

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^t-Tree)
6. C-Concepts (Pointers)

D.N.T.
-10-

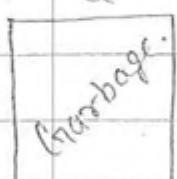
DETAILED	Page No.
	Date:

Arrays

#

Declaration

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

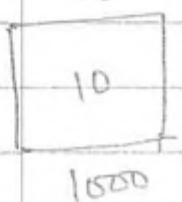


(Declaration is not for user it is for compiler)

#

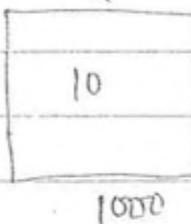
Initialization

`a = 10;`



#

`int a = 10;`



#

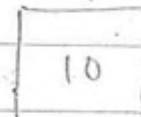
`printf("%d", a)`



variable-name indicates value.

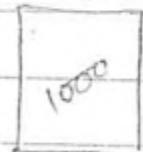
O/P : 10.

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



// Normal variable
initialization

int & h b



It is the compiler's
purpose.

not for us.

$$b = \& a$$

11 - Pointer Variable initialization.

Printf (" %d ", a) ⇒ 10

Print { ("%", & b.) } ⇒ 10

* (1002)

1e

& ⇒ Reference operator

* ⇒ de-referencie operator.

#

int a = 10

a immediate
addressing
mode

10

1000

⇒ Value.

a = 10

#

int * b

b = & a

1000

2000

b Direct addressing
mode

⇒ Address.

* b = 10

#

int ** c

c = & b

2000

3000

c Indirect
addressing
mode

⇒ (3: memory access)

x(*c) = 10

X

Array

#

int a = 10 ;

a

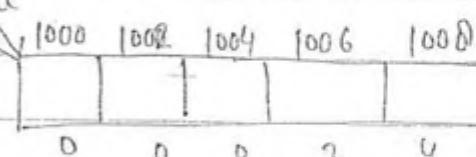
10

1000

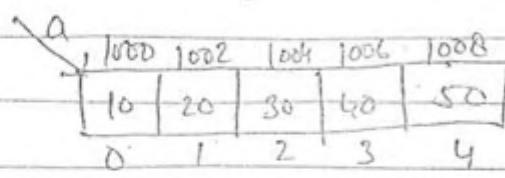
#

int a[5] ;

a



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



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

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

$a[3] = 40$

$$\ast(a+3)$$

$$\downarrow$$

$$\ast(a+6)$$

$$\ast(1000+6)$$

$$\ast(1006)$$

\Downarrow

40

$\ast(a+0) = 10$

$$a[0] = 10$$

for compiler

$$\# \quad * (a+4) = * (a+8)$$

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

 $a[4]$

$$= * (1008)$$

for user

$$= 50$$

#

$$a[i] = * (a+i) = *(i+8) = i[a]$$

in address

#

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

$$= *(1002)$$

$$= 20$$

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

$$= *(4+8)$$

$$= 4[a].$$

One Dimensional Array.

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

a	1000	1002	1004	1006	1008
	10	20	30	40	50
	0	1	2	3	4

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

~~* Consider the following array declaration.~~

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

~~Base address = 1000~~

~~c = 10, RMO, then~~

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

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

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

~~Soln~~

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

~~* Consider the array A [-5 - - - 45, -5 - - - 45]~~

~~BH = 0, S3 C = 10, RMO~~

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

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

$$\text{columns} = 5 + 5 + 1 = 11$$

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

) * 2

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

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

Column - Major - Order

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

0)] * 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

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 ₁₂	a ₁₂	a ₁₃	a ₁₄	a ₁₃	a ₁₃	a ₁₃	a ₁₄	a ₂₄	a ₃₄	a ₄₄	a ₄₄
0	1	2	3	4	5	6	7	0	9	10	11	12	13	14	15

$$\begin{aligned}
 \text{Loc}(a_{3,4}) &= 1000 + [(4-1) \times 4 + (3-1)] \times 2 \\
 &= 1000 + 28 = \underline{\underline{1028}}
 \end{aligned}$$

~~Q9~~

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

~~Q10~~

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

* Consider the array A which is declared as follows:

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

$$B = 0, C = 5, M = 0$$

Find location of $a[B(a_{2,25})]$.

~~Soln~~

$$\text{no of rows} = 5 + 3 + 1 = 9$$

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

$$\begin{aligned} \text{Loc}(a_{2,25}) &= 0 + [(25+8) \times 9 + (2+5)] \times 5 \\ &= 0 + [339] \times 5 \\ &= \underline{\underline{1695}} \end{aligned}$$

2

$$\text{Loc}(a_{-4,-1}) = 0 + [(-1+5)*11 + (-4+5)] * 5.$$

$$= 0 + [44 + 1] * 5$$

$$= 225$$

Formula

Column Major Order

$$\text{off} [lb_1 \dots ub_1, lb_2 \dots ub_2]$$

BA, c, mr, nc, CMO then

find Loc(i, r) ?

Where

BA = Base address

c = Size of element

mr = no of rows.

nc = " " columns.

CMO = Column Major Order.

$$mr = ub_1 - lb_1 + 1$$

$$nc = ub_2 - lb_2 + 1$$

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

Row Major Order

A [lb₁, ..., lb₁, lb₂, ..., lb₂]

BA, C, RMO

$$mr = lb_1 - lb_1 + 1$$

$$mc = lb_2 - lb_2 + 1$$

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

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	a ₂₃ ⁰	a ₂₄ ⁰
a ₃₁	a ₃₂	a ₃₃ ⁰	a ₃₄ ⁰
a ₄₁	a ₄₂	a ₄₃ ⁰	a ₄₄ ⁰

MTU-320

A is said to be LTM

$i \geq j$

$A[i,j] = 0$ if $i < j$

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

$[b_2]$] $\times C$

...

LTM

bed]

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									
a_{11}	a_{21}	a_{31}	a_{41}	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

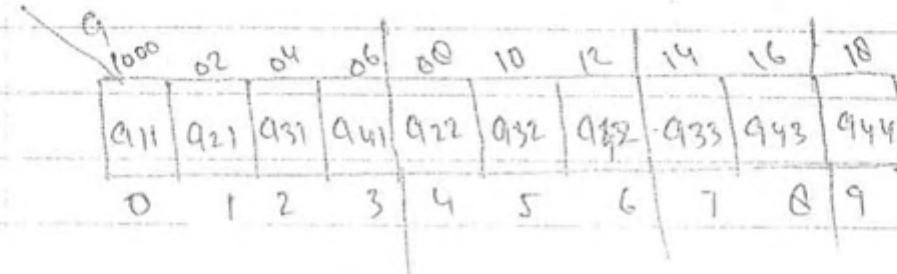
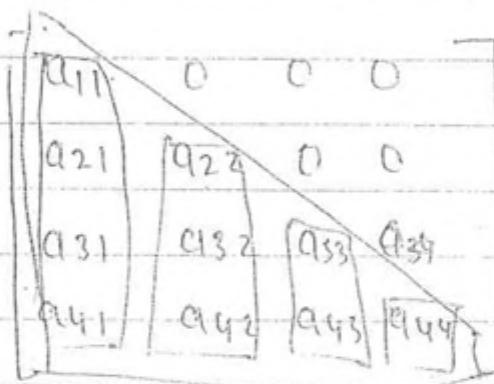
$$\text{LOC}(a_{4,3}) = 1000 + [(4-1) + (3-1)] * 2 \\ \text{(natural no)} \\ \text{sum}$$

$$= 1000 + [(4-1)(4-1+1) + 2] * 2$$

$$= 1000 + 16$$

$$= \underline{\underline{1016}}$$

Column - Major Order



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

$$\begin{aligned}
 \text{LOC}(a_{44}) &= 1000 + \left[\frac{4 \times 5}{2} - \frac{(4-4+1)(4-4+1+1)}{2} + (4-4) \right] \times 2 \\
 &= 1000 + 18 \\
 &= 1018
 \end{aligned}$$

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

#

$$A [-25 \dots +25, -25 \dots +25]$$

CMO ; LTM

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

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

Find ?

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

x2

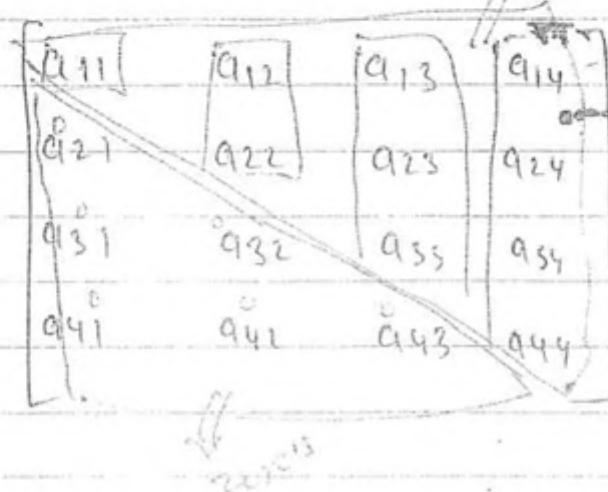
(41H)

DELM Page No. _____
Date: 29/11

UPPER TRIANGULAR MATRIX

A matrix A is said to be upper Triangular Matrix
if:

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



Column Major Order

1000	02	04	06	08	10	12	14	16	18
01	02	022	015	023	035	014	024	034	044
0	1	2	3	4	5	6	7	8	9

प्राप्त करने का तरीका

$$\text{LOC}(a_{3,4}) = 1000 + \left[\frac{(4-1)(4-1+1)}{2} + (3-1) \right] * 2$$

$$= 1000 + 16$$

$$= 1016$$

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

formula

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

BA, c, rr, cr, UTM, CMO, RMO

Find location of $a(i,j) = ?$

$$\boxed{\text{Loc}(a_{i,j}) = BA + \left[\frac{(j-ub_1)(j-lb_1+1)}{2} + (i-lb_1) \right] * c}$$

Column Major order

$$rr = ub_1 - lb_1 + 1$$

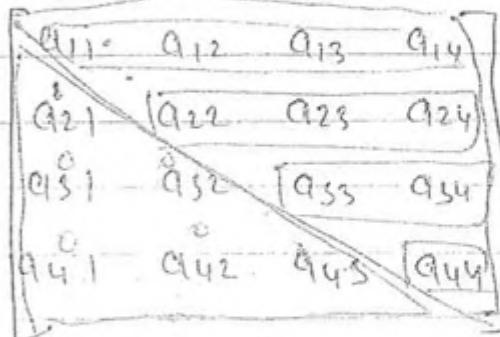
$$nc = ub_2 - lb_2 + 1$$

$$\begin{aligned}
 \text{Loc}(a_{i,j}) &= BA + \left[\frac{rr(nc+1)}{2} - \underbrace{(ub_1-i+1)(ub_1-i+1+1)}_{2} \right. \\
 &\quad \left. + (j-1) \right] * c
 \end{aligned}$$

Row-Major Order

UPPER TRIANGULAR MATRIX

Row Major Order



Row						Column				
0	1	2	3	4	5	6	7	8	9	
a ₁₁	a ₁₂	a ₁₃	a ₁₄	a ₂₁	a ₂₂	a ₂₃	a ₂₄	a ₃₁	a ₃₂	a ₃₃

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

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

$$= 1000 + 16$$

$$= 1016$$

$$LOC(a_{14}) = 1000 \left[\left(\frac{4 \times 5}{2} - \frac{(4-1+1)(4-1+1+1)}{2} + (4-1) \right) \times 2 \right]$$

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

$$= 1006$$

====

IX

Q) A one dimensional Array A contains $A[1 \dots 75]$. Each element is a string and take 3 bits. The Array store a location 1120. Then what is the location of $A(4,9)$.

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

Q) Consider the following declaration of two dimensional array in C. char $a[100][100]$. Assume that array is storing many addresses. What is location of $A[40][50]$.

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

$A[1 \dots 75]$

$$c = 3$$

$$\beta A = 1120$$

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

$$= 1264$$

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

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

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 $A[90, 90]$ is stored in $B[6]$ where is $A[90, 90]$ is stored in B ?

Ans 4175

b) 0

c) 4165

d) 4166

Sol

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

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

$B[0 \dots \frac{1}{2}(4)(4+1)-1]$

$B[0 \dots 9]$

$C = 1, BA = 0 - QM0, CFM1$

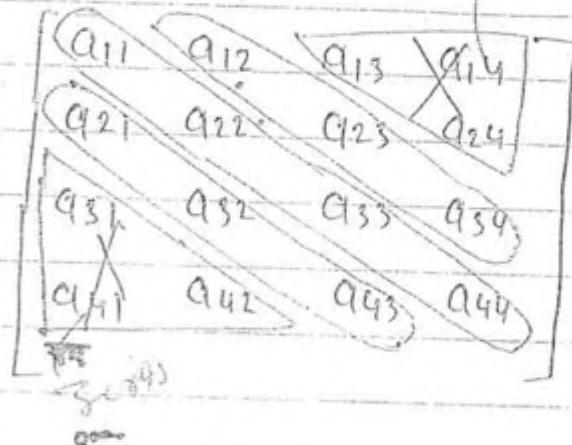
$$LOC(A_{90,90}) = C + \left[\frac{(q_0-0)(q_0-0H)}{2} + (00-0) \right] K_1$$

$$= 4175$$

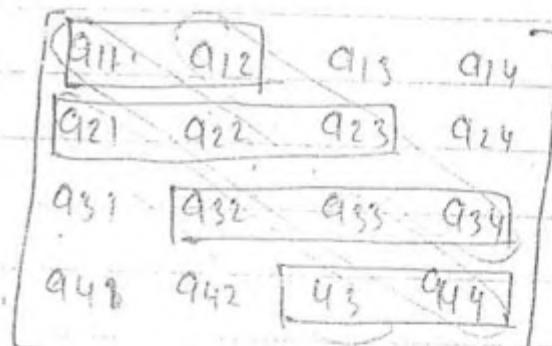
$$\begin{bmatrix} 3,0 \\ 1,1 \end{bmatrix} - (n-1)$$

$$\begin{bmatrix} 1, n(n+1)-1 \\ 2 \end{bmatrix}$$
in $B[6]$

$$\begin{array}{l|l} 4 \times 4 = 16 & 3 \times 3 = 9 \\ 4+3+3=10 & 3 \times 2 \end{array}$$



Row-Major-Order



	00	02	04	06	08	10	12	14	16	18
a ₁₁	a ₁₁	a ₁₂	a ₂₁	a ₂₂	a ₃₁	a ₃₂	a ₄₁	a ₄₂	a ₄₃	a ₄₄
	0	1	2	3	4	5	6	7	8	9

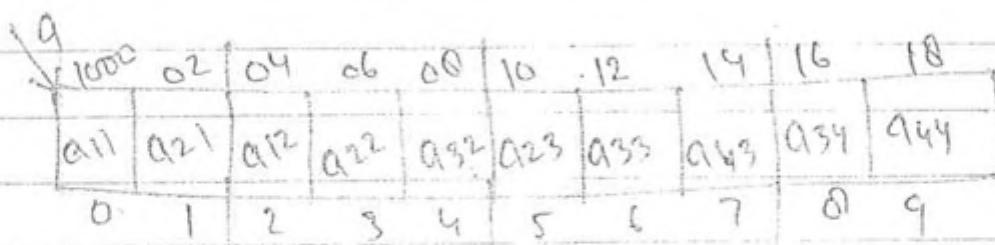
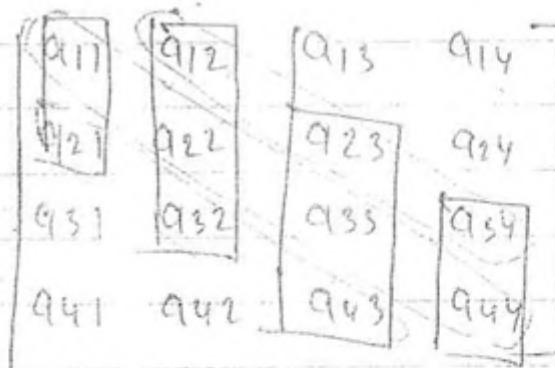
$$\begin{aligned}
 \text{Loc.}(a_{43}) &= 1000 + [3(4-1)-1 + 3-(4-1)] \# 2 \\
 &= 1000 + [16] \\
 &= 1016
 \end{aligned}$$

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

formula

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

(column Major Order)



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

12

Formula

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

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

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

if ($i == jH$)

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

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

↳

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}

Have 1st 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 ₁₃	a ₁₄	a ₂₃	a ₂₄
0	1	2	3	4	5	6	7	8	9		

Middle

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

↓
if ($i = lb_1$)

$$\boxed{\text{Loc}(a_{i,j}) = BA + (j - lb_1) * c} \Rightarrow \text{1st Row}$$

Lower

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

↓

if ($i = ub_1$)

$$\boxed{\text{Loc}(a_{i,j}) = BA + [N + (j - ub_1)] * c} \Rightarrow \text{Last Row}$$

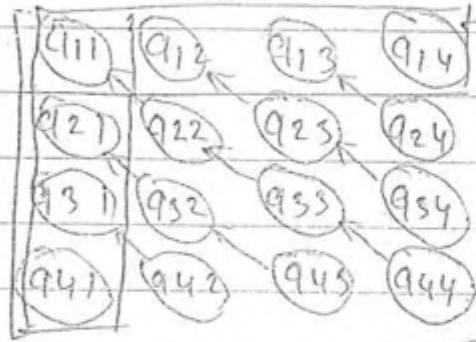
Loc($a_{i,j}$)

↓

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

$$\boxed{\text{Loc}(a_{i,j}) = BA + [N + N + (i - 2)] * c} \Rightarrow \text{Middle}$$

Toeplitz - Matrix



Matrix A is said to be Toeplitz-matrix if.

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



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

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

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

⇒ Store 1st row & remaining elements of 1st column.

a ₁₁ a ₁₂ a ₁₃ a ₁₄				1 st row (Remaining)			
				1 st column (Remaining)			
a ₁₁	a ₁₂	a ₁₃	a ₁₄	a ₂₁	a ₃₁	a ₄₁	
a ₂₁	1	2	3	4	5	6	
a ₂₂	a ₂₃			a ₃₂			
a ₃₁	a ₃₂	a ₃₃	a ₃₄	a ₄₃	a ₄₄		
a ₄₁							

$LOC(a_{i,j})$

↓

if ($i \leq j$)

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

 $LOC(a_{i,j})$

↓

if ($i > j$)

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

#	10	20	30	40	50
	0	1	2	3	4

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

$$= 1000 + 6$$

$$= * (1000)$$

$$= 46$$

1st column.

offset
needed to calculate

10	20	30	40	50
3	4	5	6	7

Offset
(should be calculated)

$$\begin{aligned}
 & \star(\text{loc}(a(6))) = BA + (6-5) \times 2 \\
 & = 1000 + 6 \\
 & = \star(1006) \\
 & = 40
 \end{aligned}$$

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

✓ $\star(a+1) = a[1]$

$$\begin{aligned}
 & \downarrow \quad \downarrow \\
 & \star(1000+2) \quad \star(1+1) \\
 & \downarrow \quad \downarrow
 \end{aligned}$$

(we calculated)

$$\begin{array}{ccc} * (1002) & & * (1+a) \\ \downarrow & & \downarrow \\ 20 & & 1[4] \end{array}$$

$$a++ X, [a = a+1]$$

\downarrow \downarrow
 L-value R-value

\downarrow

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}; ✓

\downarrow optional

✓ int a[3] = {10, 20, 30, 40, 50}

a				
10	20	30	40	50
0	1	2	3	4

a[0]

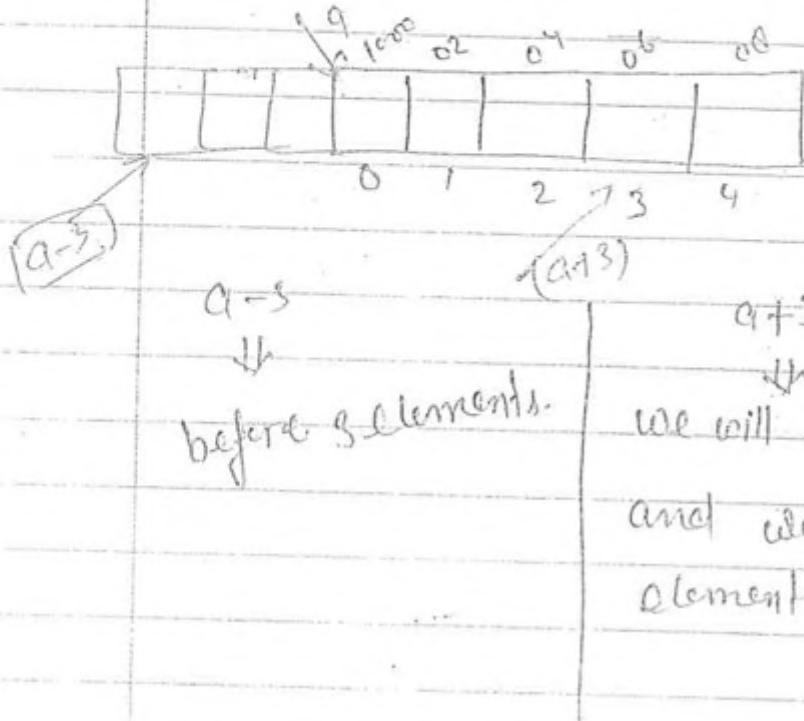
a[1]

a[2]

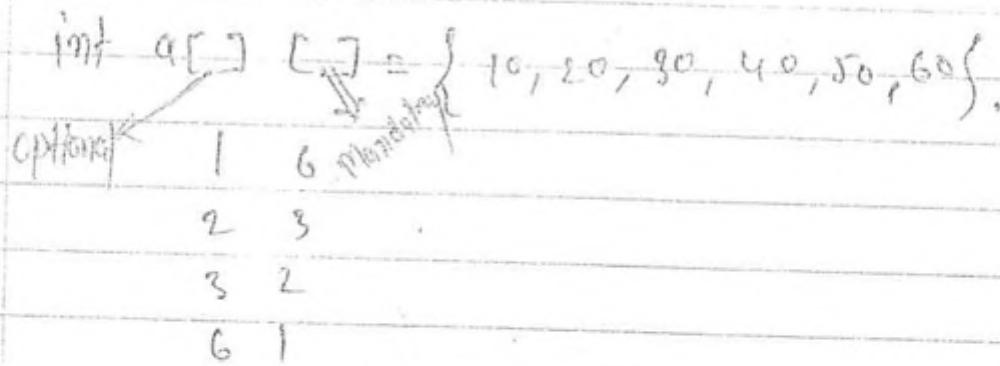
a[3]

garbage, bcoz total statement you stored.

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



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



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

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

optional ↓ Mandatory
 ↓ Mandatory

elements.

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

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

↓

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

$$\text{rr} = \text{Ub}_2 - \text{lb}_2 + 1$$

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

$$\text{nc} = \text{Ub}_3 - \text{lb}_3 + 1$$

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

$$\text{nr} = \text{Ub}_4 - \text{lb}_4 + 1$$

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

Size of array = $\text{nr} \times \text{nc}$

$a[3] \Rightarrow 5 \times 1$

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

↓

$a[\text{lb}_1 - \text{Ub}_1, \text{lb}_2 - \text{Ub}_2, \dots, \text{lb}_5 - \text{Ub}_5]$

Row major order

$$\text{LOC}(a_{i,j,k}) = \text{BA} + [(i - \text{lb}_1)(\text{nr} \times \text{nc}) + (j - \text{lb}_2) \times \text{nc} + (k - \text{lb}_3)] \times c$$

Column Major Order

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

Q What does the following C program.

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

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

```
{ p=q  
*p=q; }
```

```
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();
}
```

$$u = 4$$

Post increment

$$y = u++ + u++$$

Printf(u, y);

↓ ↓

6 6

$$u = 4 \text{ } 6$$

Pre increment

$$u = 5 \text{ } 6$$

$$y = ++u + ++u$$

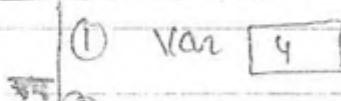
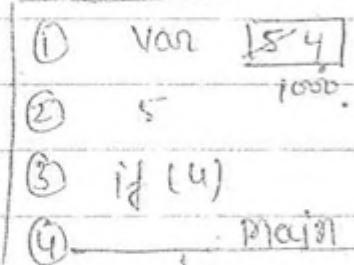
printf(u, y);

↑ ↑

6 11

\Rightarrow If Static is not (without Static)

plain



• ③ if (4)

16 Main

Digitized by srujanika@gmail.com

Stock is overfull

My Onetime

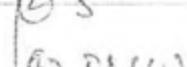
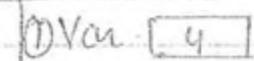
only On time

Memory Cycles

and also initialisation

→ Memory create only

time of compilation.



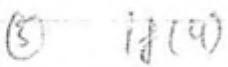
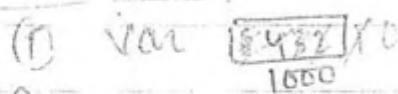
10

O/P \Rightarrow 5555 - - -

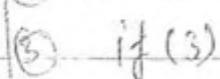
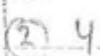
- 3 -

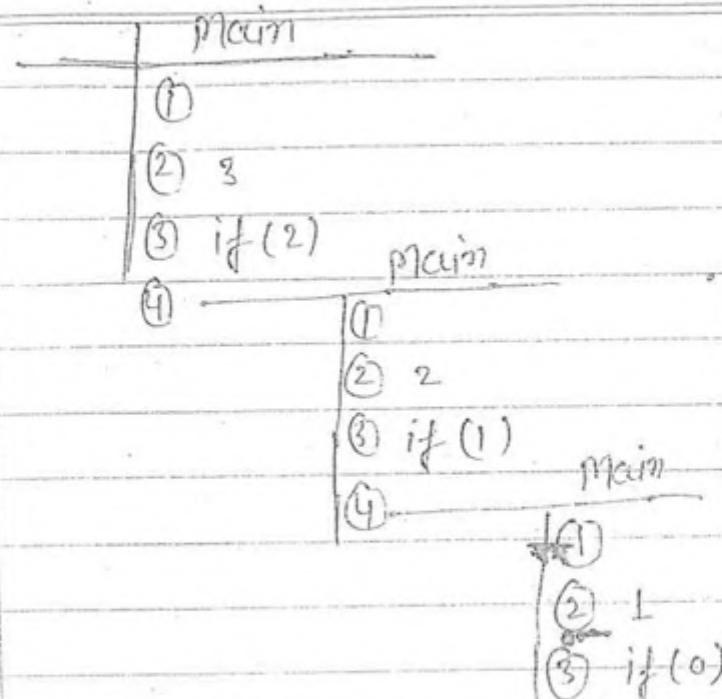
⇒ If static is available (with static)

Page

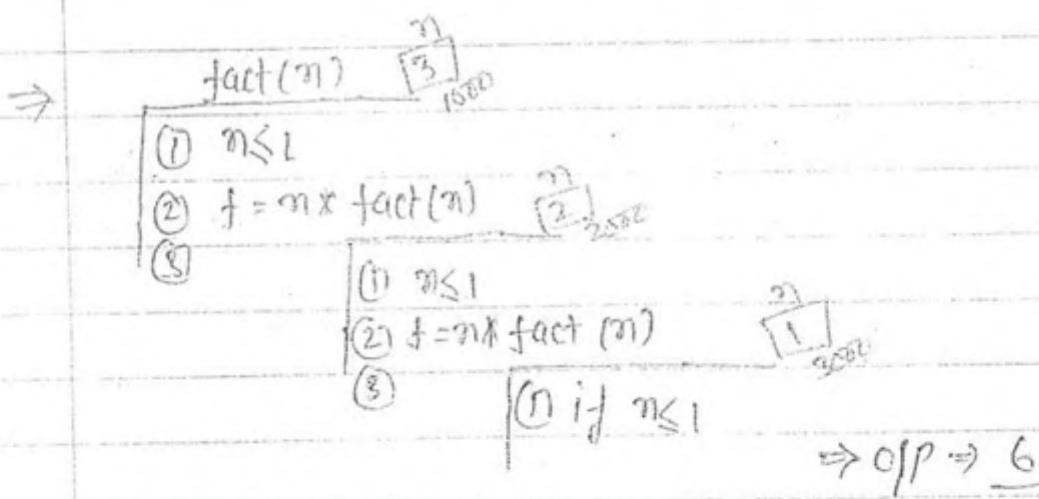


plain





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.

