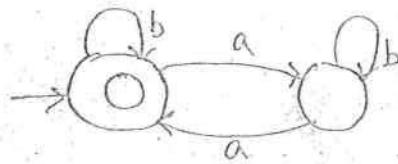


$\Rightarrow a^* b b^* (a+b)^*$

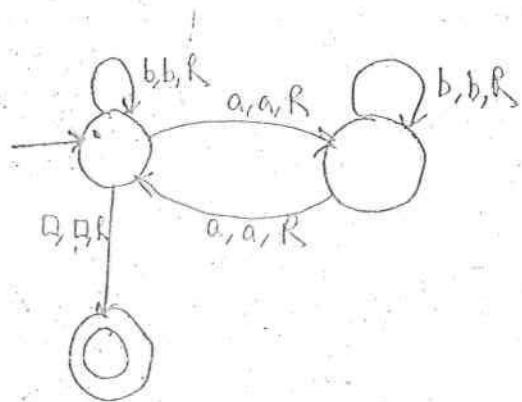
$\Rightarrow a^* b b^* a$

Even as

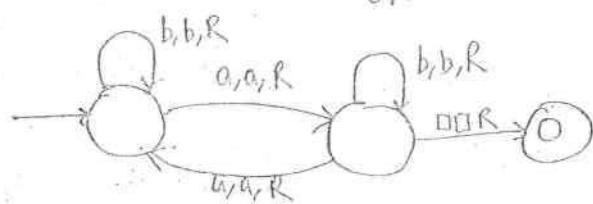
DFA



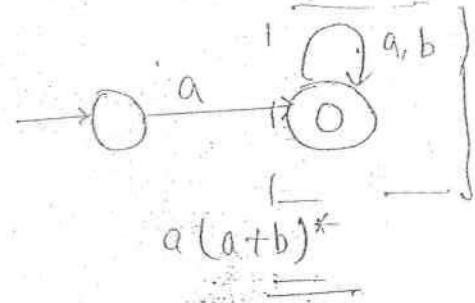
TM



OR

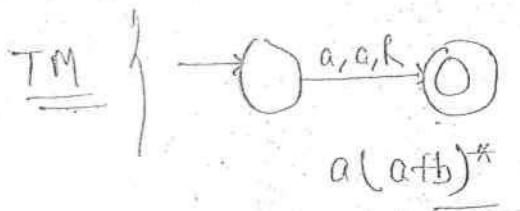


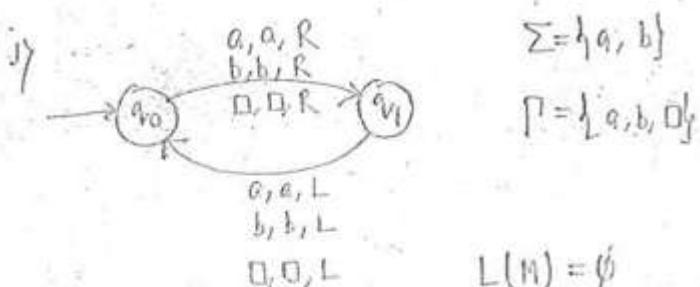
DFA



Turning M/L
does not
allow it,

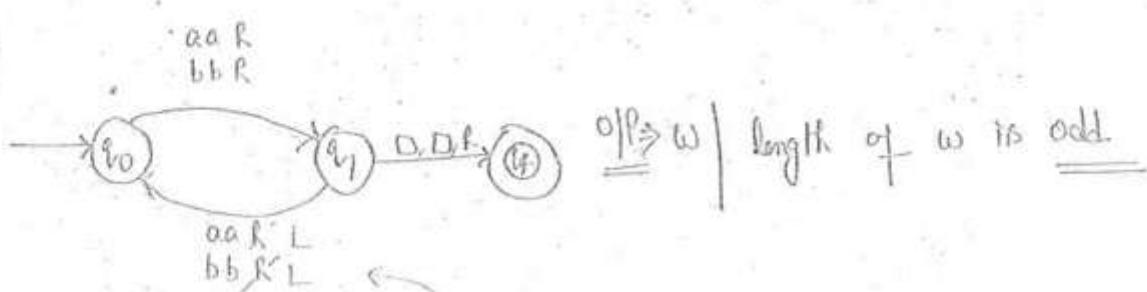
Rewriting
this,





$$\text{halt} = \emptyset$$

$$\text{hang} = (a+b)^+$$



\Rightarrow If Condition changed

Then output

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

$\square a \square \checkmark$

$$\text{halt} = \{a, b\}$$

$\square b \square \checkmark$

$$\text{hang} = \{(a+b)^+\} - \{a, b\} \text{ or } 2$$

$\square a \square \text{ Hang}$

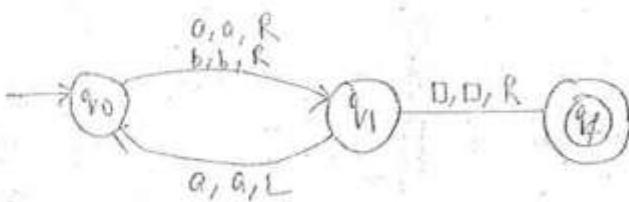
$w \mid \text{length of } w \geq 2$

$\square ab \square \text{ Hang}$

$\square ba \square \text{ Hang}$

$\square bb \square \text{ Hang}$

Content of Tape \Rightarrow Unchanged.