$b \quad n$
 $f(m) \boxed{1} + r$

- (1)
(2)
(3)
(4)

$b \quad n$
 $f(m) \boxed{0} + r$

- (1)
(2)

Ques Consider the following C program.

include <stdio.h>

int fung (int u);
int fung (int u);

main()

{ $\boxed{5}$ $\frac{1}{1000}$ }

$\boxed{143}$ $\frac{1}{2000}$

count
 $\boxed{12}$ $\frac{1}{3000}$

int x=5, y=10, count;

for (count=1 ; count <= 2 ; ++count)

{ $y = \boxed{10} \times \boxed{15} \leftarrow \boxed{150}$
 $y = \boxed{150} + \boxed{143} \leftarrow \boxed{293}$
 $y = \boxed{293} + \boxed{5} \leftarrow \boxed{298}$

printf ("%d", y);

}

fung (int x)

{ int y; \cancel{x}
 $y = \cancel{fung(x)}$; \cancel{y}
return (y);

$\boxed{16}$ $\frac{1}{1000}$

$\boxed{16}$ $\frac{1}{1000}$

$\boxed{10}$ $\frac{1}{1000}$

func (int u)

```

    {
        static int y = 10;
        y = y + 1;
        return (u + y);
    }

```

O/P =

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

~~sd~~

Under the following program.

int incr(int i) ^{0X234} count ^{0X81610}

```

    {
        static int count = 0;
        count = count + i;
        return (count);
    }

```

Main()

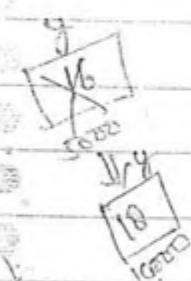
```

    {
        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=4, k=-1, d=2, m;

$$m = \left(\frac{d^{i+2} + j + 2 \cdot k + 1}{d^k + 1} \right) \mod 4,$$

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

1 -> 1 3 1

#

Void main()

{

int i=0

for(i=0, i<20, i++)

Switch(i)

Case 0

close c; i+=5;

Case 1

close l; i+=2;

Case 2

close g; i+=5;

default i+=4

break;

}

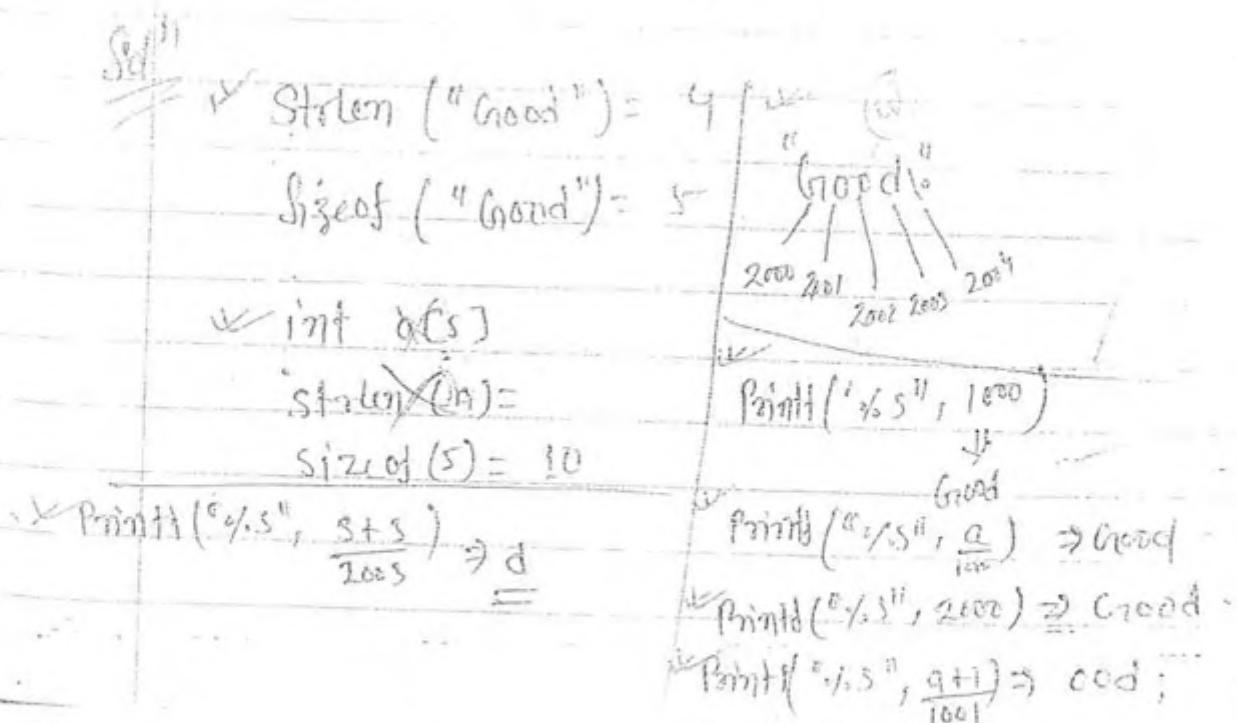
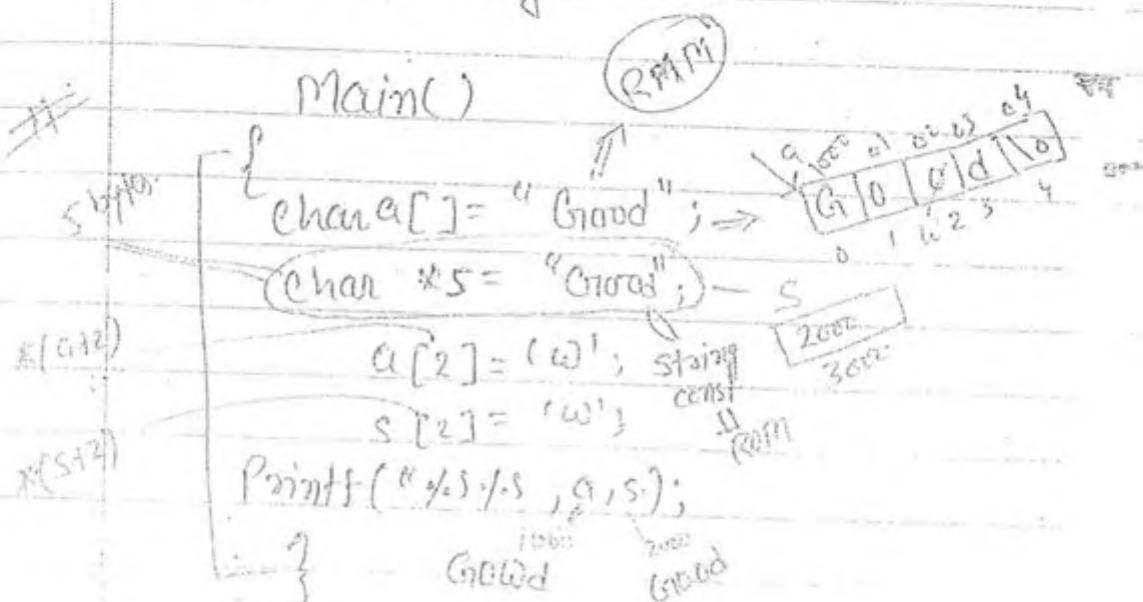
Printf("%d,%d,%d", i);

O/P = 16, 31

Xp

Note:- In switch statement after every case there 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.



✓ $\text{printf}(" \%c ", \&a) ;$

\downarrow

6

✓ $\text{printf}(" \%c ", \&(s+3)) ;$

\downarrow

d

✓ $\text{printf}("%c", \&(a+3)) ;$

\downarrow

✓ $\text{printf}("%c", a)$

✓ $\text{printf}("%d", *q)$

ASCII value

of a.

✓ $\text{printf}("%c", *(a+2)) ;$

\downarrow

a[2]

✓ $\text{printf}("%c", *(s+2))$

\downarrow

s[2]

Procedure small()

begin

Var u: real.

$$u = 0.125$$

Show();

end;

begin

$$u = 0.25$$

Show();

small();

end.

Area

main

$$u = 0.25$$

Static: 0.25, 0.25

Dynamic: 0.25,

Static

Var u, y: integer;

Procedure P(var n: integer);

begin

$$u = (n+4)/(n-3)$$

end

Procedure Q()

begin

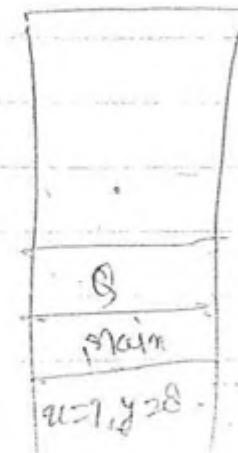
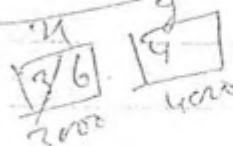
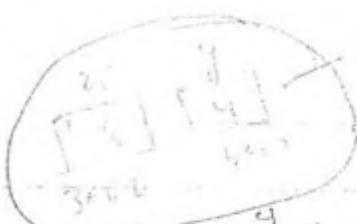
Var u, y: integers;

$$u = 5, y = 4$$

P(y)

Write(u);

end.



```

begin
  u:=1; y:=0;
  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:=1, b:=2

;

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;

Static: 2, 2, 1,

Dynamic: 2, 5, 2

Parameter Passing Techniques

$x \quad \quad \quad t$

- ① Call by value.
- ② Call by reference.
- ③ Call by copy-restore (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) [Example]

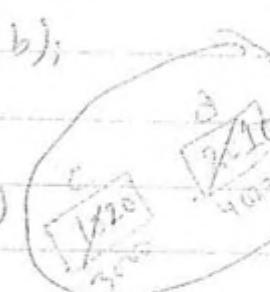
Main()

```
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;
}
```



Before - 10, 20 (without swap)
- 20, 10 (swap)

Value
Changes are taken place only here.

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

i j

i j

i j

i j

①
②

①

2 5

3

4

5

5

5

5

5

50, 40, 30, 20, 10

Print array (a, i, j)

```

    {
        if (i == j)      55
            {
                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 ($n=10$):

84^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 |

5

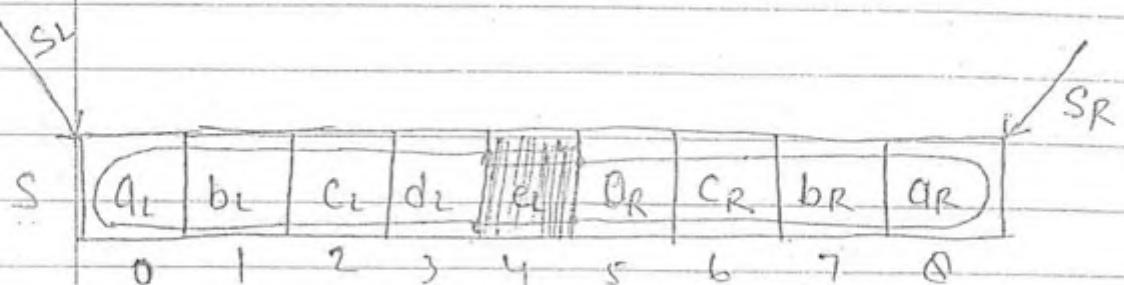
10, 20, 30, 40, 50

9

84

Efficient way of implementing multiple stack.

$$M=9$$

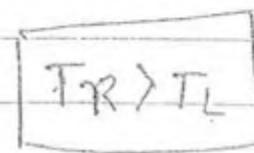


$$TL = -1, 0, 1, 2, 3, 4$$

$$TR = 5, 6, 7, 8$$

$$TL++$$

$$S[TL] = u$$



$$TR--$$

$$S[TR] = v$$

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

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

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

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

d) $TL = TR + 1$ ✓

DELTA	Page No. _____
	Date: _____

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[T_i];

T_i --;

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, if it is giving error
 message saying that stack is overflow.

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

{
if ($T_i = i + 1 \cdot \left(\frac{M}{N}\right) - 1$)

{
printf("stack is overflow");
exit(1)
}

else

{
 T_i++
 $S[T_i] = u$
}

This code true for all the stack other
than last stack. For the last stack.

Push code assume as single stack push

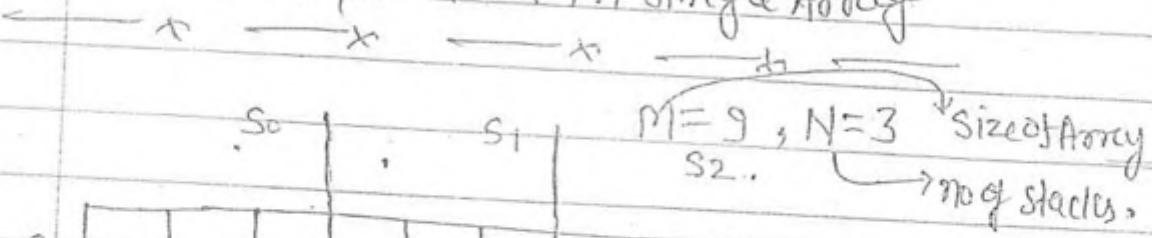
Code for Pop (S, M, N, To)

{
if ($T_i = i + 1 \cdot \left(\frac{M}{N}\right) - 1$)

else {
 $y = S[T_0]$
 T_0--
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.

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

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

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

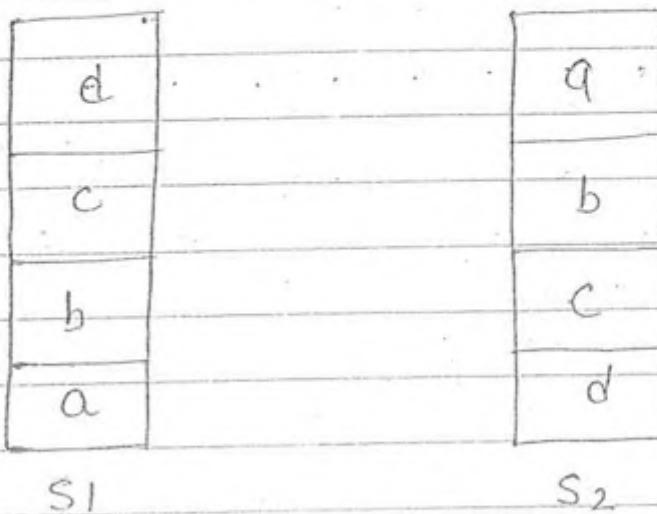
$$i^{\text{th}} = T_i = i \left(\frac{M}{N} \right) - 1 =$$

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

```

    {
        T0 = (0+1) (M/N) - 1
    }
    T0 ++
    S[T0] = u
}
  
```

Implementing Queue with Stack.



Queue:

Insertion

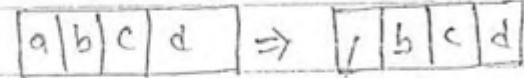
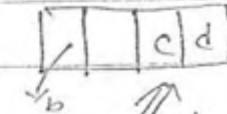
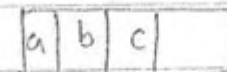
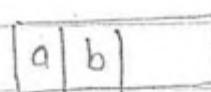
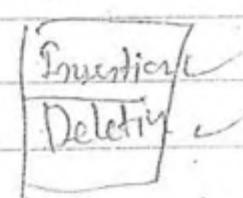
Push all the element
into stack S₁

Deletion

(1) Pop all the element
from S₁ and push
into S₂

(2) Pop elements from S₂

Queue



$\searrow a$

~~Main()~~

Char p[20]; ✓

Char *s = "string";

int length = strlen(s);

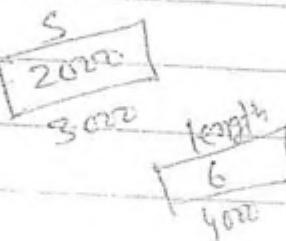
for (i=0; i<length, i++)

{
 p[i] = s[length-i]; s[6] s[6]
 s[5] s[5]
 s[4] s[4]
 s[3] s[3]
 s[2] s[2]
 s[1] s[1]
 s[0] s[0]

printf("%s", p); s[6] s[6]
 s[5] s[5]
 s[4] s[4]
 s[3] s[3]
 s[2] s[2]
 s[1] s[1]
 s[0] s[0]

}

no output

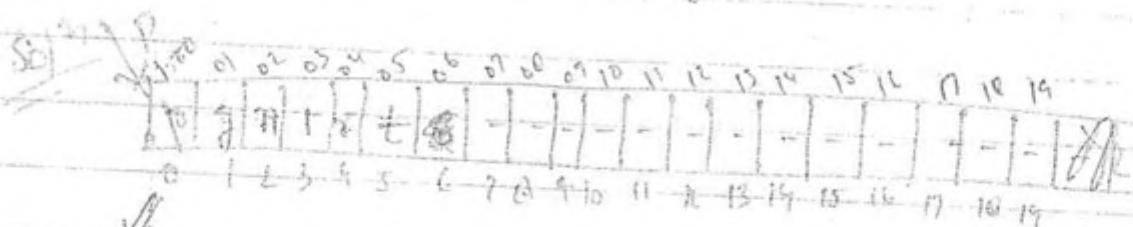


a) string.

b) gnirts. c) no output

d) string

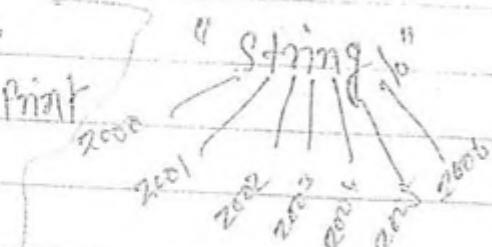
d) gnirt



null char:

Printf start don't print

anything
after null



~~Main~~ ^{b4 yes})

{
 Char a[2] = "Good"

Char *s = "Bad";

a[2] = 'o'; ⁵⁴⁷⁰ 

s[2] = 'o';

printf("%c,%c,%c", a, s); ⁷⁰⁷⁰

Note

- (1) 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]

X

②

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

a = 3002 X

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

int u=10;

Main();

{

int u=10;

f()

{

f()

f()

f()

Global

f()

{

int u=20

f(u)

{

f()

{

f(u)

{

f(u)

{

printf ("%d", u)

{

Control Stack

Dynamic Scope

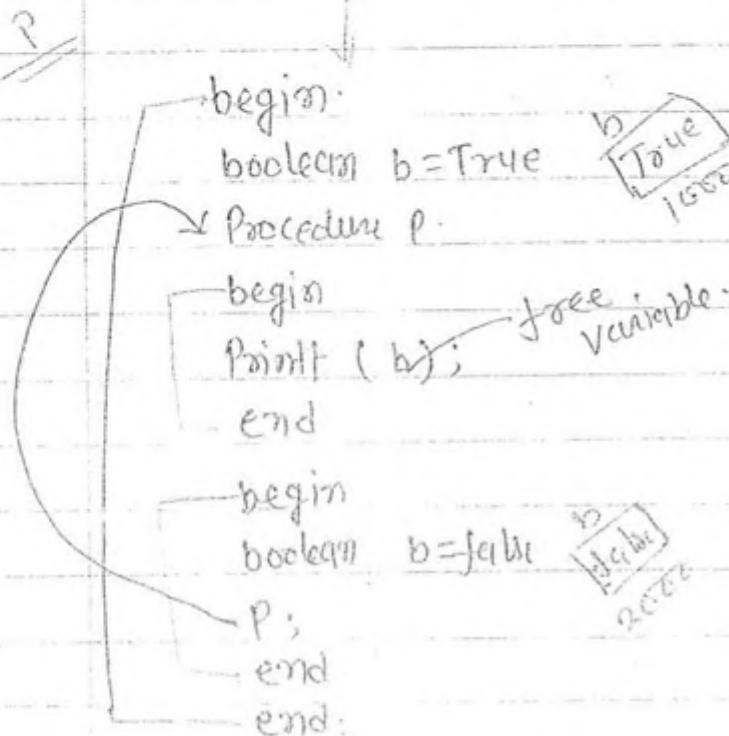
Stack Allocation

modern language can call the main like C.

DETA	Page No. _____
	Date: _____

Dynamic Scoping

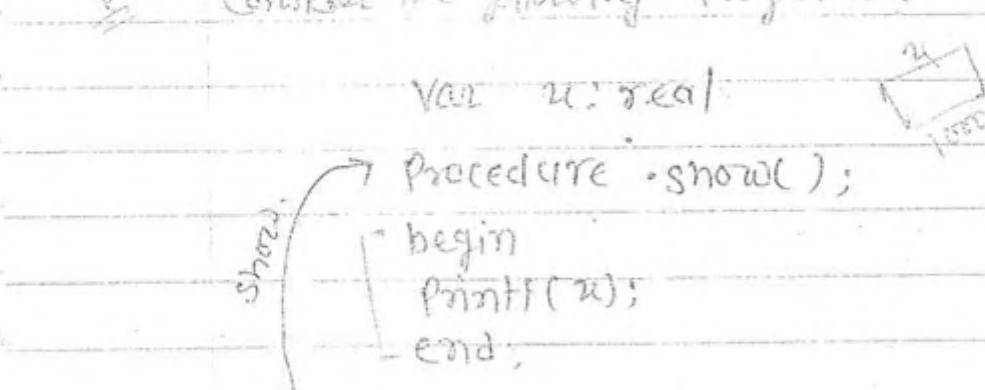
most recently called function. C, language follow the dynamic scoping.



Static : true

Dynamic: false

Consider the following Program.



DELTA	Page No. _____
	Date: _____

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

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

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

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

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

)

1

1

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

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

$$= ny + 2(u+y) \underbrace{(n-1)(n-1+1)}_2$$

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

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

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

 \cancel{x}

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

$$= \boxed{n(u+y) - u}$$

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

$$\frac{3}{+ 19}{}_{\overline{22}}$$

$\Rightarrow 11$

then $n=2$

$$u = 5$$

$$\underline{\underline{y = 3}}$$

~~$u = 2$~~

$$y = 4$$

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

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

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

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

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

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

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

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

$$e \Rightarrow 4$$

$$d \Rightarrow 16$$

$$c \Rightarrow 28$$

$$b = 40$$

$$a = 52$$

$$\underline{140}$$

$$5 = 140$$

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

$$5(6) - 2$$

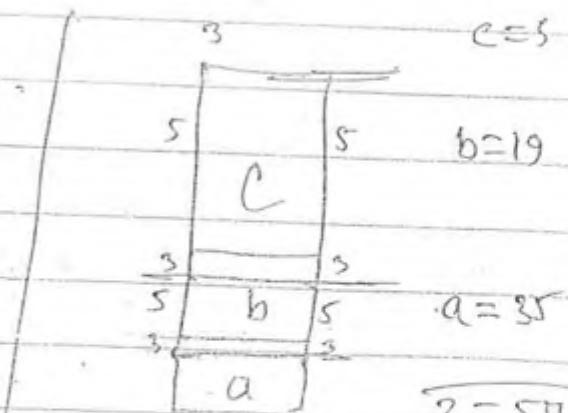
$$= \underline{\underline{20}}$$

DETA	Page No. _____
	Date: _____

~~XY~~

Let S be stack of size n , starting with empty stack. Suppose we insert first n natural numbers in sequence and then we perform n -pop operations. Assume that push and pop operations takes x seconds each and y seconds. Elapse betw the end of one such stack operation and the start 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) $y - 2x$

~~SA~~Push or Pop = $\int 2x$ Push - Push = $\int 0$ Push - Pop = $\int y$ Pop - Pop = $\int 0$ 

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

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

Observation (POP)

- No need to Pass n
- Delete
- Decrement
- need of returning element

Void Push (S, N, Top, u)

```

    {
        if (Top == N-1) → Isfull();
        {
            Printf ("Stack is overflow");
            exit (1);
        } → abnormal termination
    }
    else
    {
        Top++; → O(1)
        Constant Time
        S[Top] = u; → S[+Top] = u ✓
        {
            S[Top] = u; → S[Top+] = u ✗
        }
    }
}

```

P O P (Top)

```

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

```

S	.
4	.
3	d
2	c
1	b
0	a

N=6

(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)

```

    {
    }
    {
        Top++;
        S[Top] = a;
    }
  
```

4	e
3	d
2	c
1	b
0	a

TOP = N-1 .

DELTA	Page No. _____
	Date: _____

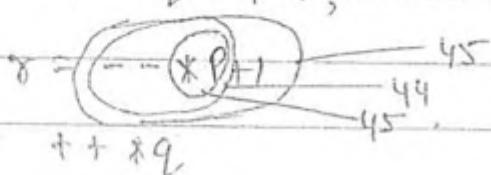
Q) int main()

int m = 44;

int * p = & m;

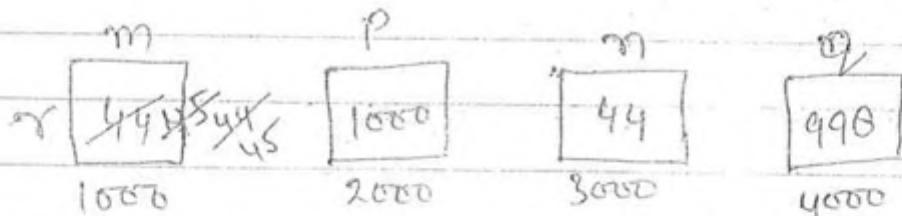
int & r = m; ~~(r address is same as m address)~~
int n = (*p)++; means r is alias to m.

int * q = p - 1;



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

Sol:



O/P 2.

a) 44, 46, 45

b) 45, 44, 45

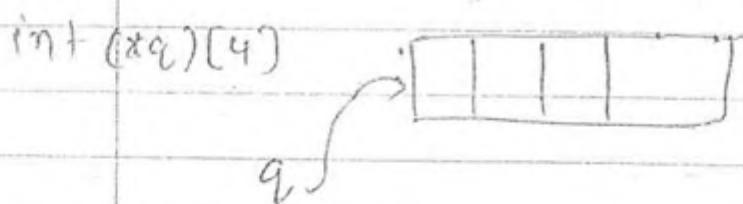
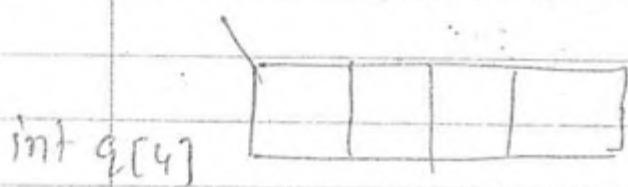
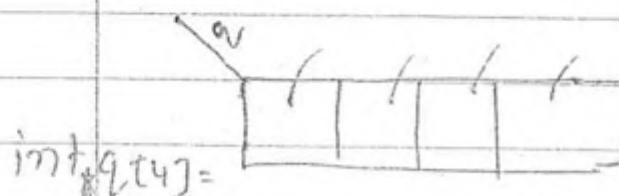
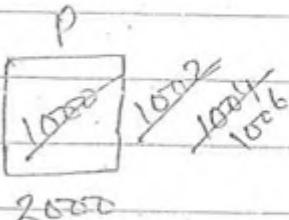
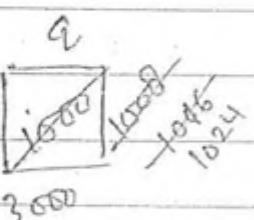
c) 46, 44, 46

d) 46, 45, 46

2, 9, 0, 6}

SOL

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{1}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{24}$	$\frac{1}{48}$	$\frac{1}{96}$	$\frac{1}{192}$
5	7	5	9	4	6	3	1	2	9	0
0	1	2	3	4	5	6	7	8	9	10



0/8	a+1	p	q.
i=0	1000	1000	1000
i=1	1008	1062	1008
i=2	1016	1004	1016
i=3	1024	1006	1024

DELTA	Page No. _____
	Date: 8 9 11

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[i])

p++;

q++;

}

}

Soln

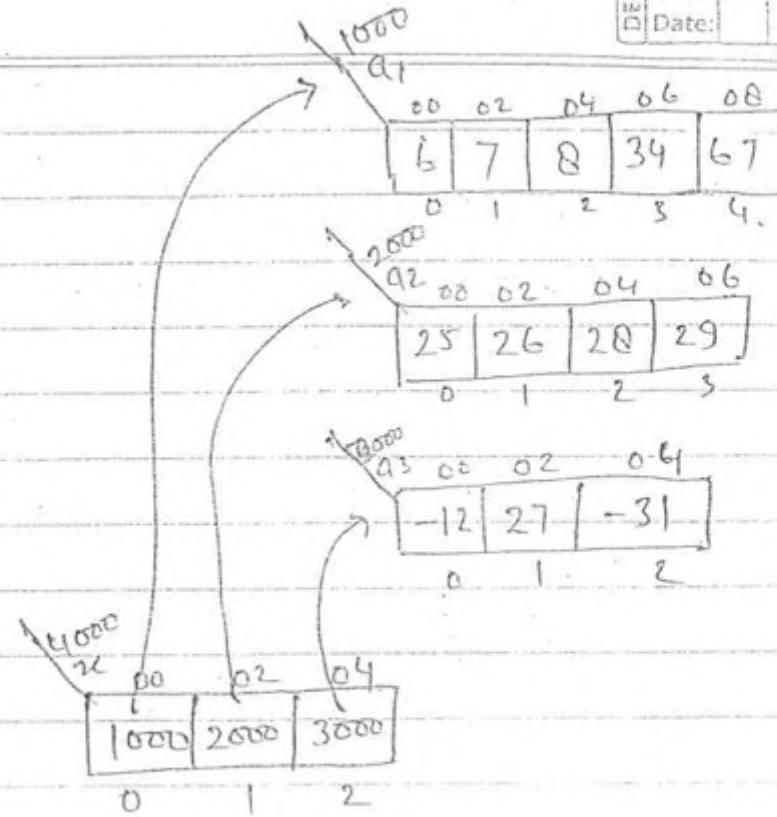
a

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

a[0] = 1000

a[1] = 1008

a[2] = 1016.



$$\checkmark \quad * (n(a+0)+2) = a[0][2]$$

$$\text{Ans} \quad \text{fun}(int \& a) \quad \begin{array}{l} 4000 \\ 5000 \\ \hline 4500 \end{array} \Rightarrow *++a[0] \\ \Rightarrow * (4+0)$$

$$\Rightarrow a[0][2] = n \frac{(a+0)+2}{4500} \\ \frac{1000+4}{1004}$$

$$\Rightarrow *++a[0] \\ \Rightarrow * (4+0)$$

$$\frac{4}{4} \\ (4000+0)$$

$$4000$$

$$\frac{4}{4} \\ ++(1000)$$

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

$$\frac{4}{4} \\ 4002[0]$$

$$\frac{4}{4} \\ * (4002+0)$$

$$\frac{4}{4} \\ * 4002$$

$$\frac{4}{4} \\ * (4002)$$

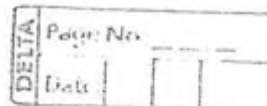
$$\Rightarrow * (a[2])$$

$$* (* (a+2))$$

$$\frac{4}{4} \\ * (4000+4)$$

$$* 3000$$

$$= -12.$$



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

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

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

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

main()

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

fun(int *a)

4000
4000
9
4000
5000

{
printf("%d", a[0][2]); } $\Rightarrow 8$

printf("%d", *a[2]); } $\Rightarrow -12$

printf("%d", *++a[0]); } $\Rightarrow 7$

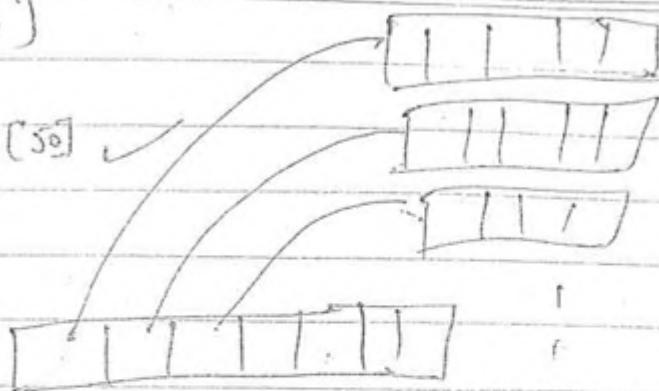
printf("%d", *(*(a+0))); } $\Rightarrow 25$

}

~~1~~

int * p[10]

int p[10][50]

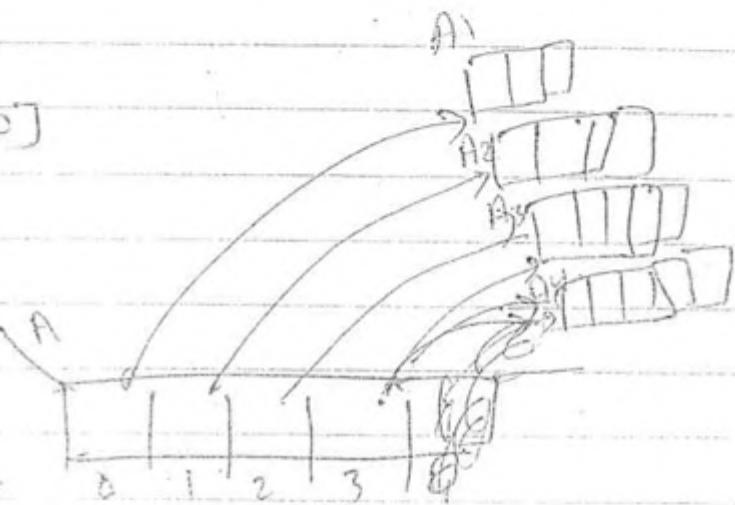
Setⁿ~~2~~

A [5] [100]

44

* A [5]

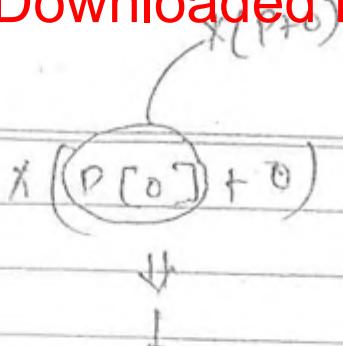
A



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

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

DELTA	Page No. _____
	Date: _____



$$\star(p[i]) = p[i]$$

Soln

i	T	$\star(\star(p+i)+j)$
0	0	1
1	2	
2	3	
1	0	4
1	1	5
2	2	6
2	0	7
1	1	8
2	2	9

In this Problem

P is 1D-Array

but it given
behaviour like
2-D Array

becoz

it given
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 $a[3][3] = \{1, 2, 3, 4, 5, 6, 7, 8, 9\};$ int * $P[3] = \{a, a+1, a+2\};$ for ($i=0$; $i \leq 2$; $i++$)

{

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

{

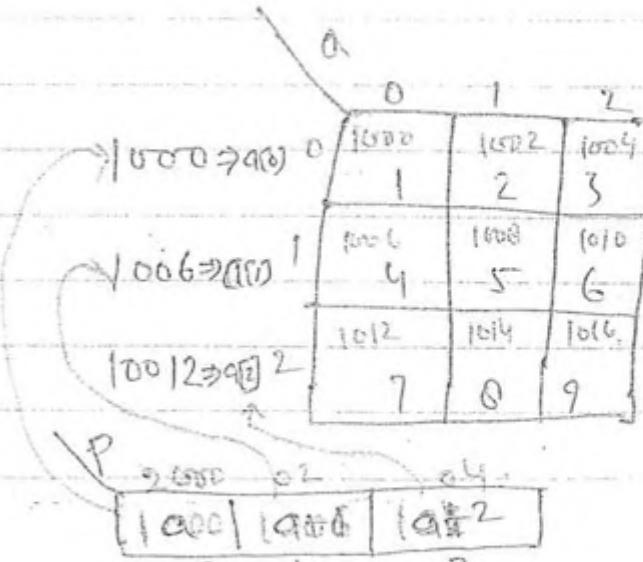
printf("%d", *(*(P+i)+j));

}

}

}

}

O/P $\Rightarrow 1, 2, 3, 4, 5,$
 $6, 7, 8, 9$ 

~~main~~

{

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

int *ptr;

ptr = n;

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

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

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

sd^m

		n[0]		
		0	1	2
n[0]	0	2	4	3
	1	6	0	5
1012	2	3	5	1

$\Rightarrow n[0] \Rightarrow 1000$
 $\Rightarrow n[1] \Rightarrow 1008$
 $\Rightarrow n[2] \Rightarrow 1012$

ptr

1000

* (ptr + 2)

1012 + 2

* (1004)

s

$$\frac{a[4]+1}{1016} \quad \downarrow \quad * \left(\frac{a[1]+1}{1016} \right) + 1$$

1020

#

$$a[2]+1$$

$$* \left(\frac{1024+4}{1024} \right) \rightarrow 1028$$

#

$$* \left(* \left(* \left(a+0 \right) + 1 \right) + 1 \right)$$

#

$$* [a+0] = * a$$

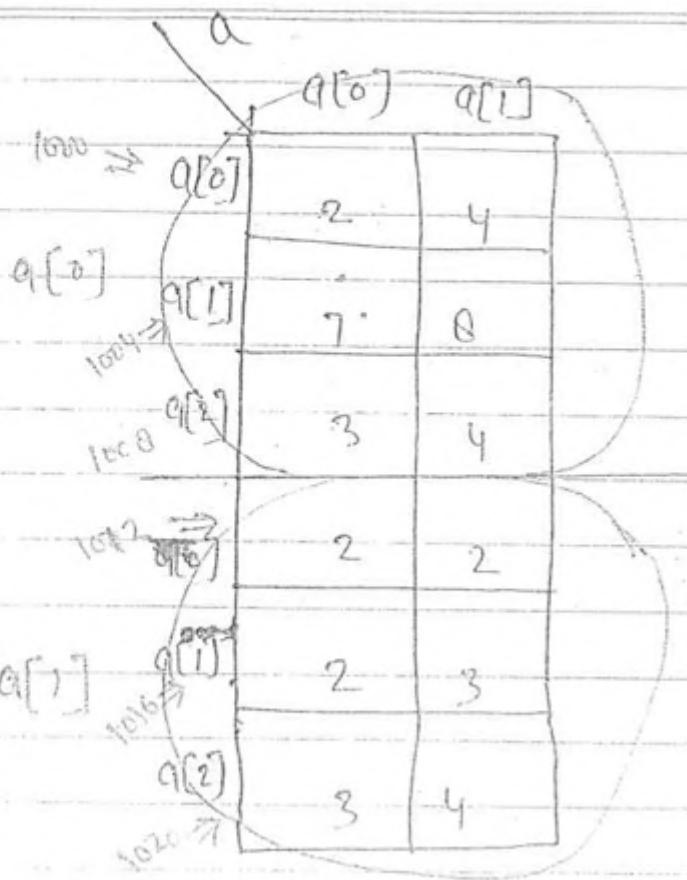
$$* * a = a[0][0]$$

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

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

Ans

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



$$a = 1000$$

$$(a+1) = 1012$$

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

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

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

$$\frac{*(*a[0]+1)+1}{20+1} \quad b$$

$$a[i][j]$$

$$21 \quad b[j]$$

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

$$*(*b+j)$$

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

$$*(*a[0])$$

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

```
# Main()
```

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

```
printf("%d%d%d%d%d", a[1][0], a[1][1], a[1][2], a[0][0], a[0][1]);
```

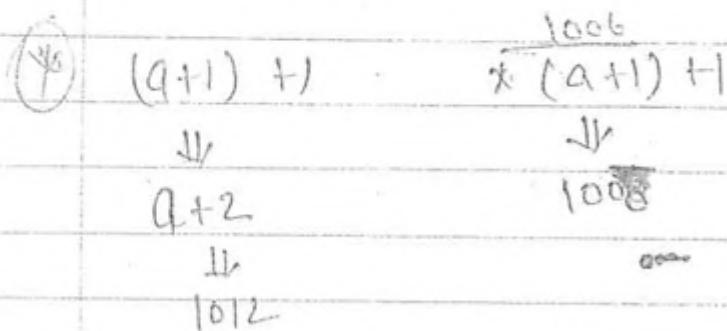
```
printf("%d%d%d%d", a[1], a[1], a[1], a[1]);
    printf("%d", a[1][0]);
```

```
printf("%d%d%d", a[1][0], a[1][1], a[1][2]);
    printf("%d", a[1][0]);
    printf("%d", a[1][1]);
    printf("%d", a[1][2]);
```

DELYA	Page No. _____
	Date: _____

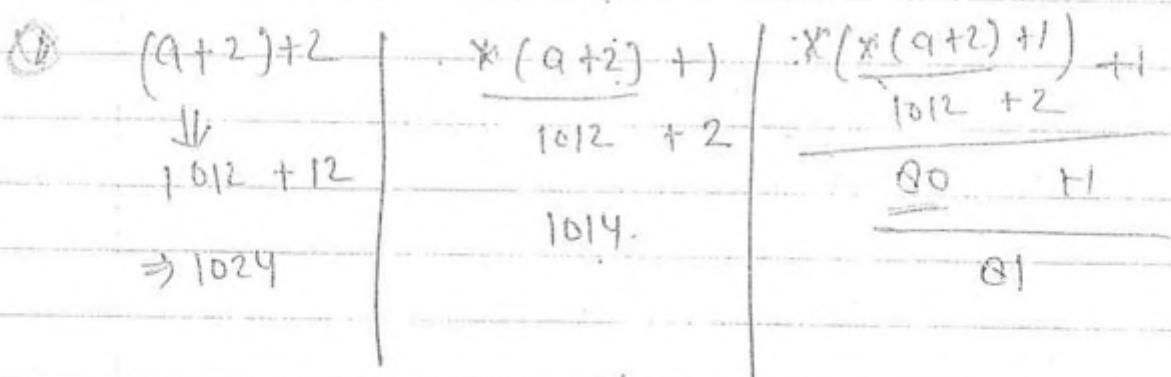


both are same but ~~$\&a$~~ $\&(a+1)$ means that is selected.



In 2D Array also there are complexity.

Even 1st $\&(base)$ indicate base address



$$\begin{array}{l} a \\ a+0 \\ \hline \end{array} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} 1000$$

$$\&(a+0) = a[0]$$

right

- * Between 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])));
```

{
a
1000}

O/P = 1

0 1 2			
0	00	02	04
1	10	20	30
2	06	08	10
3	40	50	60
4	12	14	16
5	70	80	90

$$\begin{aligned}
 a &= 1000 \\
 a[0] &= 1000 \\
 *(&a) &= 1000 \\
 \Rightarrow a[0] &\Rightarrow 1000 \\
 \Rightarrow a[1] &\Rightarrow 1006 \\
 \Rightarrow a[2] &\Rightarrow 1012
 \end{aligned}$$

$$a = 1000$$

$$a+0 = 1000$$

means In 2D array 0 rows skip.

$$a+1 = 1006$$

" " 1 row skip.

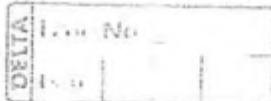
$$a+2 = 1012$$

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

$$a[0][0] \Rightarrow *(&a+0)[0]$$

$$*(\&(*(&a+0)+0))$$

~~0+1~~

Note

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

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

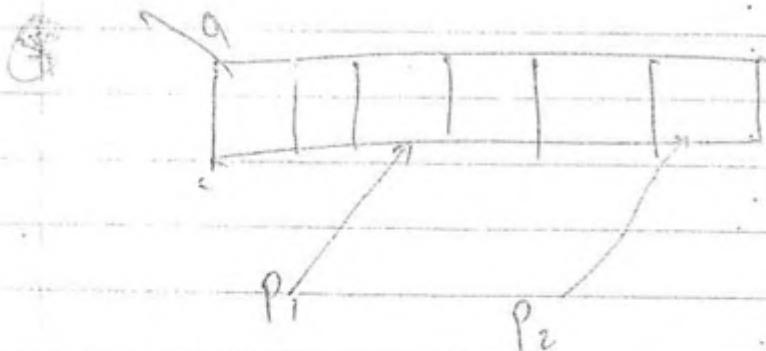
$x p \& r ++$ in this $p \& r ++$ is completed first.

Note

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

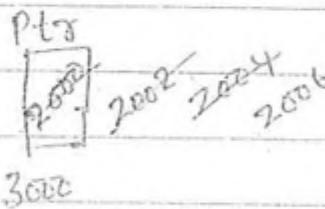
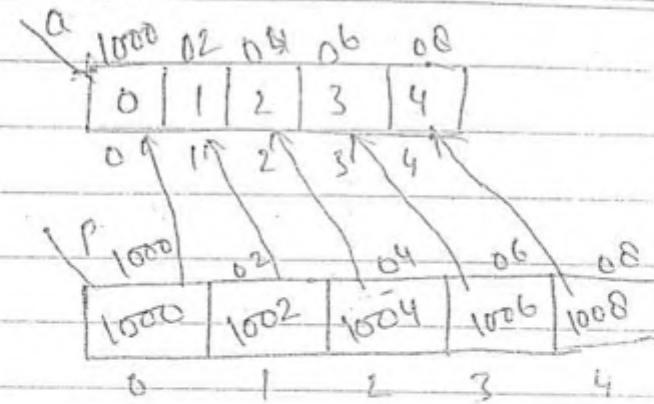
$p = p + i$ indicates p will points to after i elements from current place.

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



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

Subtraction betⁿ two pointers is allowed, but the condition is both with point will same array.

11
C++

#1 Main

{

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

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

int **pptr = p;

ptr++;

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

*ptr++;

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

*(++ptr);

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

++(*ptr);

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

}

#

Main()

{

char a[5] = "Visual C++";

[visual] [C++]

char *b = "Visual C++";

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

printf ("%d %d, sizeof(xa), sizeof(xb));

{

1

↓

↓

↓

↓

1

Sol

#2

Main()

int a[] = {0, 1, 2, 3, 4};

int xp[] = {a, a+1, a+2, a+3, a+4};

int *xptr = p;

printf ("%u %u %d", a, *a);

printf ("%u %u %d", p, xp, xptr);

printf ("%u %u %u %d", p, *p, xptr, **ptr);

}

2 00 1000 0

~~X~~

Main()

10 ²	2 ²	3 ²	4 ²	5 ²	6 ²
0	1	2	3	4	5

Char * str [] = { "Frogs", "Dogs", "Nuts", "Oil",
 "Croc", "Break" };

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

{

4

6x2

4

100

4

4422

12B.

4

2B.

~~X~~int a;

int a[5] =

10	20	30	40	50
0	1	2	3	4

size of a[10]

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

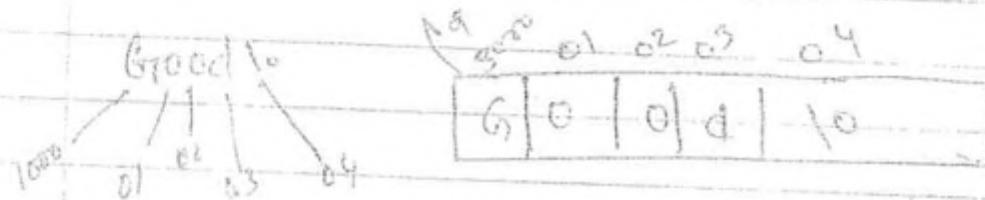
size of (a) = 10

DETA	Page No.	
	Date:	

~~QUESTION~~ Main()

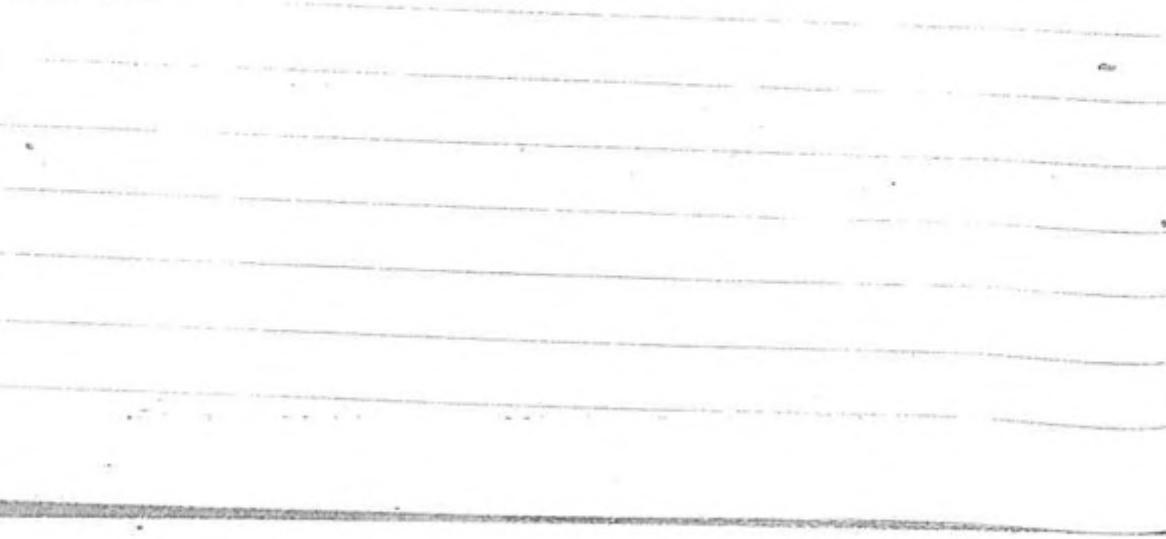
```

f
char xp = "Good";
char a[] = "Good";
printf("%d%d%d", sizeof(p), sizeof(xp),
       strlen(p));
printf("%d%d%d", sizeof(a), strlen());
}
  
```



Ans: 4 4 4 4

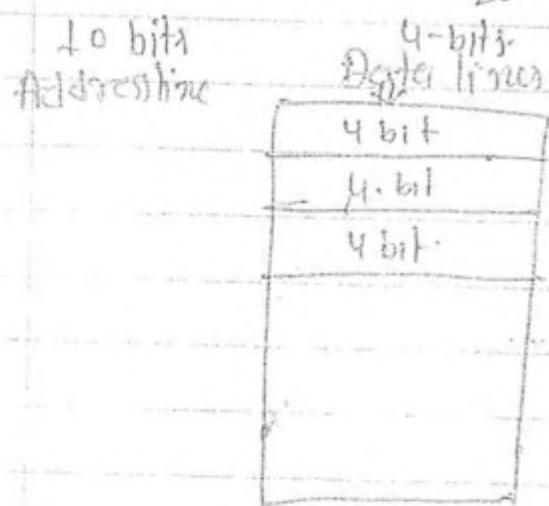
Ans: 4 4



int a = 10 a
 1000 |
 10 |

int *b
 b = &a b
 1000 |
 2000

char sum = 'c'; sum
 char *b;
 b = &sum b
 1000 |
 1000



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

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

DELTA	Page No. _____
	Date: _____

Call by need

* integer i=100, j=5;

Procedure P(integer x)

{

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

i = 200,

j = 20;

Printf (x); $\Rightarrow 105$

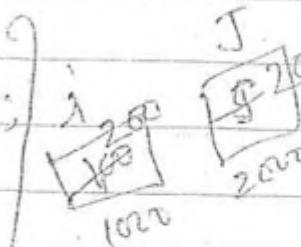
}

Main()

{

P(i+j)

}



exactly call by
same
but it will evaluate
the expression
only once.

Recently modified
thing will not be
affected.

Main() Assume:

Address line = 16 bit

char *p;

21

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

address

(register)

,

float x; ;

123456

Size of (b) = 2B

Size of (x) = 4B

* Consider the following Program . Call by name
Passing Parameters

integer i=100 , j=5

Procedure(p integer u)

begin : i+j

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

i = 200;

j = 20;

Printf (u); $\Rightarrow 220$

end ;

Main()

Call by Name :- 110, 220

{ p(i+j) integer i=100 , j=5 i=100 j=5 200

Procedure(p integer u)

begin int:i=5000

Print(u+i) = 5100

i = 200 ;
j = 20 ;

Printf (u) ; $\Rightarrow 120$

end ;

Main()

{ p(i+j)

}

DELTA	Page No.	
	Date:	

$$a = 1000, n = 6$$

~~(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)~~

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

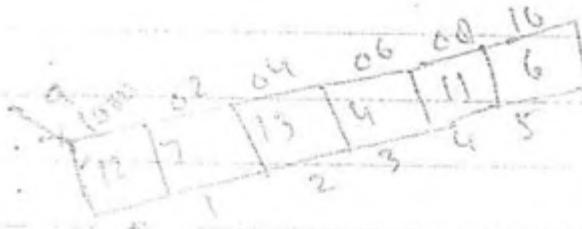
ing-C

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

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

main()

```
{
    int a[] = {12, 7, 13, 4, 11, 6};
    printf("%d", f(a, 6));
}
```



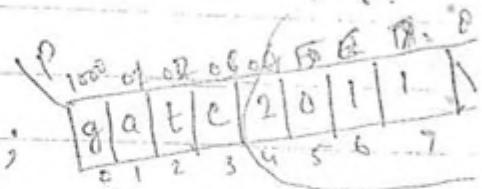
- a) -9 b) 5
 c) 15 d) 19

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

100 + e - a
100 - 101 - 97

c) e2011 d) 011

101 - 100 + 4
104

Q Consider the following C program.

int x;

main :

{

x=5
p(&x)



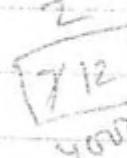
printf ("%d", x);

}

void g(int z)

{

z+=x



printf ("%d", z);

}

Void P(int *y)

{

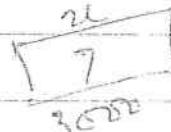
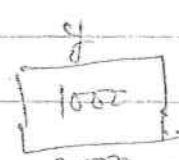
int $x = \frac{5}{2}y + 2;$

Q(x);

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

printf("%d", y);

}



O/P - ?.