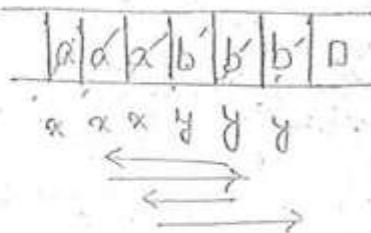
op $L(M) = \{a, b\}$

halt $\Rightarrow (a+b) \in (a+b)^*$

hang $\Rightarrow (a+b)a(a+b)^* \{ \text{Anything starting with } a \}$

3. Turning Machine as acceptor

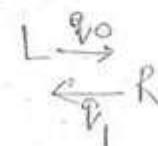
$\{ a^n b^n ; n \geq 1 \}$



$$\delta(q_0, a) = (q_1, x, R)$$

(When L and Right move
stat must change)

$$\delta(q_1, a) = (q_1, a, R)$$



$$\delta(q_1, y) = (q_1, y, R)$$

$$\delta(q_1, b) = (q_2, y, L)$$

$$\delta(q_2, y) = (q_2, y, L)$$

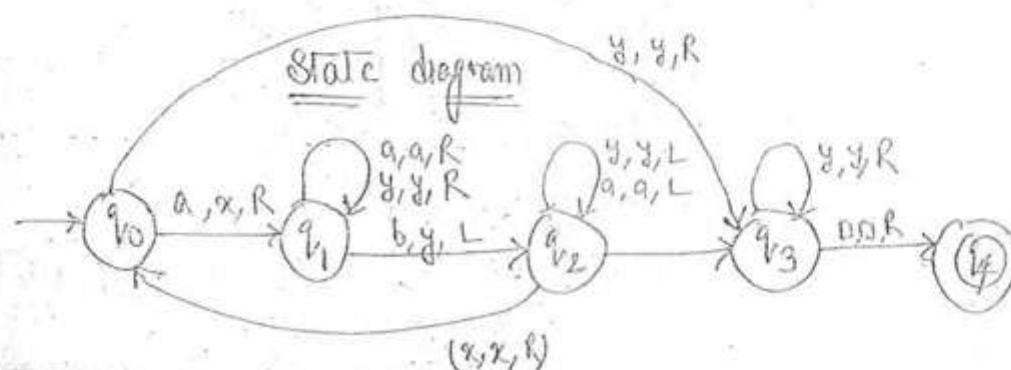
$$\delta(q_2, a) = (q_2, a, L)$$

$$\delta(q_2, x) = (q_0, x, R)$$

(It is only possible $\leftarrow \delta(q_0, y) = (q_3, y, R)$
when, a is finished)

$$\delta(q_3, y) = (q_3, y, R)$$

$$\delta(q_3, \square) = (q_4, \square, R)$$



$$q_0 = q_0 1 + q_0 0 1 + \lambda$$

$$q_0 \Rightarrow q_0(1+01) + \lambda$$

$$\Rightarrow R = R P + Q$$

$$R = Q P *$$

$$\therefore q_0 = \lambda (1+01)^*$$

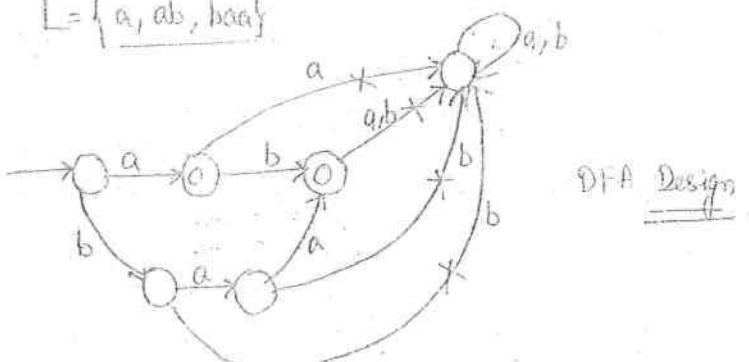
$$\hookrightarrow q_1 = q_1 0 0^* 1 (0+1)^*$$

$$\Rightarrow q_0 0 0 0^* 1 (0+1)^*$$

$$q = \boxed{(1+01)^* 0 0 0^* 1 (0+1)^*}$$

Ques $L = \{a, ab, baa\}$

Sol



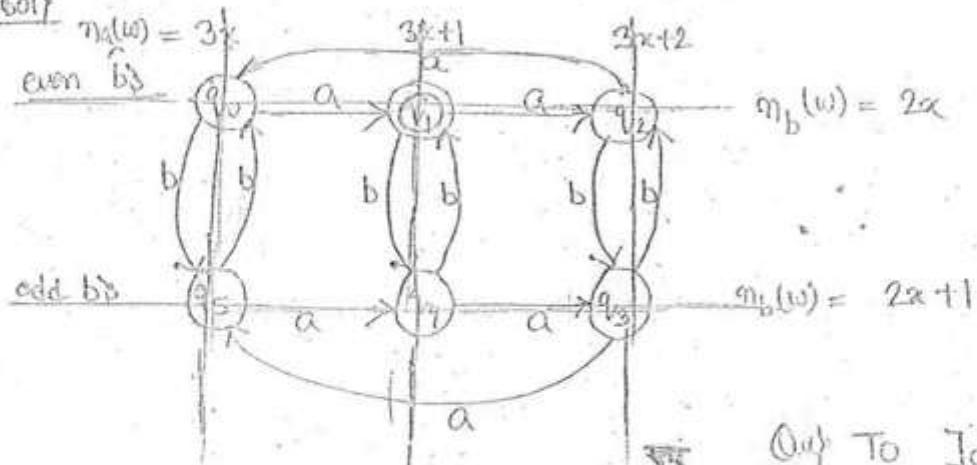
DFA Design

Ques

Engineering

~~(Q1)~~
$$(n_a(w) \bmod 3 = 1 \quad \& \quad n_b(w) \bmod 2 = 0)$$

Sol:



Here

T.D.C AUTOMATA

136