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

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

~~Main () Call by name~~

{
int i=5, A[100]; i is
A[i]=15;

Printf ("%d%d", i, A[i]);
Swap (i, A[i]);

Printf ("%d%d", i, A[i]);

}

Swap (int c, int d)
i a[i]

{
int t;

t=c \Rightarrow t=5

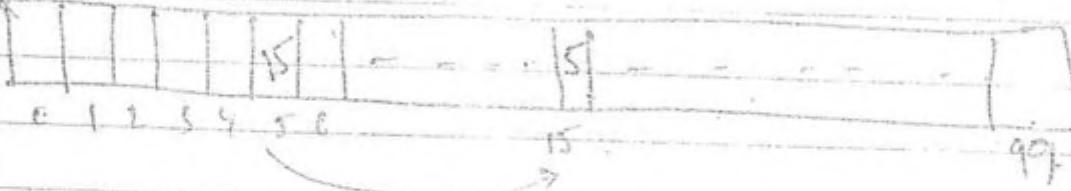


c=d; \Rightarrow i=a[i]

d=t; \Rightarrow a[i]=t;

}

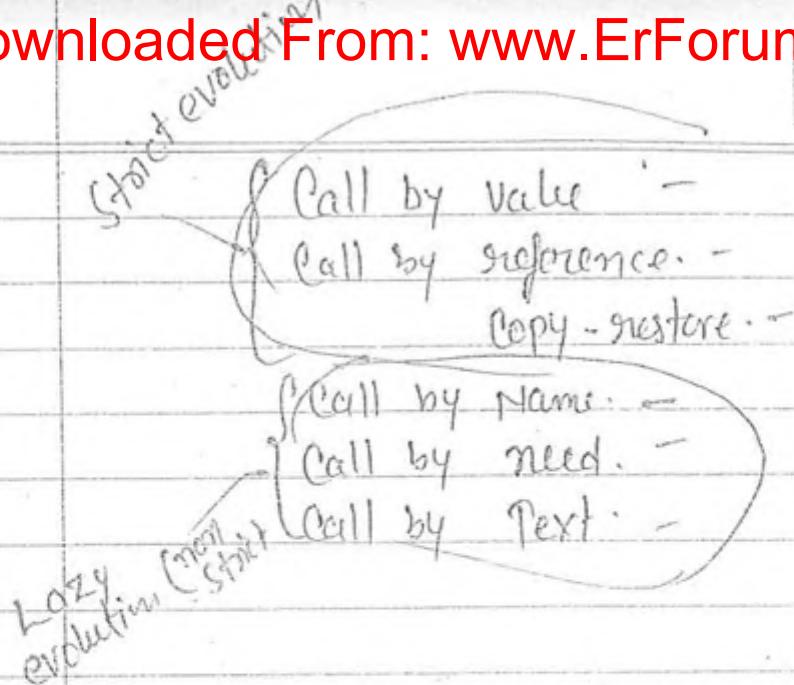
Sol



Call by name

O/P = 5, 15

15, 5



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

In the above program `c[m]` evaluated 5 times,

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

{ `c[m]` , `m` }

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

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

} in this
case `c[m]` 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 = 2;
```

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

```
swap (c[m], m);
```

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

```
}
```

Swap (int c , int d)

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

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

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

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

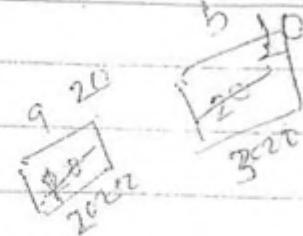
evaluated
by you

$$\begin{array}{r} 3 \\ \times 7 \\ \hline 21 \end{array}$$

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;

t = c; // t = a;

c = d; // a = b;

d = t; // b = t;

printf



Name conflict

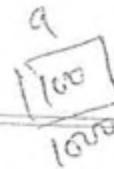
new local variable will
be proceed further
means existing local
variable will be ignored

C/P ⇒ 10, 20

20, 10

DELYA	Page No.	
	Date:	

- int a = 100
main()



{
int a = 10, b = 20



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

swap(a, b);

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

} swap(int a, int b)



{ int t;



t = c; \Rightarrow t = a; $a = 20$
 $b = 10$

c = d; \Rightarrow a = b;

d = t; \Rightarrow b = t;

} a + t

$a = 21, b = 10$

o/p

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

10, 20

10, 20

21, 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;

}

O/P \Rightarrow 10, 20 (without swapping)

\Rightarrow 20, 10 (after swapping)

DETA	Page No. _____
	Date: _____

8

What is the value printed by the following C program?

```
int f(int u int xpy, int &ppz)
```

```
{ int y, z;
```

u	94	ppz
147	11000	20000
4500	5000	6000

```
x & ppz = 1;
```

```
z = *x & ppz
```

```
x. ppy = z;
```

y	z
7	5
1000	8000

```
y = &ppz;
```

```
u + = 3
```

```
return (u+y+z);
```

```
}
```

void main()

```
{ int c, a, b, &a;
```

```
c = 4, b = &c, a = &b;
```

```
printf("%d,%d", f(c,b,a))
```

```
}
```

c	a
147	2000
1000	3000
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)

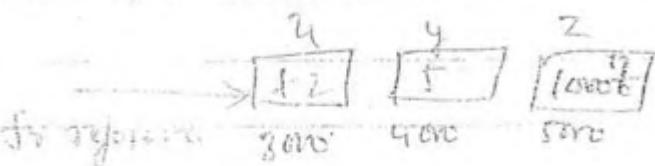
begin (int a, int b, int c)
 a=2 300 6000 5000

c=a+b; a b c
 end.

begin

int x=1, y=5, z=1000;
 p(x, y, z);

Print (x, y);
 end.



for before 2 6000 5000

call by value - 1, 5

copy-swipe - 2, 5

reference - 2, 5

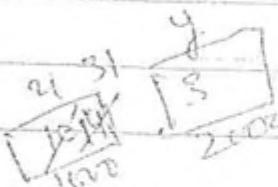
Q Consider the following C program.

Program P1()

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

```
fun1 (int a, int b, int c)
```

```
{
    a=y+y;
    b=y+y;
    c=y+y;
}
```



Aliasing

two variables
having same
address



Call by Value.

Copy Restore Reference

10, 3

14/27, 3

31, 3

Aliasing

Smart compiler

S

Main()

{
int a=5

Test. (a, a)

printf(a);

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

{
int

}

15 / 10
5 / 5x2c = d * 2 ; $\Rightarrow c = d + (d * 2)$ d = c - 3 ; $\Rightarrow d = d * (c - 3)$ {
}15 / 15-3
12 / 100O/P \Rightarrow 100

2

Consider the following Program.

8

Main()

```
{
    int a=5;
}
```

a
5
100

Test (x, a)

printf(a);

Test (int c, int d)

```
{
    c+= d%2;    => c = c + (d%2)
    d *= c-3;  => d = d * (c-3)
}
```

c
15
200

d
60
300

$$\left. \begin{array}{l} c = d \% 2; \Rightarrow c = c + (d \% 2) \\ d * = c - 3; \Rightarrow d = d * (c - 3) \end{array} \right\}$$

Call by Value

or Copy - Return

Call by Reference

eliding

i=4

5=++i

a[5]=++i

a[5]=6

a[5]=6

a[5]=6

a[5]=6

a[5]=6

a[5]=6

a[5]=6

or

a[5]=5

a[5]=5

a[5]=5

a[5]=5

a[5]=5

a[5]=5

3. Call by Copy - Restore [Indirect]

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;
    }
    
```

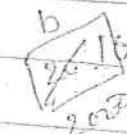
O/P - 10, 20
20, 10

8 Main()

int a=10, b=20;



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

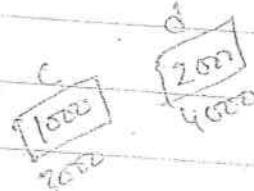


swap(a, b);

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

{ a 1000 b 2000 }

swap(intc, intfd);



exchange(c, d);

exchange(intc, intfd);



intc;



intfd;

intf=t;

{ }

L1P = 10, 20

20, 10

C language by default follows call by value
 C supports both call by value and also
 call by reference;

Swap without temp Variable.

$$a = 10, b = 20$$

$$a = a + b$$

$$b = a - b$$

$$a = a - b$$

Directly changes in original

Call by Reference

Main ()

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

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

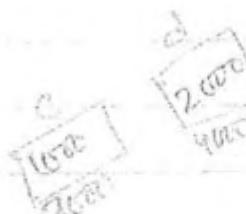
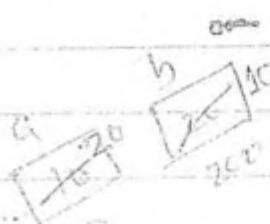
swap (a, b);

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

}

swap (int &a, int &b)

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



Out - 10, 20
20, 10

(in)
(out)

so
args are
place only

ie.

without swap
swap!

#8

Write a recursive C-program to find the \max^m element of the given array.

~~SJM~~

10	20	30	40	50
1	2	3	4	5

10, 20, 30, 40, 50
 | J

Point max (a, i, J)

```

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

```

eg

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

Types of Recursion

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

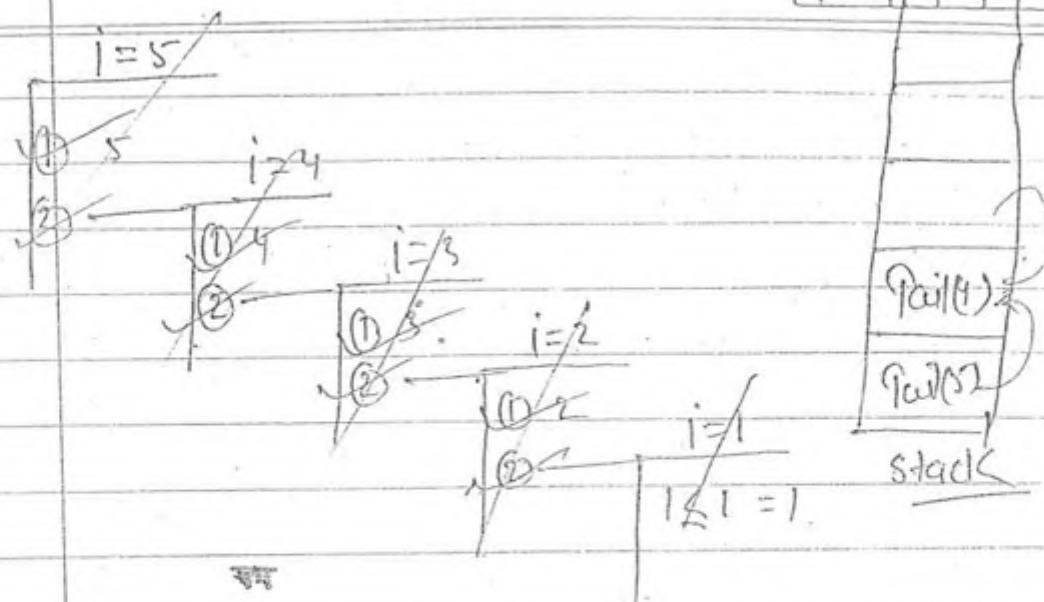
Tail-Recursion

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.

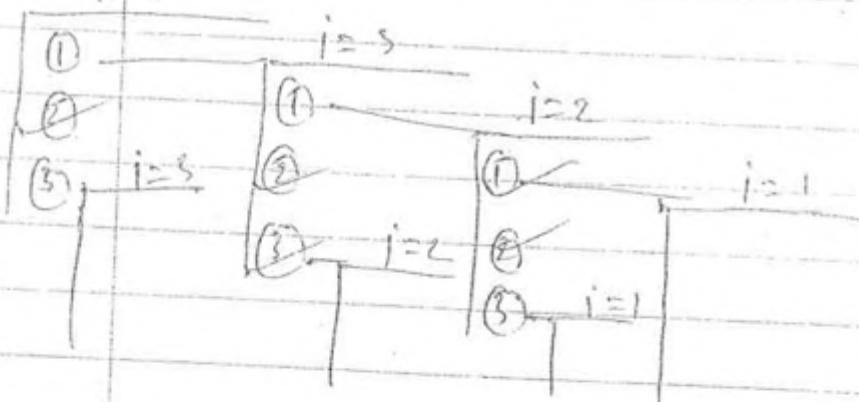
Non-Tail Recursive

Non Tail (int i)

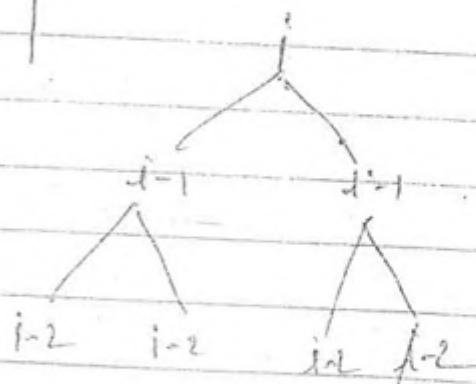
```

    {
        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.

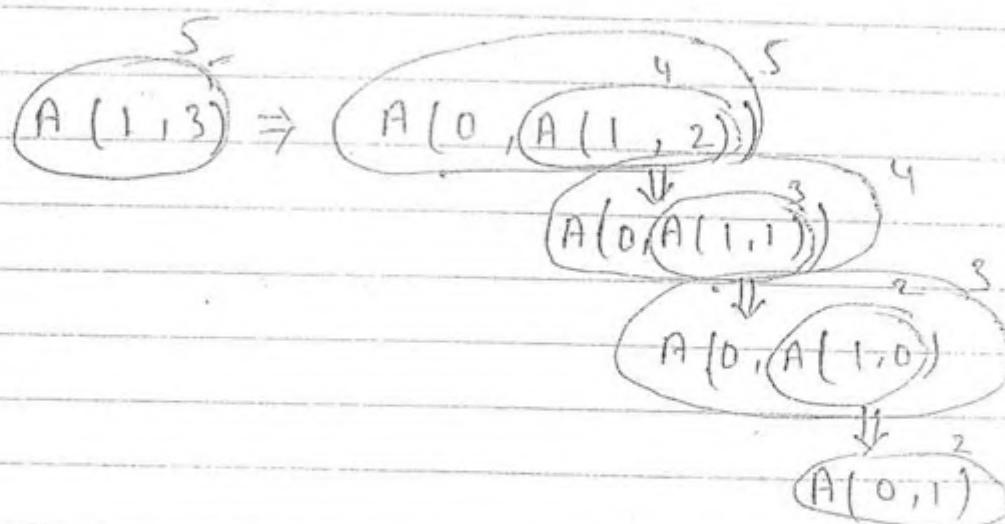
Note

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 $\rightarrow 7$

$$A(2, 2)$$

$$A(1, A(2, 1))$$

$$A(1, A(2, 0))$$

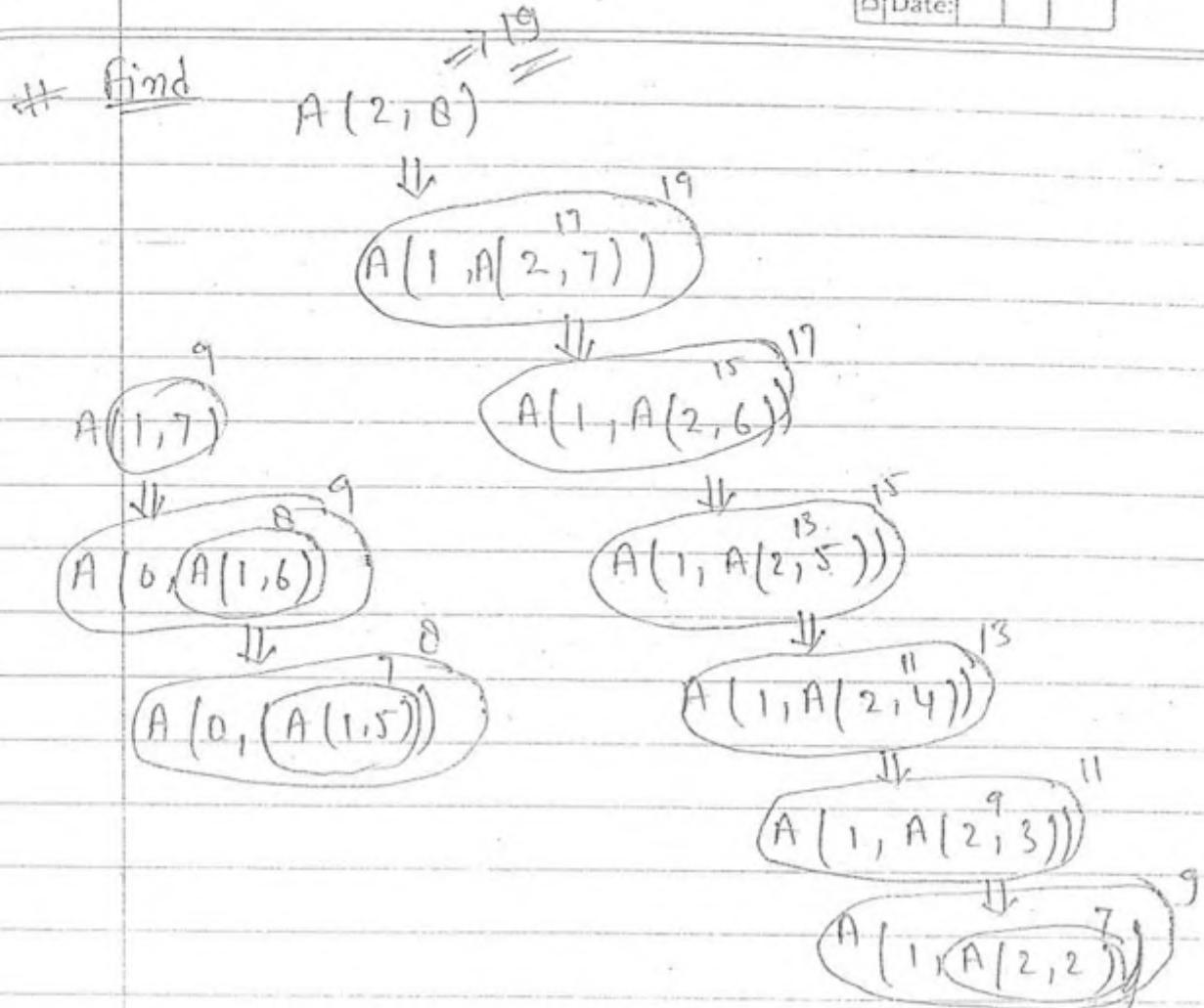
$$A(1, \emptyset)$$

$$A(A(A(A(1, 0))))$$

$$A(1, 5)$$

$$A(0, A(1, 4))$$

$$A(0, A(1, 3))$$



Indirect Recursion

Aorder(int i)

Let
 $I = 3$

```

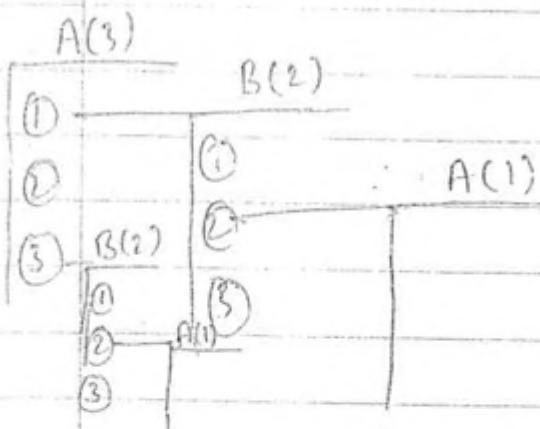
  {
    if ( i <= 1 ) return ;
    else
      {
        ① Border( i-1 );
        ② printf( "%d", i )
        ③ Border( i-1 )
      }
  }
  
```

Border (int i)

```

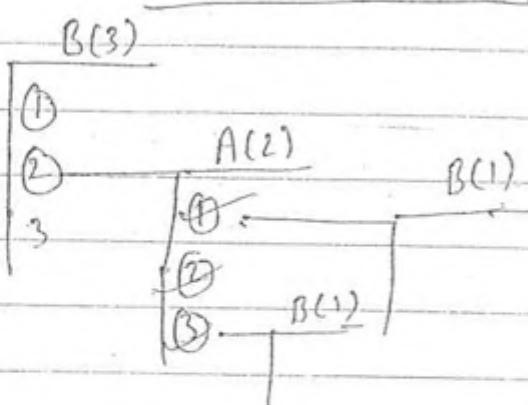
    {
        if (i < 1) return;
        else
        {
            ① printf ("%d", i);
            ② Border (i - 1);
            ③ printf ("%d", i);
        }
    }
  
```

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 :- $a b c * +$

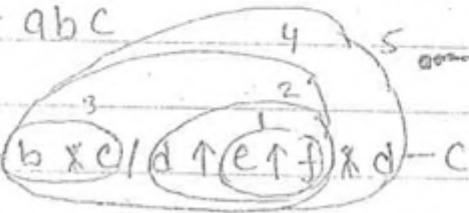
Prefix :- $+ a * b c$

Infix :- $a + b - c$

Postfix :- $a b + c -$

Prefix :- $- + a b c$

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



Postfix :-

$$a \frac{b c}{3} \frac{d e \uparrow f}{1} / \frac{g}{2} + c - b +$$

4

Prefix :-

$$+ - + a \frac{* b c}{3} \frac{\uparrow d \uparrow e f}{2} d c b$$

4

~~✓ Infix :- $a+b/c * d - e \uparrow f * g + h \uparrow i \uparrow j \uparrow k \uparrow l \uparrow m \uparrow n \uparrow o \uparrow p \uparrow q \uparrow r \uparrow s \uparrow t \uparrow u \uparrow v \uparrow w \uparrow x \uparrow y \uparrow z$~~

Postfix :- $a+b/c * d - e \uparrow f * g + h \uparrow i \uparrow j \uparrow k \uparrow l \uparrow m \uparrow n \uparrow o \uparrow p \uparrow q \uparrow r \uparrow s \uparrow t \uparrow u \uparrow v \uparrow w \uparrow x \uparrow y \uparrow z$

$a + t_5 - t_6 + t_9 - e$

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

Postfix: abc/d*ef\g*x-hiab*\c*d/+e-

Prefix: -+--*/bcd*\uparrow efg/*\h\uparrow i\abcd-e.

NoteInfix: $a + b * c$

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 + 1 2

1 4

Many

Prefix

+ 2 * 3 4

H₂O

many

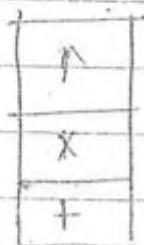
Note:-

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.

DELTA	Page No. _____
	Date: _____

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

* $a+b*c+d$



abcd ↑ * +

Top next
lower Higher \Rightarrow Push(next)

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



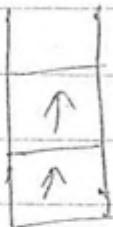
abcd ↑ * + -

Higher lower \Rightarrow Pop(Top)
L-R \Rightarrow Pop(Top)
R-L \Rightarrow Push(next)

One

try

* a↑ b↑ c



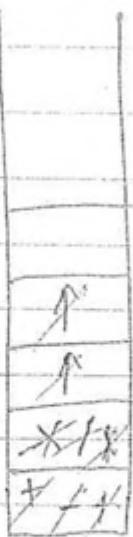
abc↑↑

* a+b-c



ab+ -

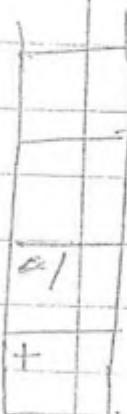
Prefix :- ~~a+b*c/d*efg*d-c+b.~~



PREFIX

abc*def11/d*c-bt

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

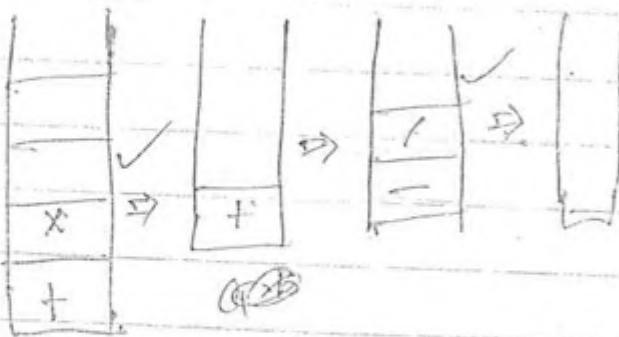


a b c / d * e + f * g + h * i + a b * c / d - e

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 operation $a + b * c - d / e$ into postfix.

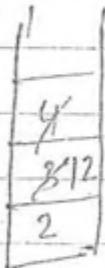


abc + de Max 2

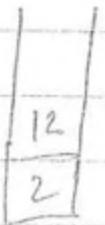
E/d-e

Postfix Evaluation

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

 $4 \Rightarrow \text{op}_2 \Rightarrow \text{Pop}()$ $3 \Rightarrow \text{op}_1 \Rightarrow \text{Pop}()$ $r = \text{op}_1 \text{ op } \text{op}_2$

Push(r)

 $12 \Rightarrow \text{op}_2 = \text{Pop}()$ $2 \Rightarrow \text{op}_1 = \text{Pop}()$ $r = \text{op}_1 \text{ op } \text{op}_2$

Push(r)



Postfix-

Postfix Evaluation

{
while (character is not null)

{
if (character is operand)
Push (character)

else

{
OP₂ = Pop ()

OP₁ = Pop ()

r = OP₁ character OP₂

Push (r)

}

}

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

} .



$$\begin{aligned} OP_2 &= 12 \\ OP_1 &= 2 \\ r &= 2 + 1^2 \\ \text{push}(r) & \end{aligned}$$

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

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

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

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

- ② all are R-L.

$$2 \times 2 - 1 \$ 1 \$ 4 - 2$$

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

Solⁿ

$$\begin{array}{|c|c|} \hline & * \\ \hline 7 & \times \\ \hline 4 & 32 - 4 \times 12 + \$ 3 \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline & 2^4 \\ \hline 2 & 16 \\ \hline 4 & (16)^3 = (2^4)^3 = 2^{12} \\ \hline 3 & \\ \hline 1 & = 4096 \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline & - \\ \hline & \times \\ \hline 2 & 221 - 2 \times 142 - \$ \\ \hline \end{array}$$

Q

Evaluate the following expression.

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

- a) 32^{30} b) 16^3 c) $\cancel{49151}$ d) 173458 .

Sol



$$123*45*6*$$

Q

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

The Postfix expression corresponding to the following Infix expression.

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

$\cancel{a) abc * + defln -}$ b) $a b c * + d e n f -$

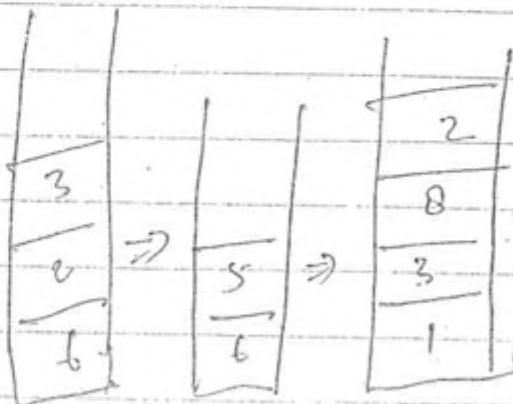
c) $a b + c * B - e n f n d)$

Q

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

6 2 3 + - 3 8 2 / + *

Soln



Ans

ans

10

Prefix to Postfix Conversion

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

(1)



$a b c d e \times b + - c -$

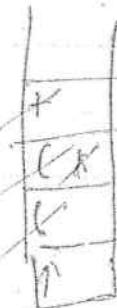
(2)

(3)

(4)

(5)

$c \uparrow (a + b) * d)$



$c a b + d \times \uparrow$

(6)

Prefix

Postfix

$$\textcircled{1} \quad a \text{ (minimal)} \longrightarrow a$$

$$\textcircled{2} \quad + \underline{a} b \longrightarrow ab+$$

$$\textcircled{3} \quad * \underline{a} \underline{b} c \longrightarrow abc**$$

$$\textcircled{4} \quad + \underline{xab} / cd \longrightarrow \textcircled{16} ab*c d/ +$$

$$\textcircled{5} \quad + - * \underline{(abc)} def \quad \downarrow \quad + - * \underline{(abc)} def$$

$$ab/c*d - ef +$$

$$\textcircled{6} \quad * \underline{\underline{abc}} - \underline{\underline{abc}} + ab$$

$$\downarrow \\ ab*c/d - ab*c/d + ab+ - *$$

$$* \underline{(abc)} - \underline{(abc)} + ab$$

Prefix* $a + b c$

↓

Postfix: - $a b c + *$ Infix: - $a * (b + c)$ * $a[6] = "abcde"; \quad S_1 = "ab"$ Substr (a, 1, 3, temp)
bcd $\quad S_2 = "cd"$ Strct (S₁, S₂)

1 = Is operand (Symbol)

0

PretoPost (Prefix, Postfix)

{
 + abcs
 4* + ab - cd\n
0 1 2 3 4 5 6 7

length = strlen (Prefix)

if (length == 1)

{



Postfix[0] = Prefix[0];

Postfix[1] = '0';

return (Postfix);

}

else

```

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

```

Substr(Prefix, 1, length-1, temp)
 $m = \text{find}(\text{temp})$

Substr(Prefix, 1, m, opnd₁)

Pre-to-Post (opnd₁, Prefix₁)

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

$n = \text{find}(\text{temp})$

8th Substr(Prefix, m+1, m+n, opnd₂)

Pre-to-Post (opnd₂, Postfix₂)

strct(Postfix₁, Postfix₂)

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

strcpy(Postfix₁, Postfix) $\xrightarrow{ab+}$

$\{ ab+cd- \quad ab+cd- \}$

$\{$

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

$\text{find}(\text{str})^o$

{
 $\text{ab}-\text{cd}$
 $\text{ab}-\text{cd}$
 $\text{ab}-\text{cd}$

$\text{length} = \text{Strlen}(\text{str})$
 $\text{if}(\text{str}[0] == \text{operand})$

return(1);

else

{

$\text{Substr}(\text{str}, 1, 5, \text{temp})$

$m = \text{find}(\text{temp})$
 $\text{ab}-\text{cd}$

$\text{Substr}(\text{str}, m+1, length-1, \text{temp})$

$\text{find}(\text{temp})$

$b-\text{cd}$

return(m+n+1)

}

3

Fibonacci Series.

n	0	1	2	3	4	5	6	7	8	9	10
fib(n)	0	1	1	2	3	5	8	13	21	34	55

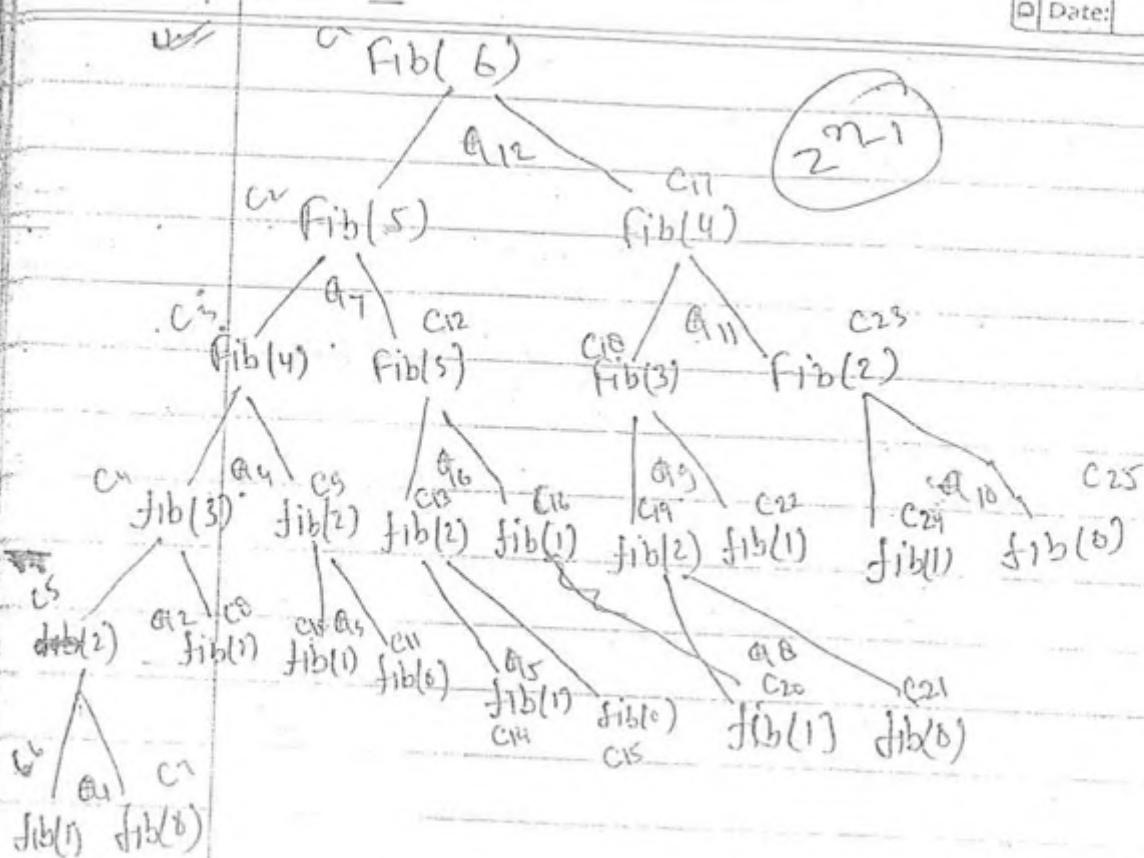
$$\text{fib}(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{otherwise} \end{cases}$$

fib (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

$$\underline{2^{n-1}}$$

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

P In fibonaci of 10,000 after how many funcⁿ call st addition taken place.

Solⁿ after 100th call ~~func~~ 1st addition will be done.

Q) Note

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

Q)

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)

Q)

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} .

Q)

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

$$\begin{aligned} f(0) &= 1, f(1) = 1, \text{fib}(2) = 3, \text{fib}(3) = 5, \text{fib}(4) = 9 \\ f(5) &= 15, f(6) = 25, f(7) = 41, f(8) = 67, \text{fib}(9) = 109 \\ f(10) &= 177. \end{aligned}$$

Q)

in fibonaci of n how many additions will be performed.

$$f(0) = 0, f(1) = 0, \text{fib}(2) = 1, \text{fib}(3) = 2, \text{fib}(4) = 4$$

$$f(5) = 7, f(6) = 12, f(7) = 20, f(8) = 35$$

$$f(9) = 54, f(10) = 88$$

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
	0 addition	1 addition	2 addition	3 addition	4 addition	5 addition	6 addition	7 addition	8 addition	9 addition	10 addition

Note

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

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

value

The no fem^n calls in fibonic of n .

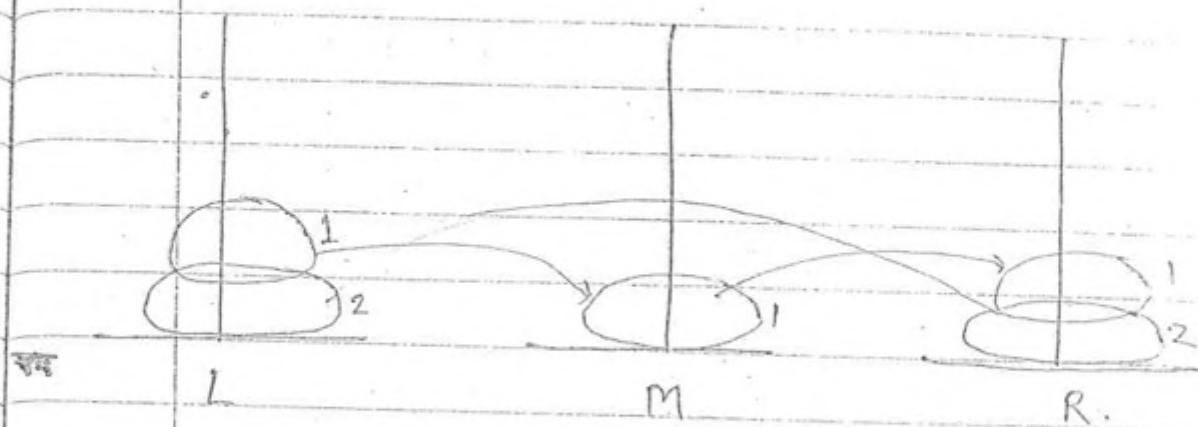
$$\text{function} = 2 * \text{fib}(n+1) - 1$$

value

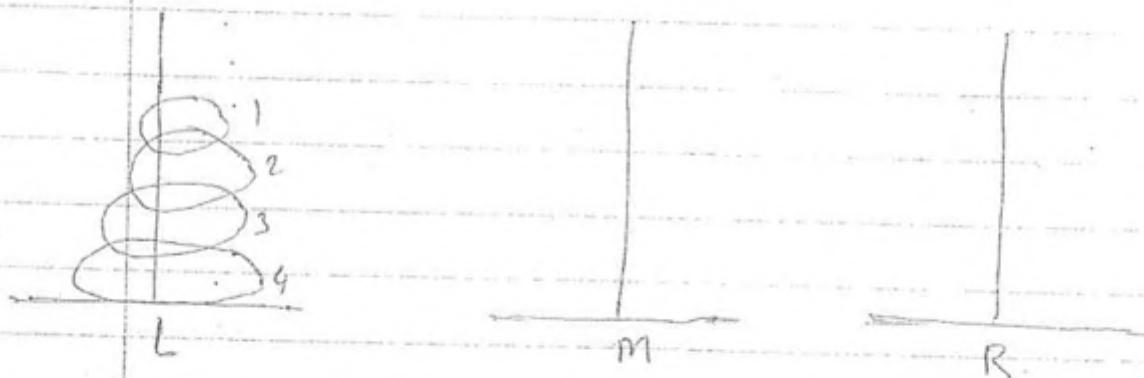
①

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.



$\text{TOH}(n, \text{L}, \text{M}, \text{R})$

- ```

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

```

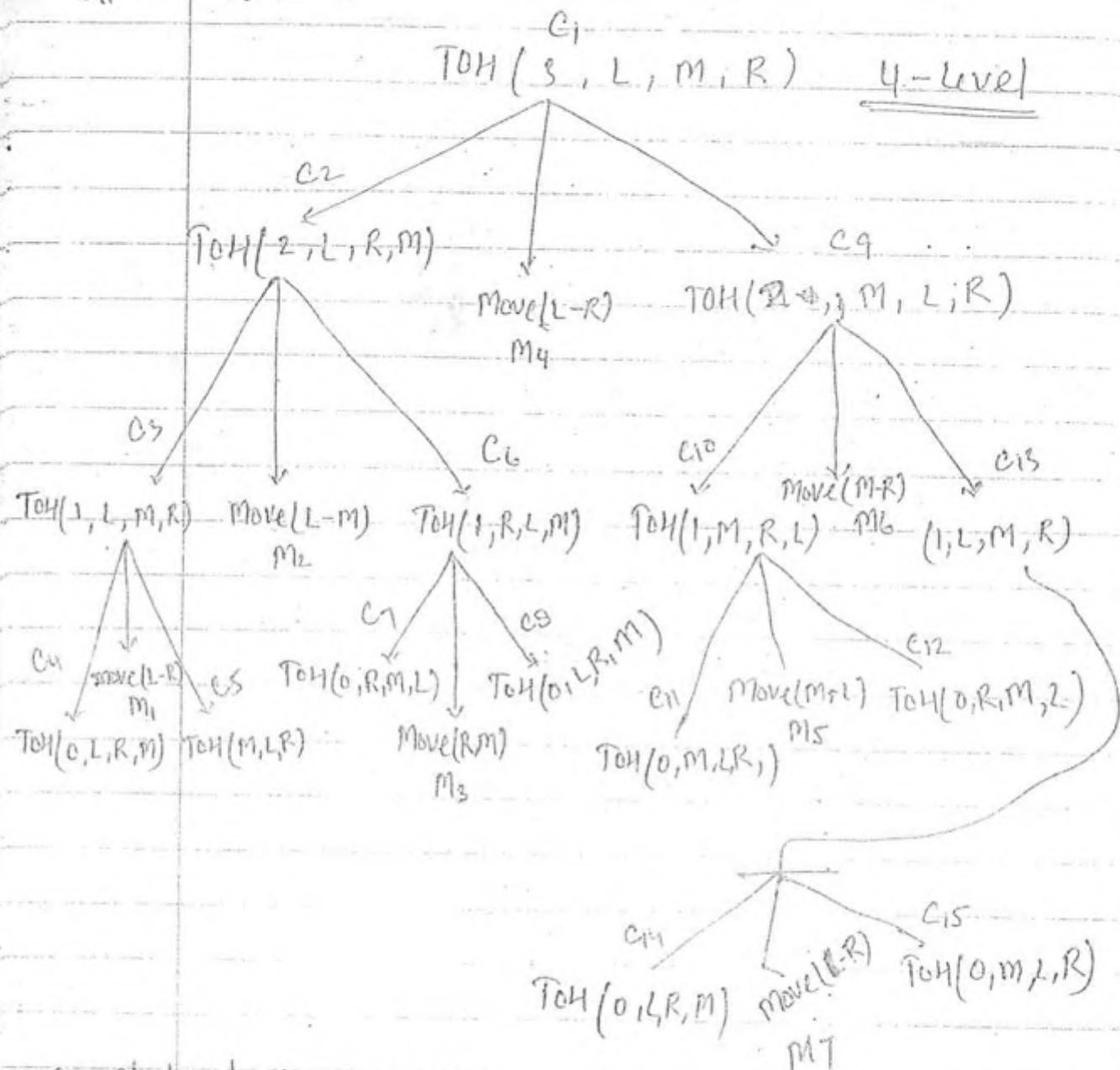
#

$\text{TOH}(n, \text{L}, \text{M}, \text{R})$

- ```

    {
        if(n==0) return;
        else
        {
            ①  $\text{TOH}(n-1, \text{L}, \text{R}, \text{M})$ 
            ② MOVE( L - R );
            ③  $\text{TOH}(n-1, \text{M}, \text{L}, \text{R})$ ;
        }
    }
  
```

#

 $n = 3$ 

no of level = 4

no of nodes = 2^{n-1} no of fun 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

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

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

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

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

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

Notes

The number of 2^n calls in $TOH(n)$

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

Q.

How many move are there in $TOH(n)$.

$$TOH(0) = 0 =$$

$$TOH(1) = 1 = (2^1 - 1)$$

$$TOH(2) = 3 = (2^2 - 1)$$

$$TOH(3) = 7 = (2^3 - 1)$$

$$TOH(4) = 15 = (2^4 - 1)$$

$$TOH(n) = 2^n - 1$$

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

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()  $\leftrightarrow$  int a
    int (*fp) ()  $\leftrightarrow$  int *b
    fp = display;  $\leftrightarrow$  b = &a
    (fp)();
    display();
    int display()
    {
        printf("Hi ");
    }
}
```

fp is a funⁿ which contains funⁿ address.

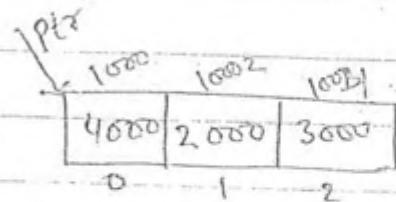
funⁿ name indicates
base address.

Q.

Consider the following Program.

Main()

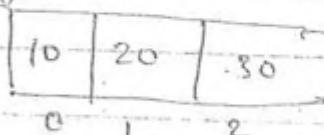
```
{  
    int (*ptr [3])();  
}
```



ptr[0] = aaa;

ptr[1] = bbb;

ptr[2] = ccc;



(* ptr [2]) () ;

{

4000

aaa()

{

printf ("aaa");

{

2000 bbb()

{

printf ("bbb");

3000 ccc()

{

printf ("ccc");

{

O/P : ccc

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

$\text{int } q[5] \rightarrow$ array of integer values.

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

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

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

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

$\text{int } (*a)[10]$

a is a pointer pointing to array of 10 elements

Q What does the following C statement declaration?

$\text{int } (\text{int } *f)(\text{int } x)$

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

f
 ↗ (int *)
 {
 ↗ return (2u)
 }

8

What does the following c Statement declare.

void.

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

Soln

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

- (1) integer.
- (2) def is a ptr to a funⁿ will return void.
- (3) which will return void.

8

What does the following c-Statement declare?

Soln

char * [p : (* a [s]) () ()]

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

Q) What does the following C statement declare:

$\text{char}^*(\text{x})[0](\text{x PTR}[N])()$;

Soln

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

Q

Consider the following C-program.

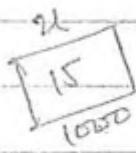
```
int main()
```

```
{
```

```
    int u=15;
```

```
    printf ("%d", fun (s, &u));
```

```
}
```



```
fun (int n, int *fp)
```



```
{
```

```
    int t,f;
```



```
    if (n<=1)
```



```
{
```

```
    *fp = 1;
```

```
    return 1;
```

```
}
```

$t = \text{fun}(n-1, fp)$

$f = t + *fp$

$*fp = t$

return $t+f$)

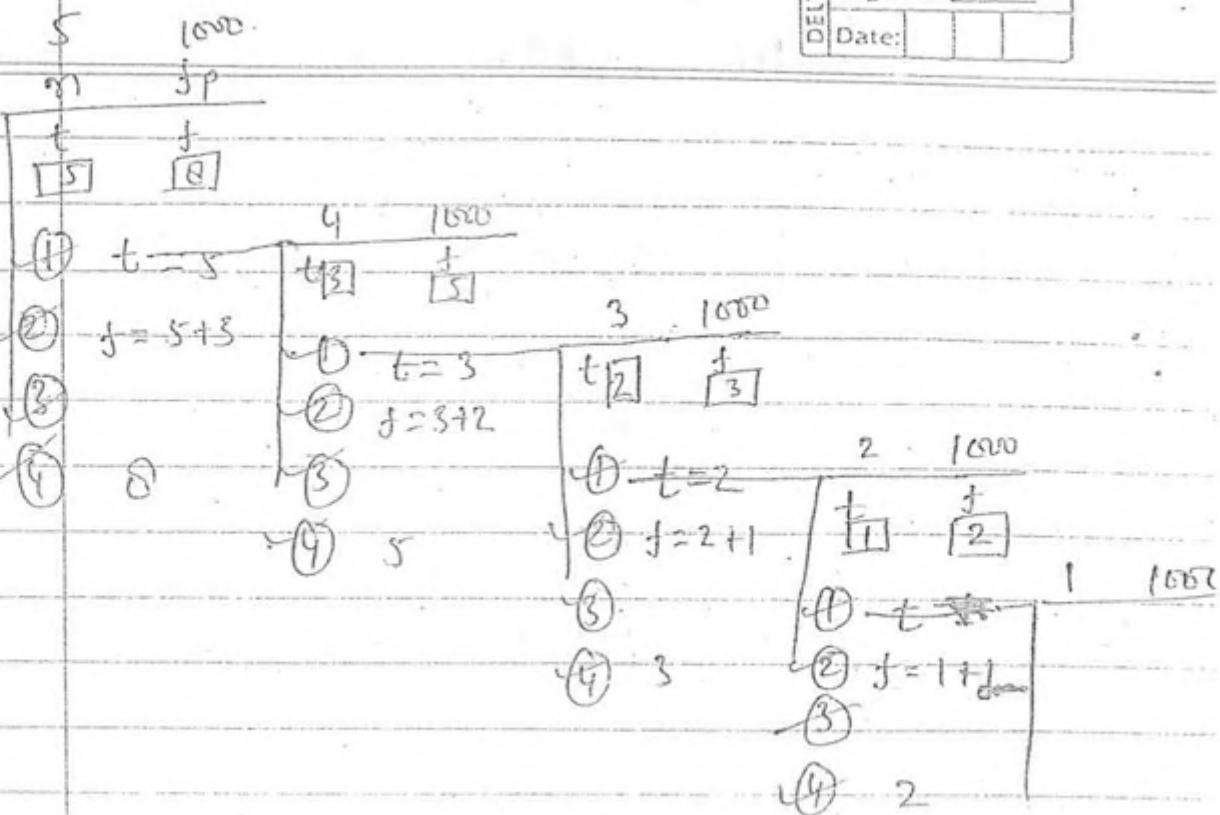
```
}
```

The value pointed is.

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

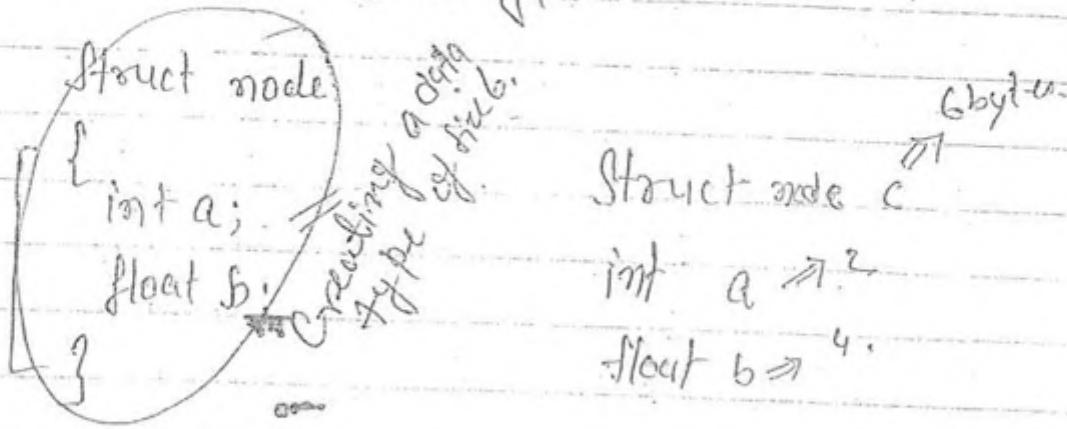
© Wiki Engineering

www.raghul.org



Structure

↳ User defined
Data type.

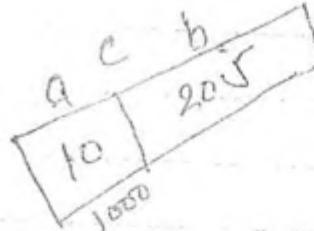


→ Struct node

```
{  
    int a ; // int a=10  
    float b ; // float b=20  
}
```

not allowed

Main ()
{



Struct node c = { 10, 20.5 } ;

printf ("%d\n", a);
g. select operator(.)

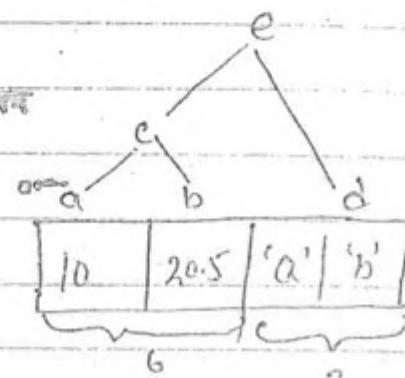
\Rightarrow Struct node \Rightarrow^b

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

6 bytes

Struct node₁ \Rightarrow^B

```
{
  struct node c;  $\Rightarrow^6$ 
  char d[2];  $\Rightarrow^2$ 
}
```



Main()

```
{
  struct node1 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() {

 {

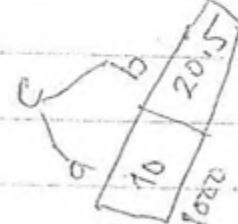
① Struct node c = { 10, 20.5 };

② Struct node *d;

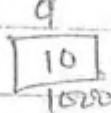
③ d = &c

Printf ("%d %f",

);



④ int q = 10;



⑤ int *x;



⑥ x = &q;

⑦ b = &a;



⑧ b

⑨ x b

d = 1000

*d = a, b

(*d).a = 10

(*d).b = 20.5

(*d).a \Rightarrow d \rightarrow a

(*d).b \Rightarrow d \rightarrow b

O/P :-

Main ()

{ struct a

{ char ch[7];

char *str;

};

struct b {

struct a

char *c;

struct a s1;

};

"bytes"

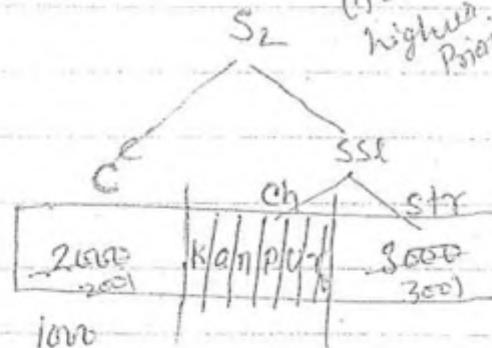
struct b s2 = { "Raipur", "Kapur", "Jaipur" };

printf ("%s.%s.%s", s2.c, s2.s1.str);

printf ("%s.%s.%s", +s2.c, +s2.s1.str);

(1) Dot have
higher
Priority

Raipur



O/P → Raipur . Kapur . Jaipur

 eipur , aipur

"Raipur."

2000
2000
2000

"Jaipur"

3000
3000

~~X~~ Consider the following C. Program.

Main()

```

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

```

6 by 10.

P

```

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

```

Struct st *pa = a;

int j;

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

```

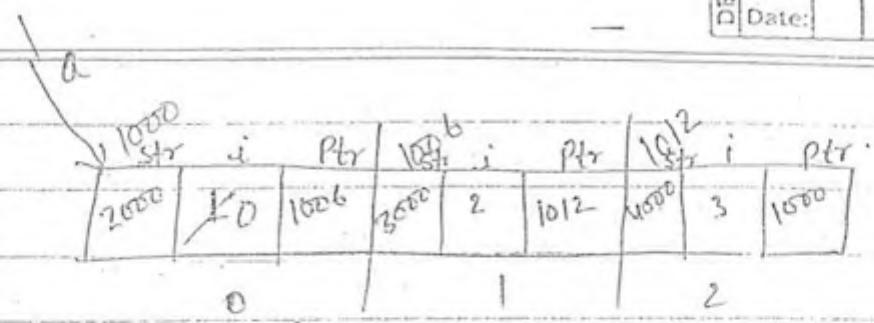
{
    Printf ("%d", a[j].i);
    Printf ("%s", a[j].str);
}

```

2001

}

O/P: 0; 1 2
Nagpur; Raipur; Kanpur



P

Main()

```

struct S1 {
    char x2;
    int i;
    struct S1 *p;
}

```

6 by 3.

① $a[0] \leftarrow \text{++}(\text{ptr} \rightarrow 2)$

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

③ $a[2] \leftarrow \text{--}(\text{ptr} \rightarrow p \rightarrow i)$

4000

```

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

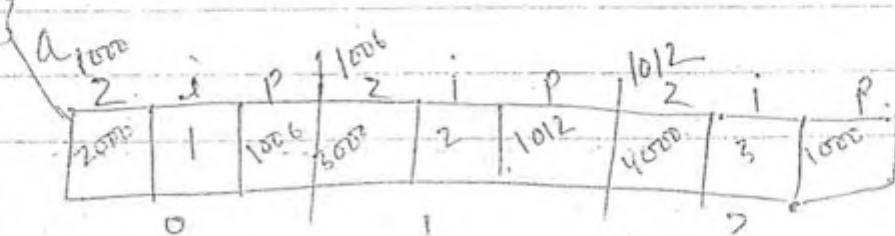
```

Struct S1 * ptr=a

$\text{printf}("%s", \text{++}(\text{ptr} \rightarrow 2)) \Rightarrow \text{agpur}$

$\text{printf}("%s", a[(\text{++}(\text{ptr})) \rightarrow 1].z) \Rightarrow \text{Kanpur}$

$\text{printf}("%s", a[--(\text{ptr} \rightarrow p \rightarrow i)].z) \Rightarrow \text{Kanpur}$



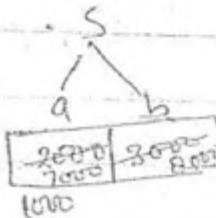
```
Typedef Struct P
{
    char *a, *b;
}
```

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

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

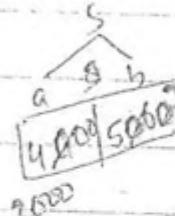
```
Main()
```

```
{
    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 $f_2(t \times p)$

{
 $p \rightarrow a = "V"$
 7000
 2000 4000
 3000 2000

$p \rightarrow b = "W"$; 8000

Printf ("%.5%.5", p->a, p->b);
 ↓ ↓
 v w;

3.

O/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
 w v v w v w u v

Y

B7F

Struct Test

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

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

Main()

```
{  

    struct Test *p = st;  

    pt = 1;
```

printf("%d.%c", ³⁰⁰(₃₀₀₀)(¹⁰⁰⁴→₁₀₀₄(¹⁰⁰⁴)));

printf("%d.%c", ³⁰⁰(₃₀₀₀)(¹⁰⁰⁴→₁₀₀₄(¹⁰⁰⁴)));

printf("%d,%d", ³⁰⁰(₃₀₀₀)(¹⁰⁰⁴→₁₀₀₄(¹⁰⁰⁴)));

printf("%d.%d", p - c);

}

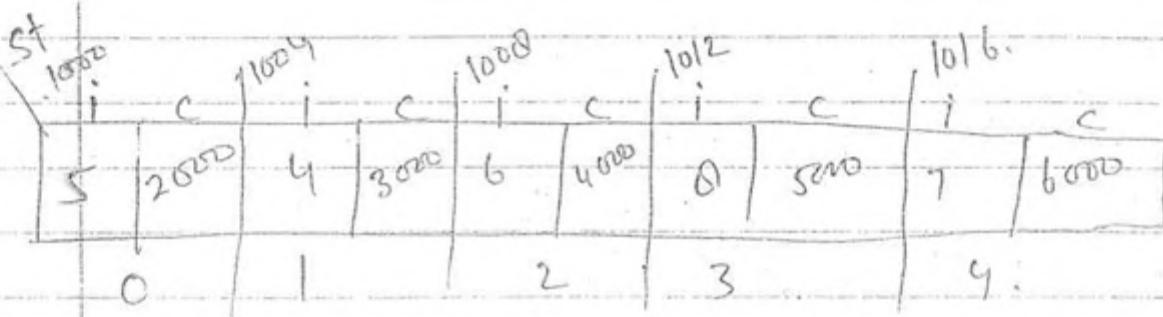
O/P :-

a) jungle, n, θ, mester

b) effe~~r~~, u, b, ~~w~~, ungle

c) ceHer, K, b, jungle.

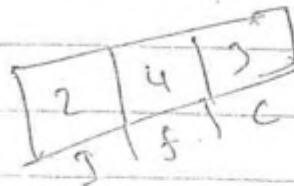
d) etter, u, θ, mester.



struct s

{
 int a; $\Rightarrow 2$
 float b; $\Rightarrow 4$

 char c = "abc"/c[3]; $\Rightarrow 3$



}
Size of (struct s) = 9.

Unions

U
 int a; $\Rightarrow 2$

 float b; $\Rightarrow 4$

 char c[3]; $\Rightarrow 3$

};



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

Union s1

U
 int a; $\Rightarrow 2$

 float b; $\Rightarrow 4$

 char c[3]; $\Rightarrow 3$

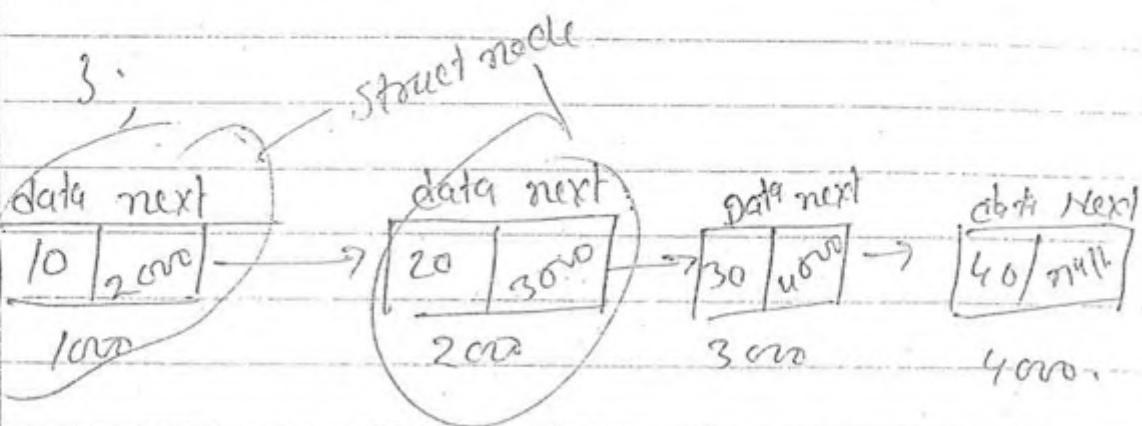
 struct s d; $\Rightarrow 9$

}
Size of (union s1) = 9.

~~S~~ Struct node

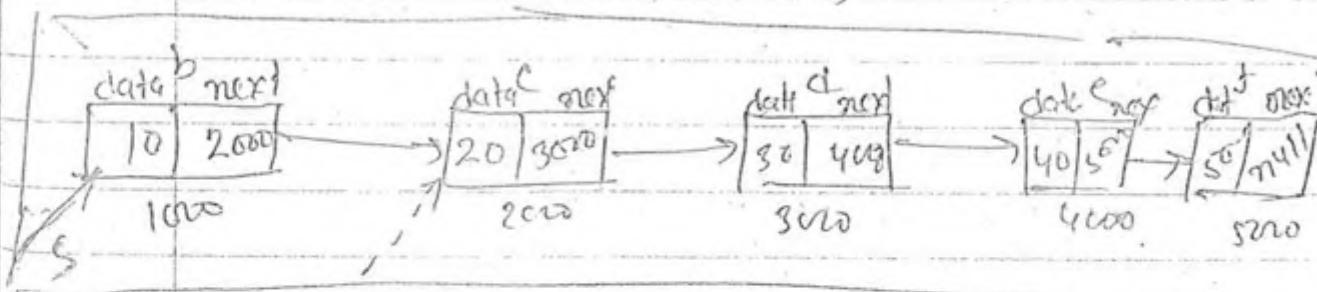
S
int data;

struct node * next;



self referential
data structure

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



b. data = 10

b. next = 2000

s++ ; 2000 X

↓
1004

$s = s + 1 \Rightarrow 2000 X$



1004 ✓

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

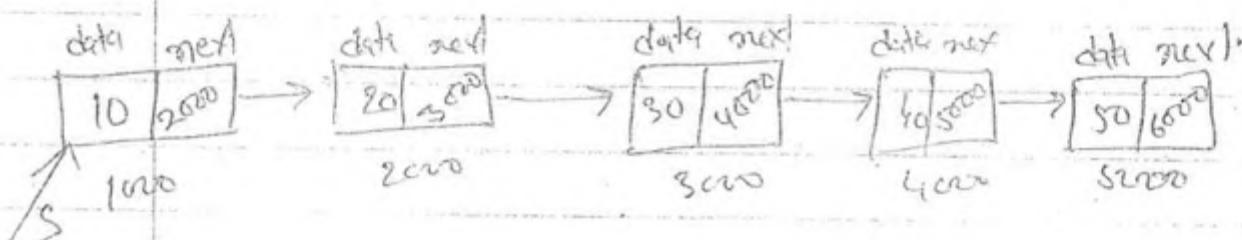


1004 X



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

80ⁿ



Struct node

s
1000

```

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

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

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

{

}

o/p $\Rightarrow 10, 20, 30, 40, 50$

Q

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

solⁿ

100

$s \rightarrow \text{data} \Rightarrow 10$

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

null

Dangling pointer

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

$s \rightarrow \text{next}$ { " " }

Print last (struct node *s)

{
if (s == null)

 return (-1);
else

 { while (s -> next != null)

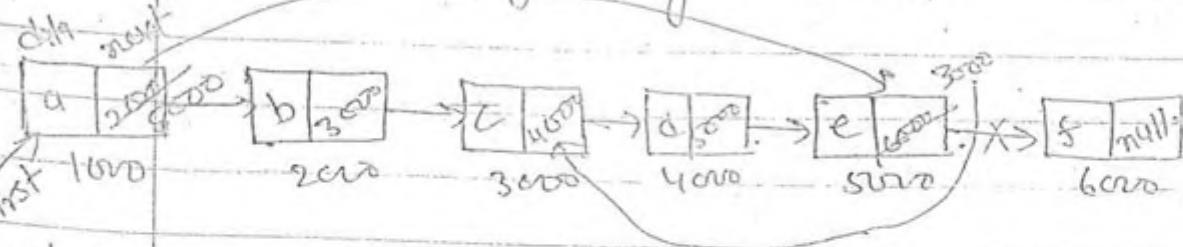
 { s = s -> next

 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

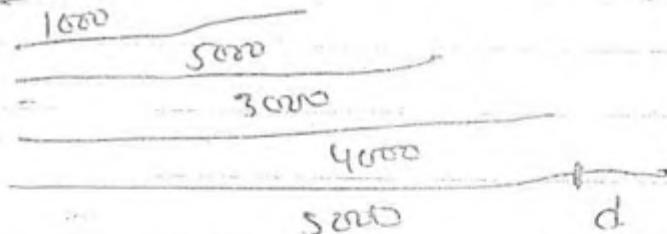
(iv) (p → next) → next → next) = p;

④

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

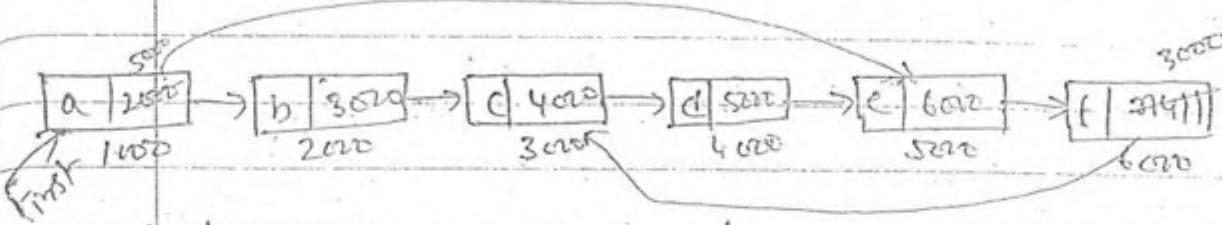
⑤

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

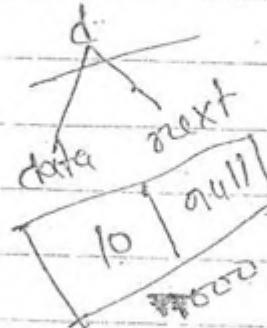
(5) $\text{printf}(\underline{\text{first} \rightarrow \text{next} \rightarrow \text{next} \rightarrow \text{next} \rightarrow \text{data}});$

$qp \rightarrow C$

DELYA	Page No.			
	Date:			

X Main();

Struct node d = {10, null};



}

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

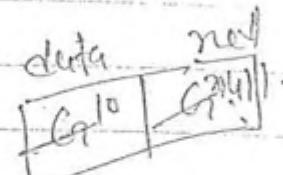
Stacks

Heap

Dynamic memory allocation means Heap

Stack

Malloc
memory allocation.



Struct node *P;

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

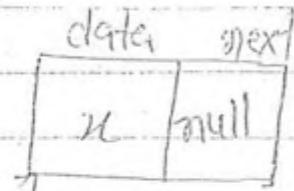
Typecasting

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

(*P).next = 1411 or P->next = 1411

return (P)

3



~~Getnode(u)~~

{
}

Struct node *p;

a

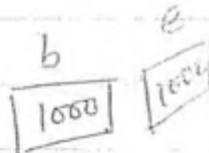


int *p
float *q
msg
msg

int *p
float q
msg
msg

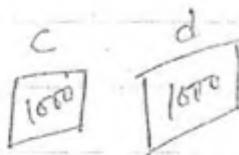
double a=10.5

int *b=(int*)&a

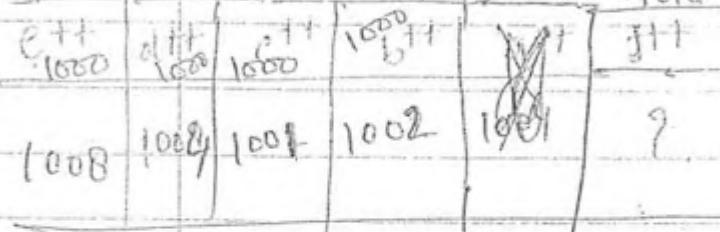


char *c=(char*)&a

float *d=(float*)&a

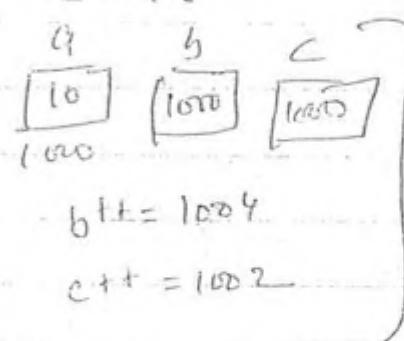


double float char int



int a=10
b=(float)a

int *c
int a=10
b=(float*)&a



~~XX~~

Create node (u)

{

Struct node *P;

data next

u null

1000

P

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

If (p == Null)

{

printf (" memory over")

exit(1);

}

else

{

p → data = u ;

p → next = null ;

return (p)

}

}

data next

p

1000

000

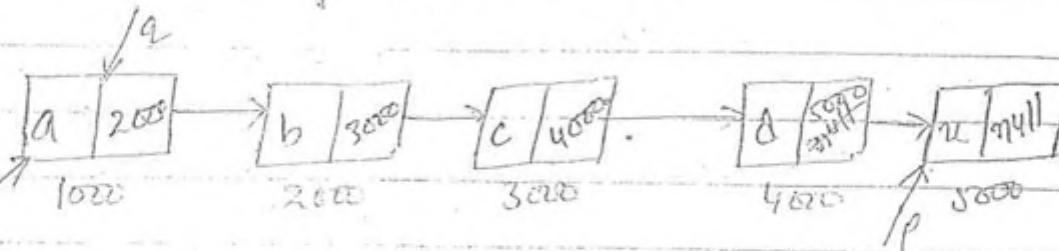
u 344

1000, 1001, 1002, 1003

000

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

~~Sol~~



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

{

Struct node *p, *q;

$q = s;$

$p = (\text{structnode}) \text{malloc}(\text{sizeof}(\text{structnode}))$

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

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

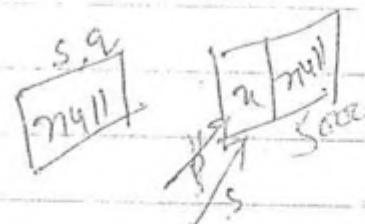
if ($q == \text{Null}$)

{

$s = p;$

return (s)

}



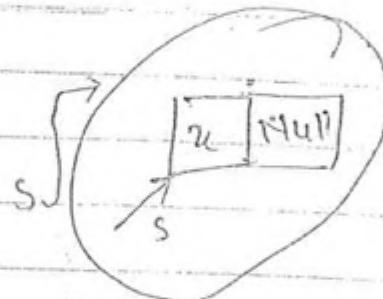
else

{

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

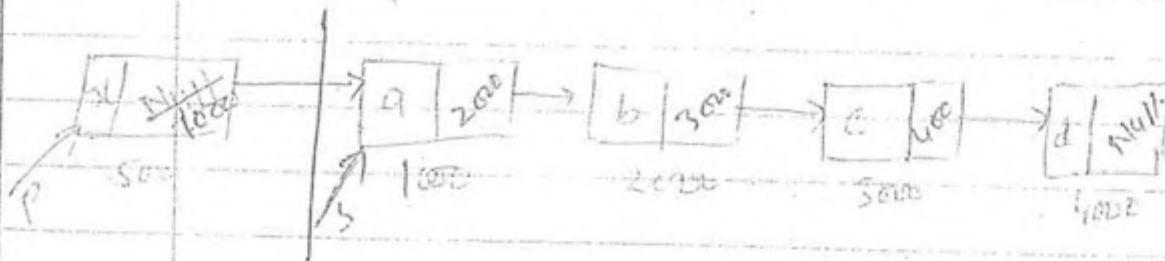
return (s);

3



3

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



Add u before (L (struct node *s, int u))

{

struct node *p;

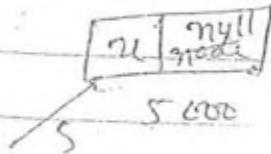
 $p = (\text{struct node}) \text{ malloc}(\text{sizeof}(\text{struct node}))$; $p \rightarrow \text{data} = u$, $p \rightarrow \text{next} = \text{null}$,

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

$s = p;$

$\text{return}(s);$

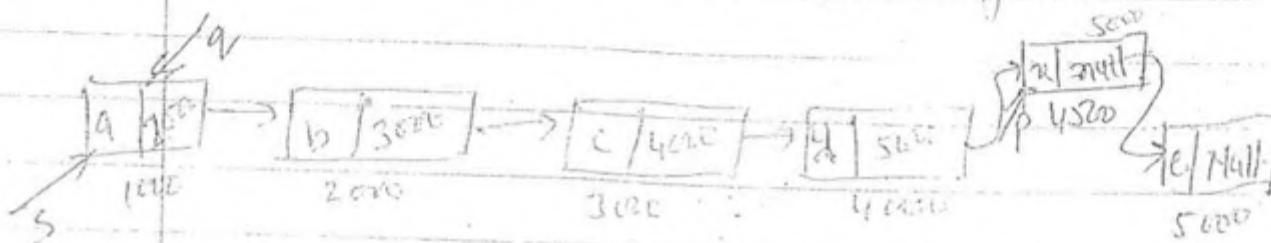
}



#

WACP to insert a node with data 20 after a node with data 10.

/a



Add 20 after 10 (struct node *s, int z1, int y).

struct node *p, *q;

p = (node *)z1;

q = s;

if (s == null)

{

Print ("Nope L:Li is empty");

exit(1);

}

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

{

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

}

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

{

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

}

else

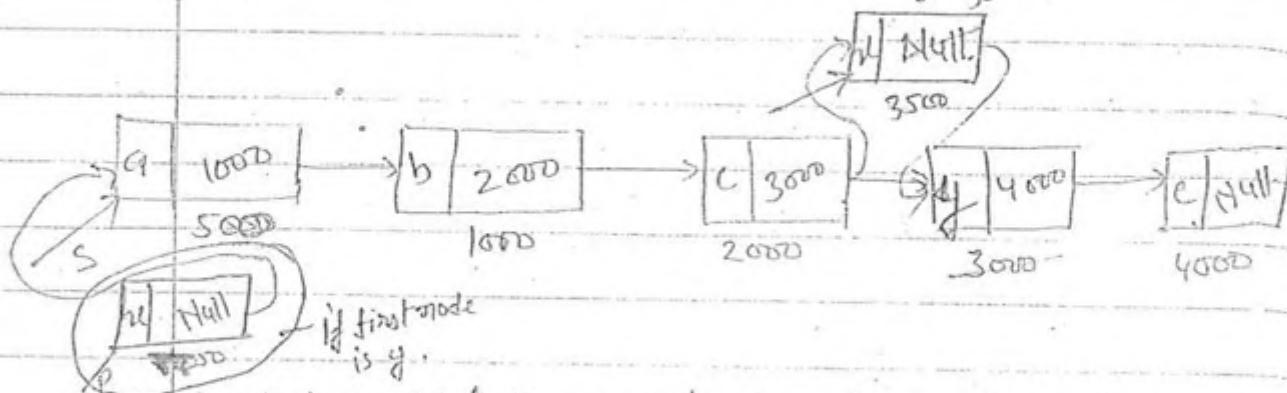
{

 $\text{printf}("y \text{ not present" })$

exit(1);

{ }

WACP to add a node with data 26 before a node with data 4.



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

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

```
p = Get nod(u);
```

```
q = s; r = null;
```

```
if (s == null)
```

```
{
```

```
Print (" LL is empty")
```

```
exit(1);
```

```
}
```

```
else
```

```
{
```

```
if (s->data == y)
```

```
{
```

```
p->next = s;
```

```
s = p;
```

else

{

→

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

{

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

with 2 pointers

3

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

{

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

3

else

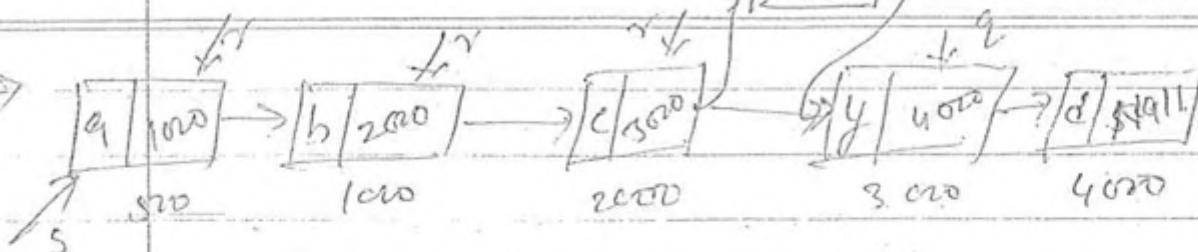
{

printf (" Not possible ");

3

3

3



$q = s;$

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

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

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

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

else

{

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

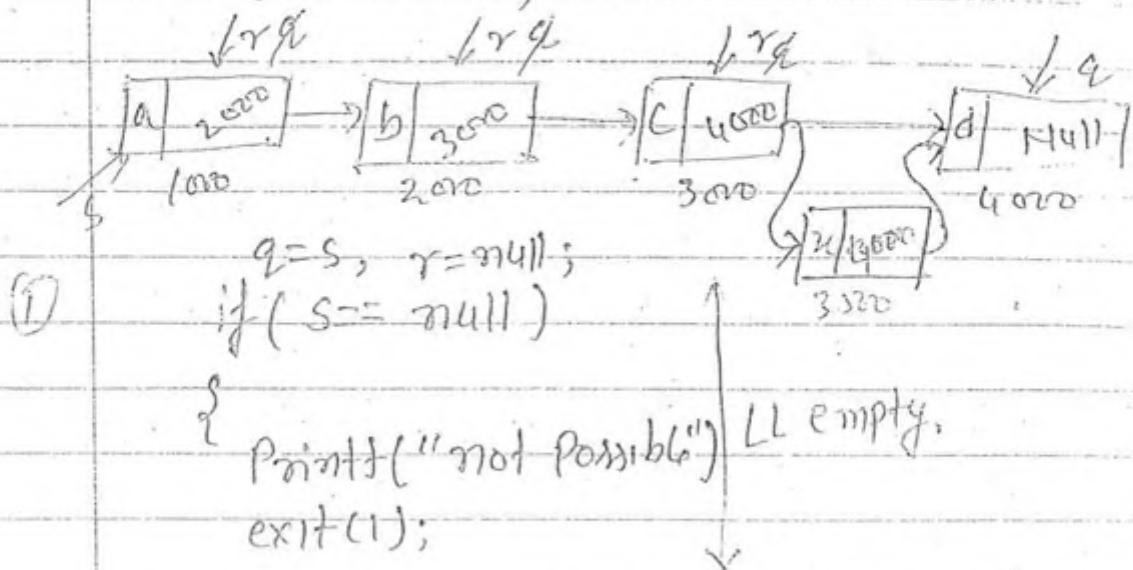
$q \rightarrow \text{next} = p;$

}

singly
pointer

(2)

WACP to add a node with data 26 before
the last node;



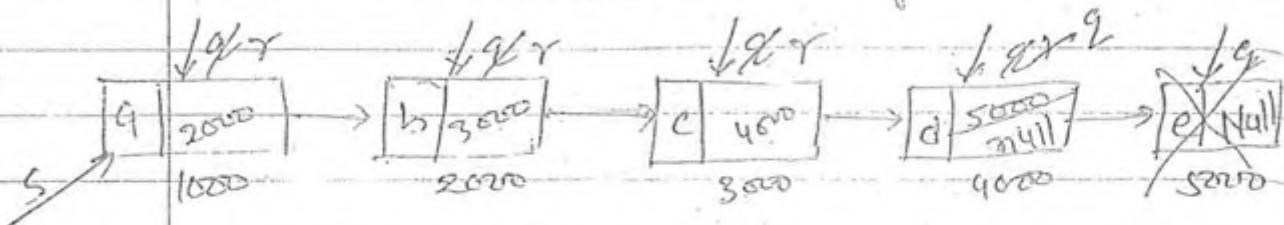
① if ($s == null$) {
 Print("not possible");
 exit(1);
}

3
② if ($s \rightarrow next == null$) {
 $p \rightarrow next = s$;
 $s = p$;
 3
 while ($q \rightarrow next != null$) {
 $r = q$;
 $q = q \rightarrow next$;
 $r \rightarrow next = p$;
 $p \rightarrow next = q$;
 }
}

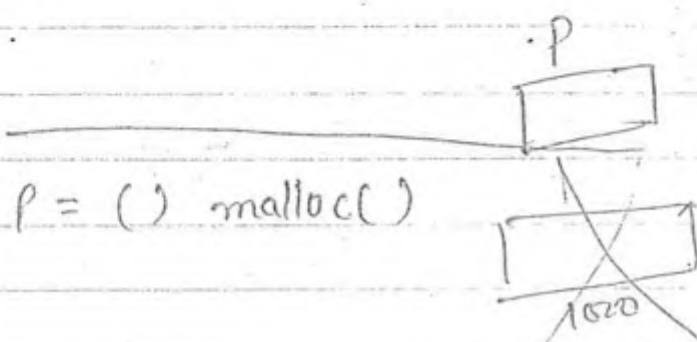
more than 1 node.

(node)

* WACP to delete last element of Linked list.

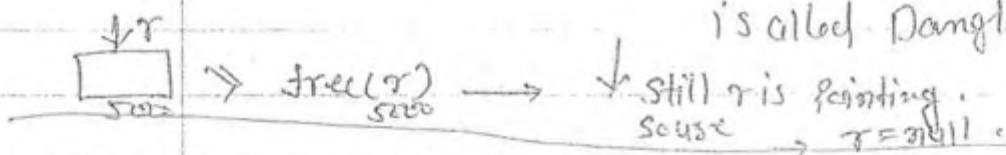


Struct node *p



* no one can point where no memory.

is called Dangling Pointer



Delete node LL(struct node *s) :

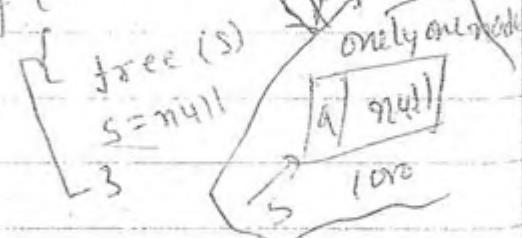
{ if ($s \rightarrow \text{next} = \text{null}$)

struct node *q;

 $q = s$

① empty :

↑-node

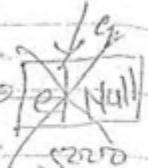
② while($q \rightarrow \text{next} \neq \text{null}$) ↑
 $\left\{ \begin{array}{l} r = q \\ q = q \rightarrow \text{next} \end{array} \right.$

more than

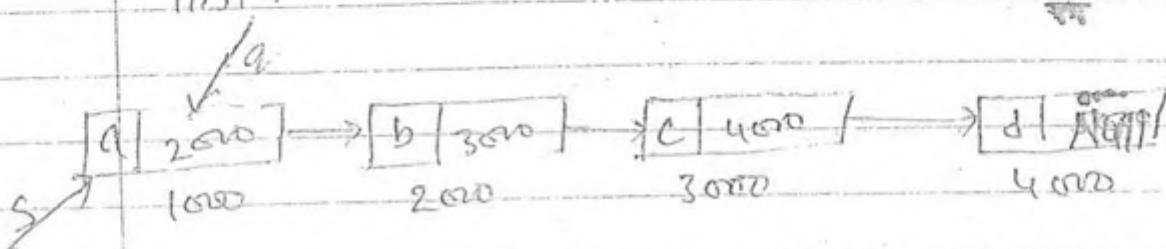
list:

 $x \rightarrow \text{next} = \text{null};$ $\text{free}(q);$ $q = \text{null};$

+ node.

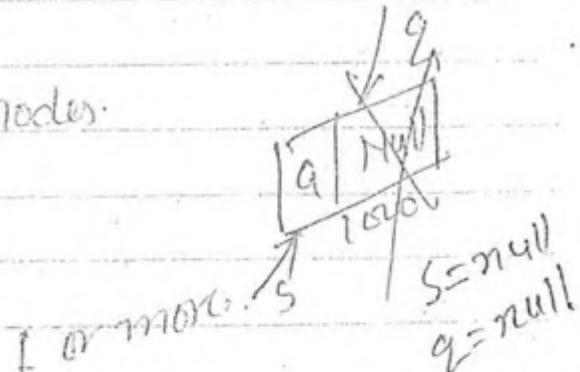


P WAP to delete a node from start of linked list.



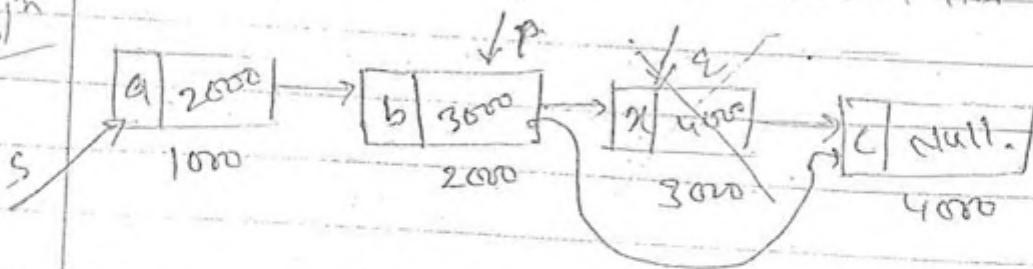
deletestart (struct node * s)

{
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 XL (struct node *s, int x)

{

struct node *p, *q;

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

{

empty

3

else

{

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

{

$p = s$

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

free p

$p = \text{null};$

3

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

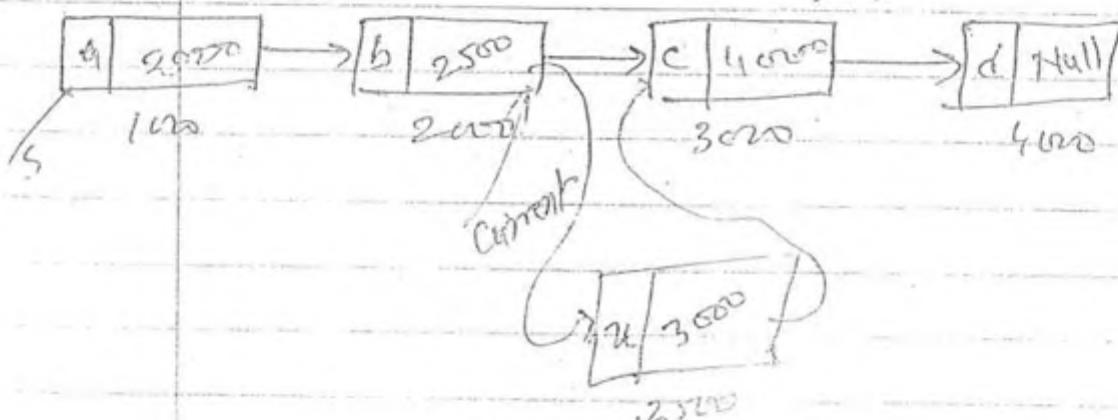
3

~~#~~ Which one of the following statement insert an element x after Position (current).

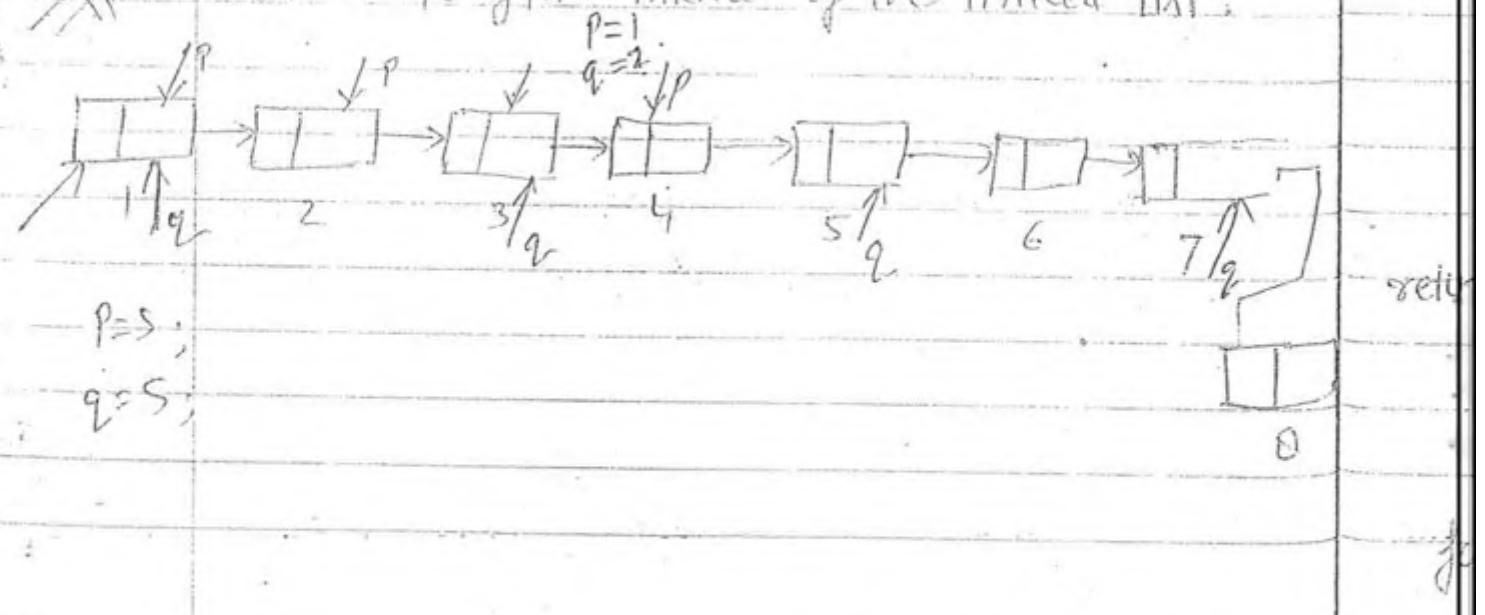
- a) Current = NewNode (x , current)
- b) current = NewNode (x , current \rightarrow next)

c) current \rightarrow Next = NewNode (x , current)

~~d) current \rightarrow next = NewNode (x , current \rightarrow next)~~



~~#~~ WAP to find middle of the linked list.



ment

(प्रत्येक).

$$p = q = 5 ;$$

```
while( q != null & & q->next != null & & q->next->next != null )
```

8

$P = P \rightarrow next$.

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

8

return (P)

Ex Consider the function f defined below:

Struct item

1

int data

struct item *next;

3

int f(struct item *p)

1

```
return ((p==null)||((p->next==null)||((p->data <= p->next->data)
```

88 + (p->next)

-3

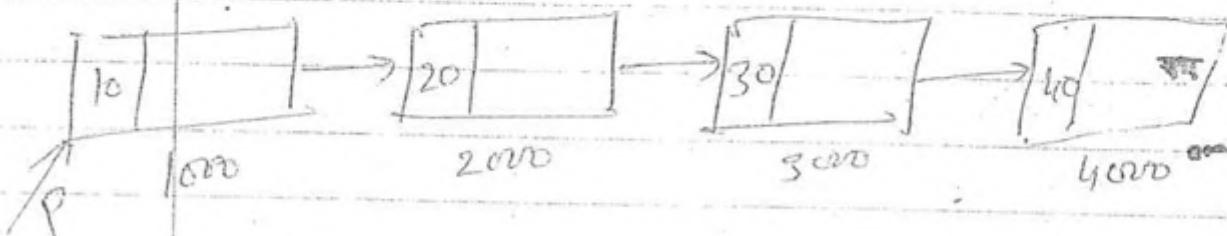
for a given linked list P. The above function

f returns iff.

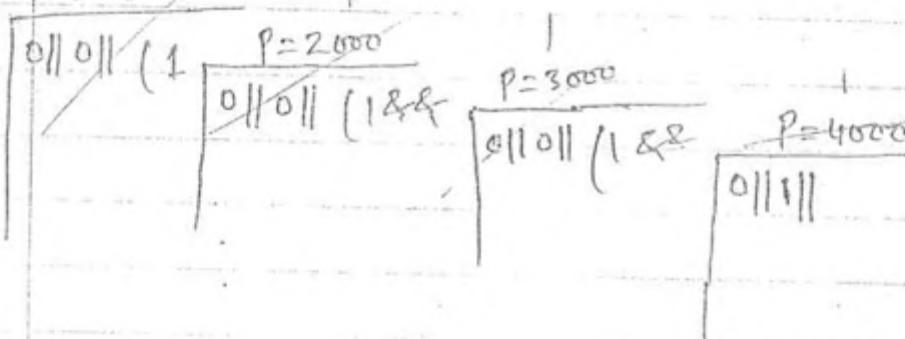
2

10
90
91
92
11

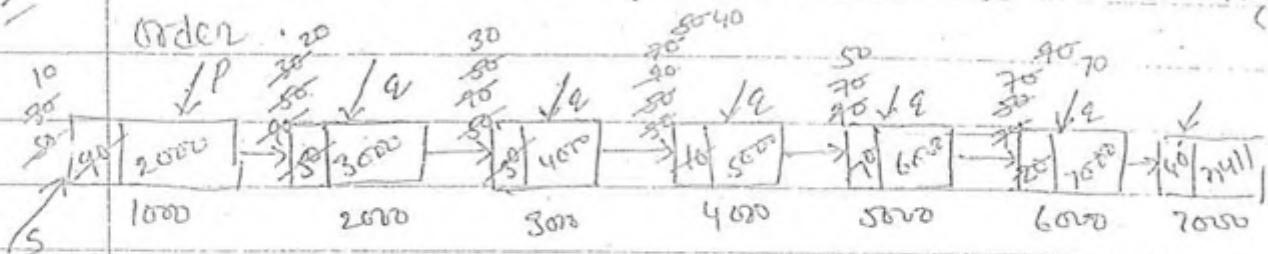
- a) P is empty (or) exactly 1-element.
- b) non-decreasing order.
- c) non-increasing order.
- d) none.



$$I = P = 1000$$



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 ;

}
}

#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

Ans

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

```
Void rearrange(struct node *list)
```

#8

10

S

Sol

```

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;
}
```

Q) P →

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

if (list != null)

1000

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

if (list != null)

11

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

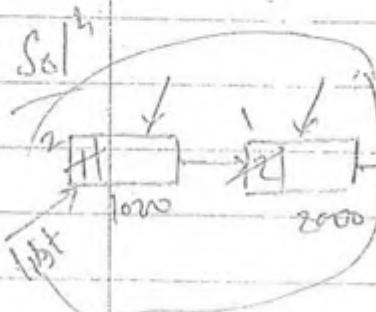
if (list != null)

11

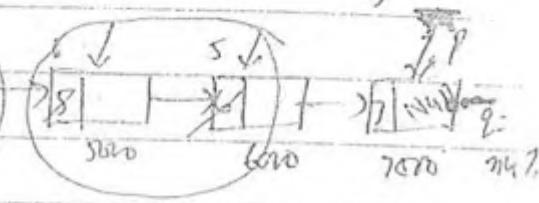
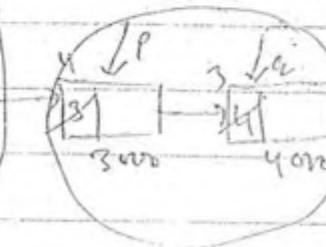
d) 2, 1, 4, 3, 6, 5, 7

if (list != null)

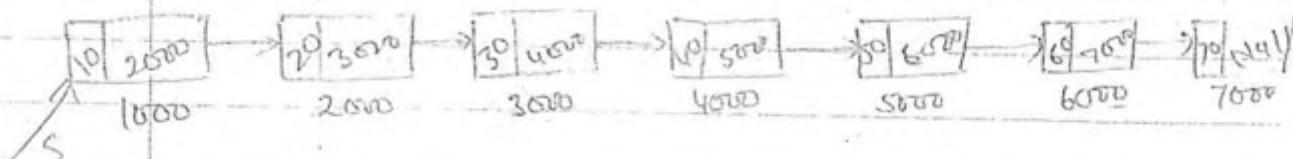
11



```
#define NULL 0;
#define VO 0;
```



#^Q C/C++ 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.

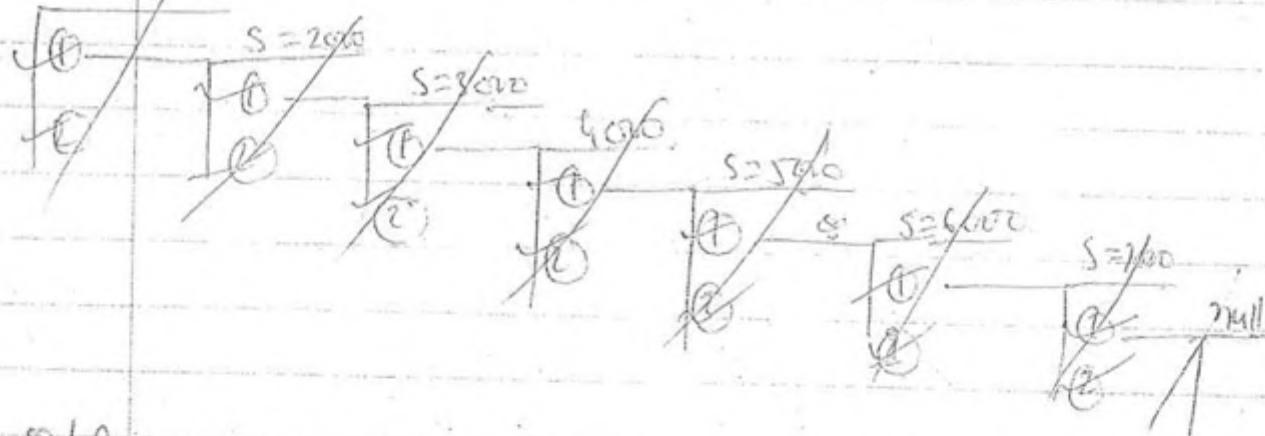
- Recursion

Print LL reverse (struct node *s)

```

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

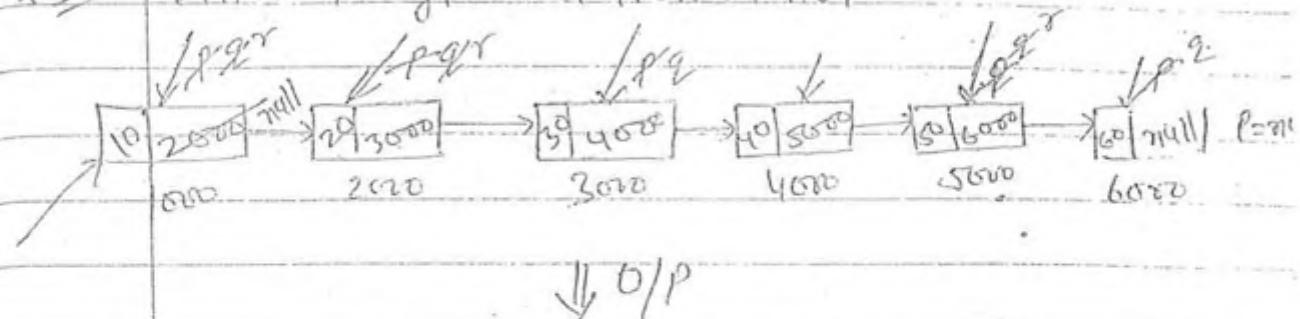
$s = 1000$



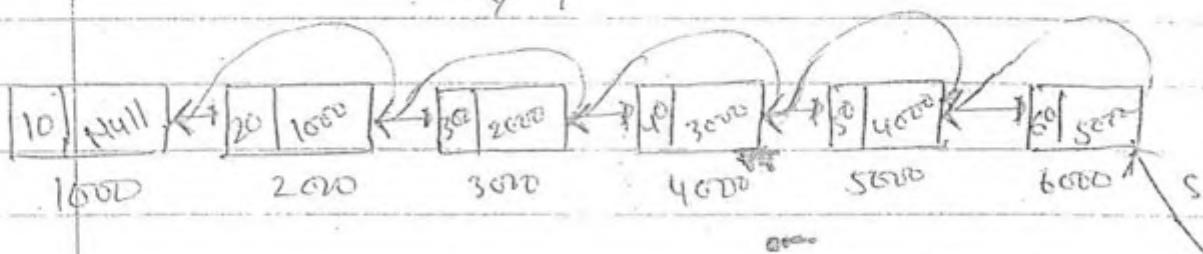
O/P : - 100, 60, 50, 40, 30, 20, 10

#8

WACP to give a linked list.



↓ O/P



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.t \Rightarrow $3 \times 2^6 = 6$
 $= O(1)$

 {
 r=q
 q=p
 p=p->next
 q->next=r
 }
 S=q

Note:-

DELTA	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 returning modified list. Some parts of the function left blank (because some 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)

```

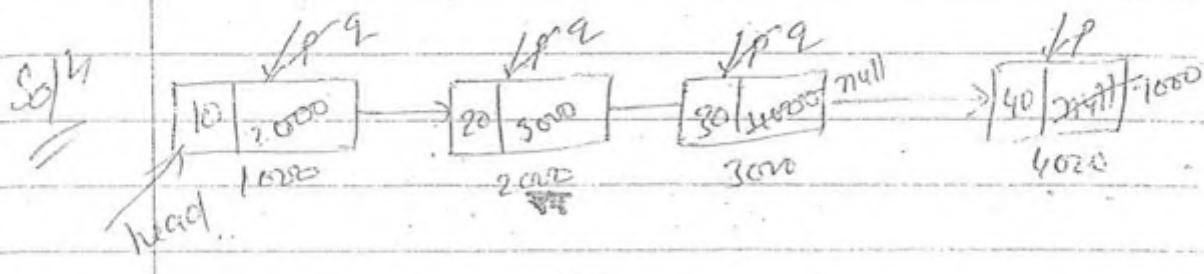
{ q = p; p = p->next;
=3 }
```

Solving.

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

Ques 1st

by



Q/p -

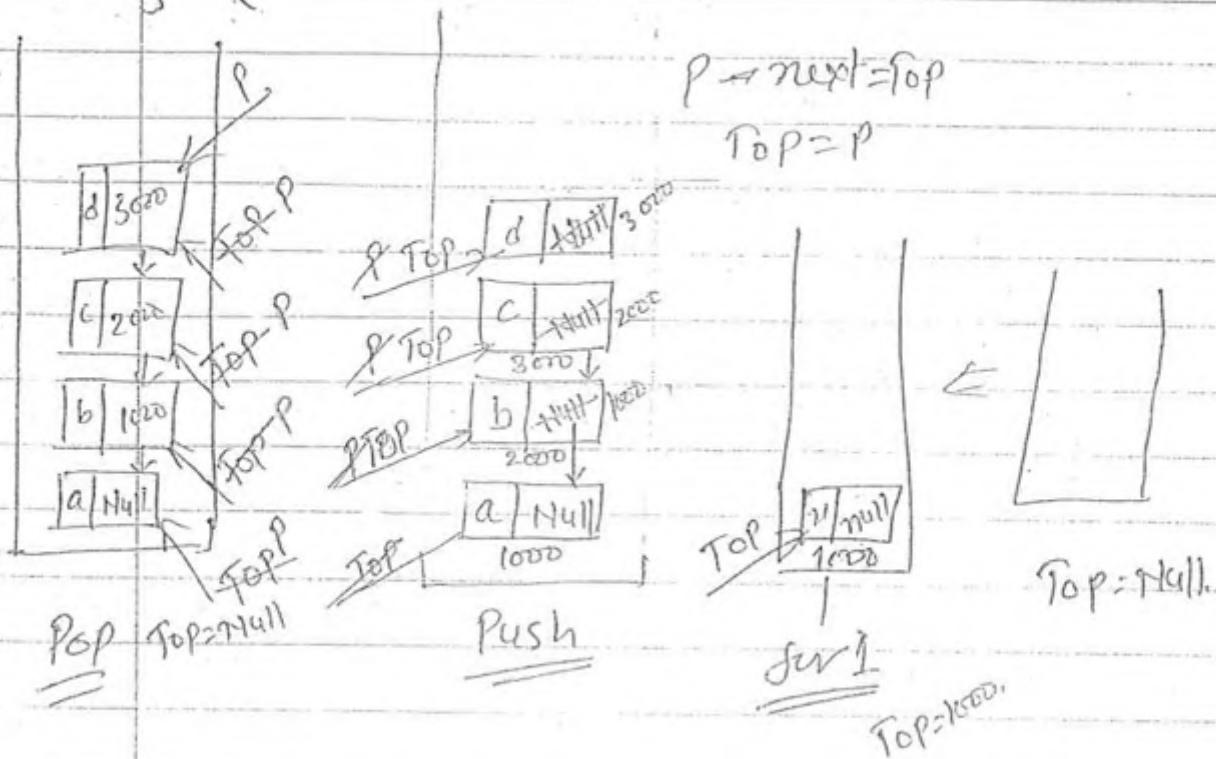
- a) $q = \text{null}$, $p \Rightarrow \text{next} = \text{head}$, $\text{head} = p$;
- b) $q \Rightarrow \text{next} = \text{null}$; $\text{head} = q$; $p \Rightarrow \text{next} = \text{head}$, $\text{head} = p$;
- c) $\text{head} = p$; $p \Rightarrow \text{next} = q$; $q \Rightarrow \text{next} =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

null))

if ?

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

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



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 << "stack is underflow";
```

```
            exit(1);
```

```
}
```

```
    if (y = Top->data);
```

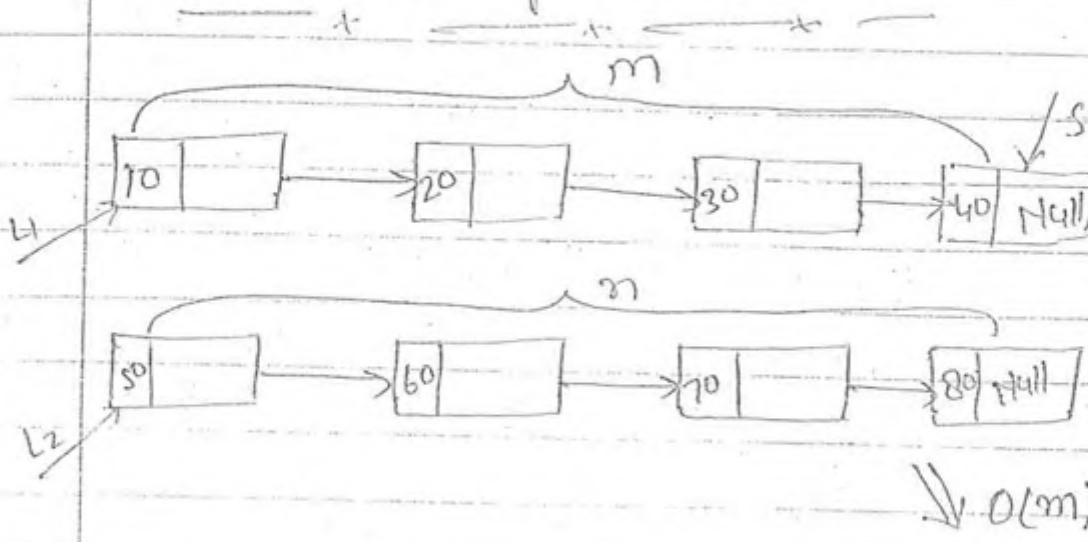
```
        P = top; Top = Top->next;
```

```
        free(p); p = NULL;
```

```
    return(y);
```

```
}
```

Concatenation of 2 Linked List.



if both L1 and L2
aren't null

$\Rightarrow O(n)$

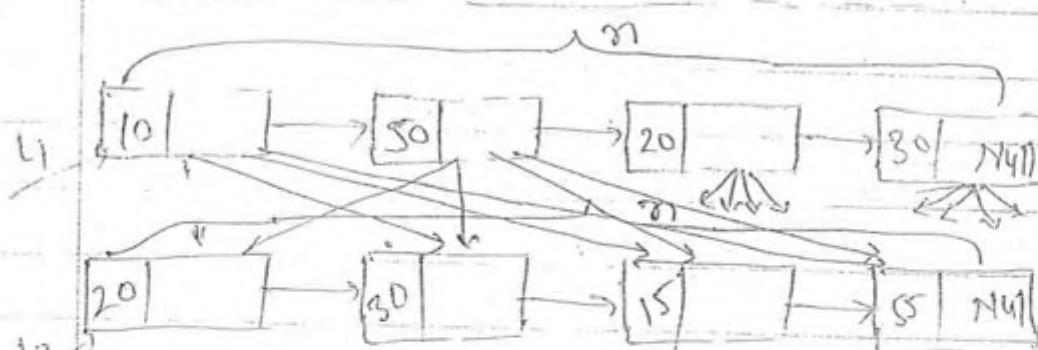
$S = L_1;$
 $while (S \rightarrow next \neq \text{null}) \{$
 $S = S \rightarrow next;$
 $S \rightarrow next = L_2;$
 $\text{return}(L_1)$

(1)

(2)

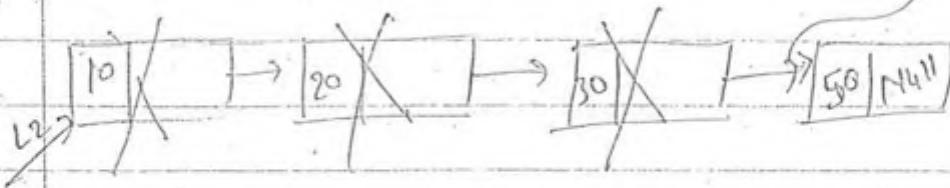
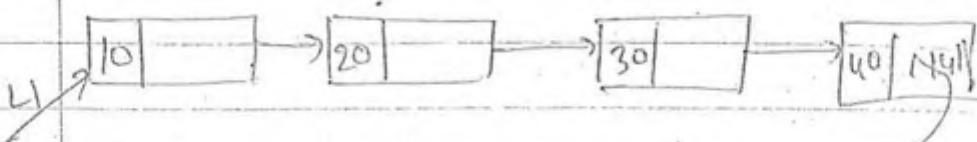
(3)

Intersection of 2 Linked List.



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

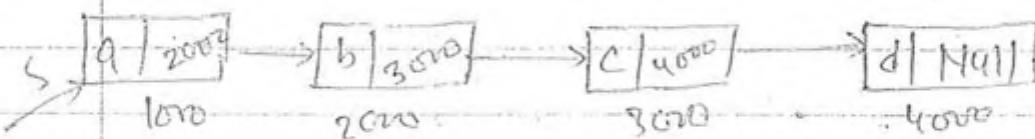
(1)

Union of 2 LL'sAlgo

resulted 3rd linked list

- (1) Add 1st LL to the another ans then ans.
- (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)

- total) ① Random access is not possible.

10	20	30	40	50	60	70
1	2	3	4	5	6	7

Dis Advantage

Delete $\Rightarrow O(n)$

Insertion $\Rightarrow O(n)$

go place $\Rightarrow O(1)$

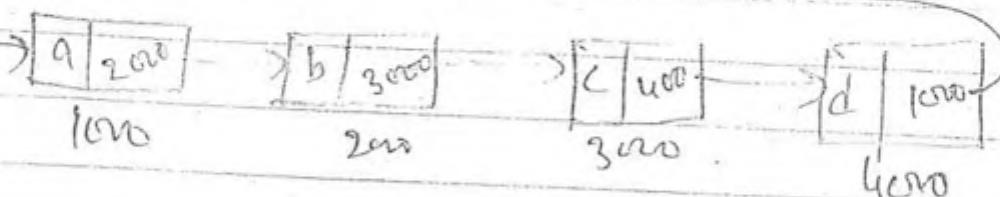
Advantage

Random access.

- (2) From every node we always move only one direction. ie 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.



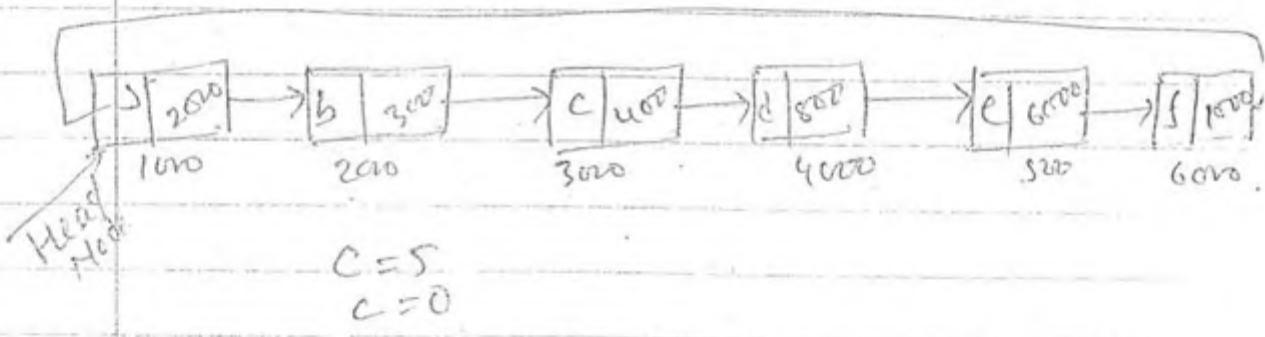
Using CSLL from a given node we can go back also.

Drawback

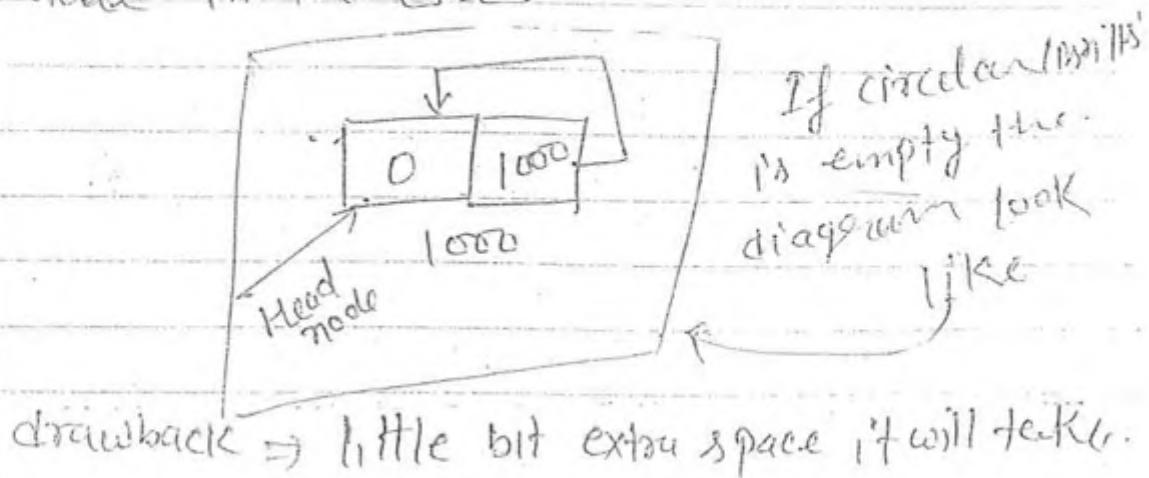
The draw back of CSLL is it will go to infinite loop if we don't write the code properly.

Head Node

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

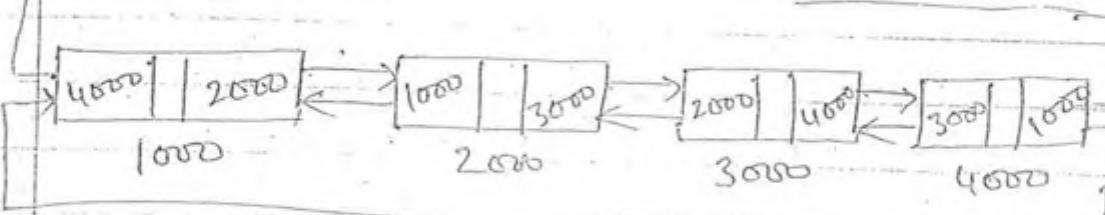


Using Head node data part will contain valuable information like no. of node in the CSLL.



Double linked list

(Double Circular linked list (DCLL))

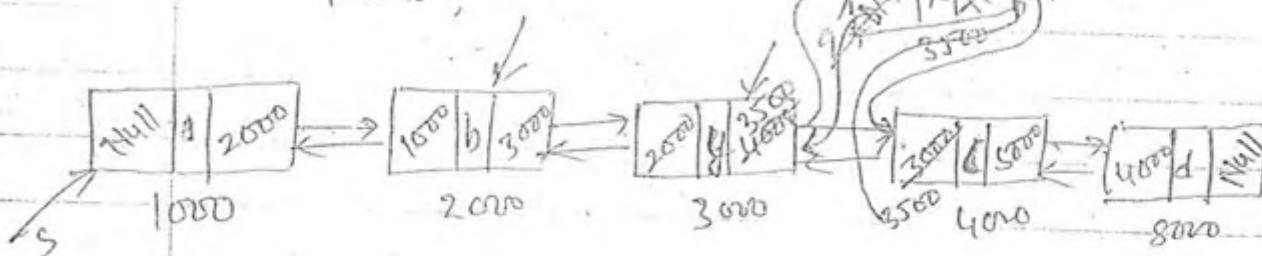


The advantage of DLL is if we lost one pointer we will get it back with the help of another pointer.

P

WACP 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->r(p) != null)

P = P->r(p);

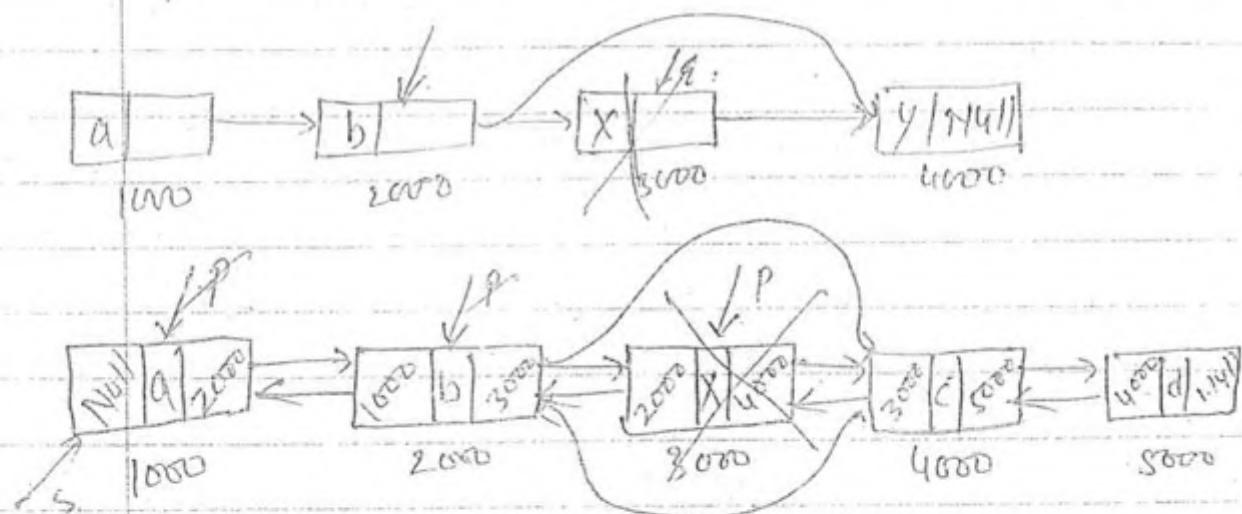
if (P->data == y)

{

 $q = \text{GetNode}(x)$ $q \rightarrow rp = p \rightarrow rp$ $p \rightarrow rp = q$ $q \rightarrow rp \rightarrow lp = q$ $q \rightarrow lp = p;$

}

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

 $P = S;$ While ($P \rightarrow \text{data} != n \& \& P \rightarrow rp != null$) $P = P \rightarrow rp.$

Polynomial Addition (Using Single LL)

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

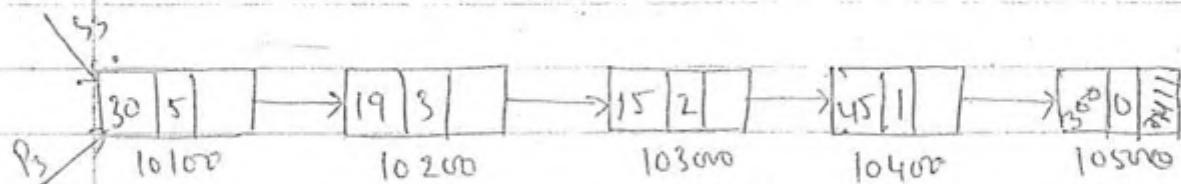
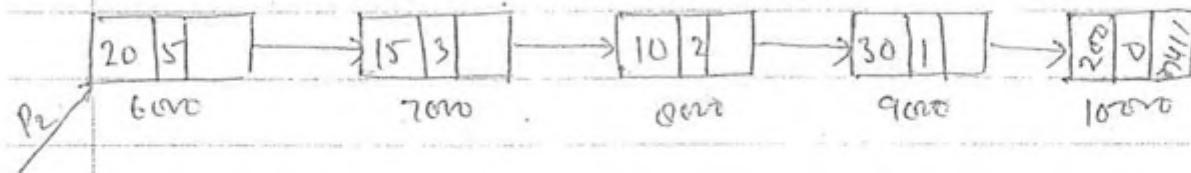
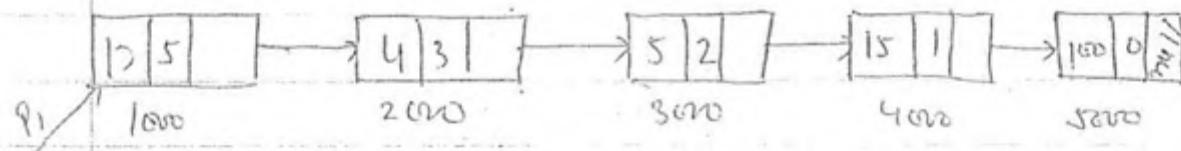
$$P_2 = 20x^5 + 15x^3 + 10x^2 + 30x + 200$$

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

Struct Poly

```

{
    int coe;
    int pow;
    struct Poly *next;
}
```



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

{ S₃ = p₃; }
 if m > n
 if m = n

while ((p₁ != null) && (p₂ != null)); o(n)

{
 if (p₁ → Pow == p₂ → Pow)

{
 p₃ → Pow = p₁ → Pow + p₂ → Pow;

 p₁ = p₁ → next , p₂ = p₂ → next ;

 p₃ = p₃ → next ;

}

else

{
 if (p₁ → Pow > p₂ → Pow),

{
 p₃ → Pow = p₁ → Pow;

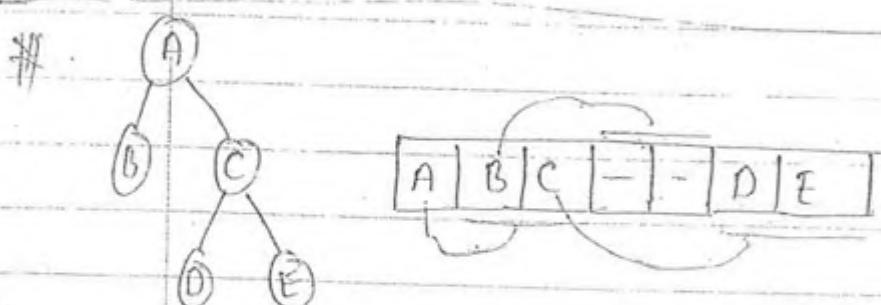
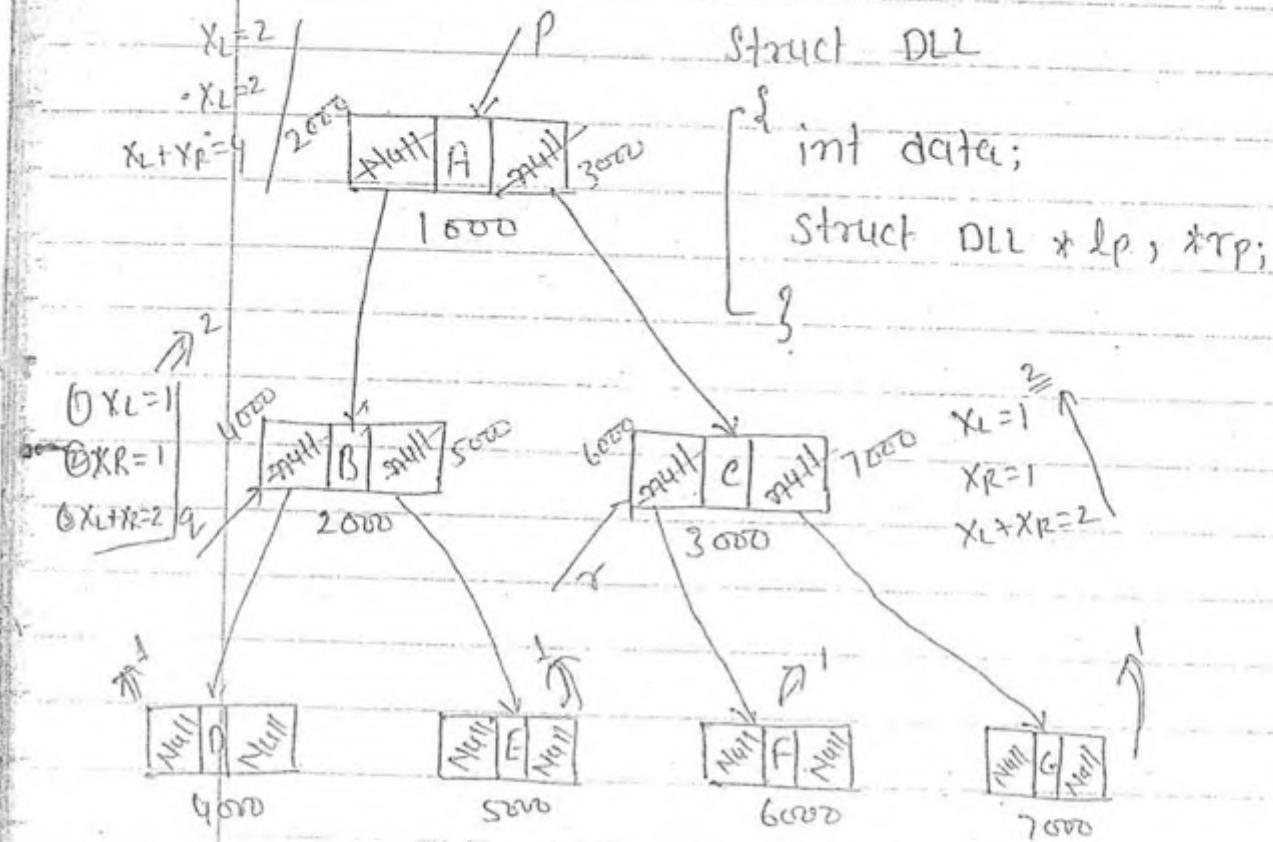
 p₃ → Coe = p₁ → Coe;

 p₁ = p₁ → next ;

 p₃ = p₃ → next ;

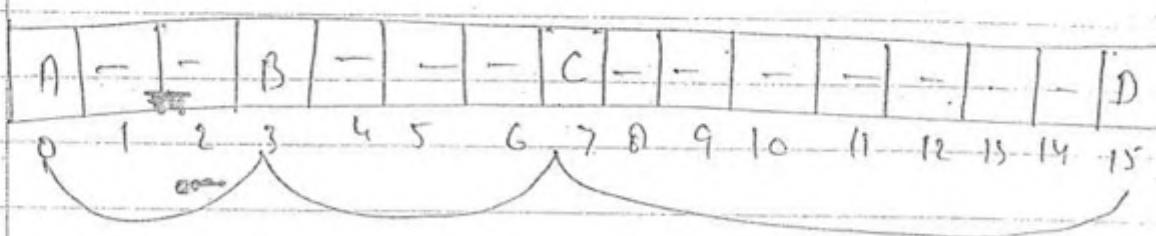
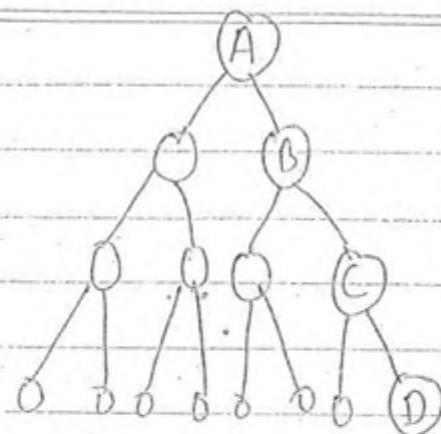
}

Binary Tree



linked list is better approach for Binary tree not a array

#



R

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

{

$$X_L = \text{Num leaf node } (P \rightarrow l_p)$$

$$X_R = \text{Num leaf node } (P \rightarrow r_p)$$

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

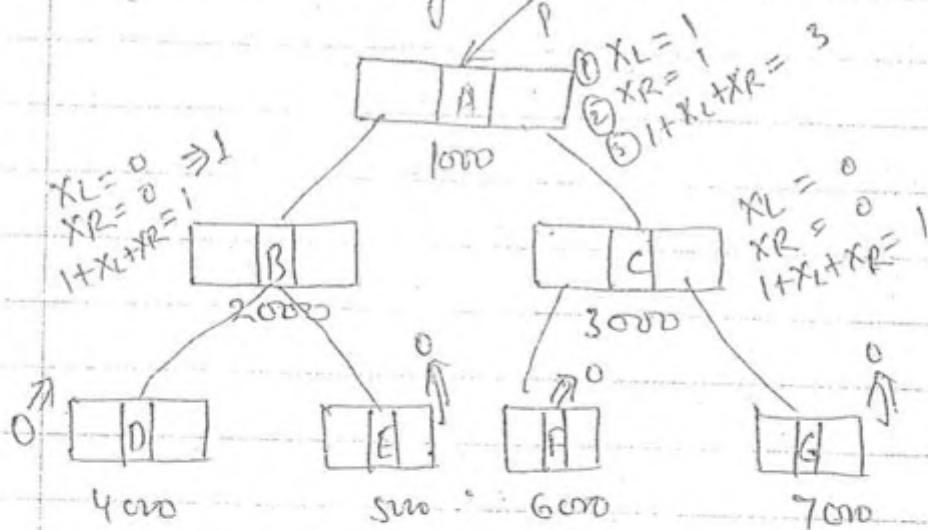
{

}

{

P

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



leaf node means internal node is zero.

Numinternalnodes (struct BtNode *P)

{

if (P == Null)

return (0);

else

{

if (P->lp == Null & & P->rp == Null)

return (0);

else

{

$X_L = \text{Numinternalnodes}(P \rightarrow lp)$

$X_R = \text{Numinternalnodes}(P \rightarrow rp)$

return (1 + $X_L + X_R$)

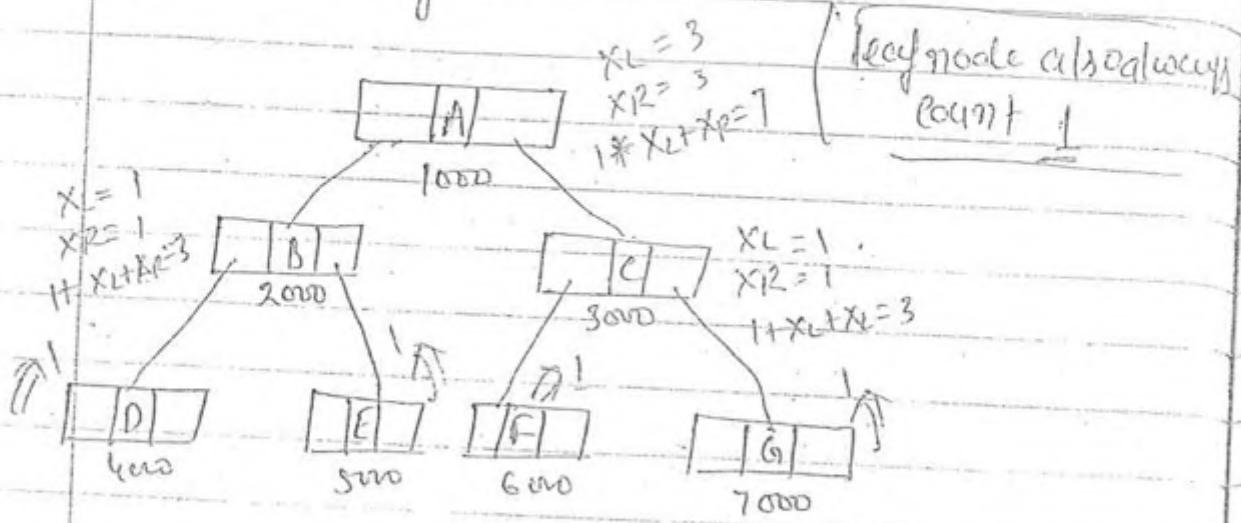
3

3

3

2

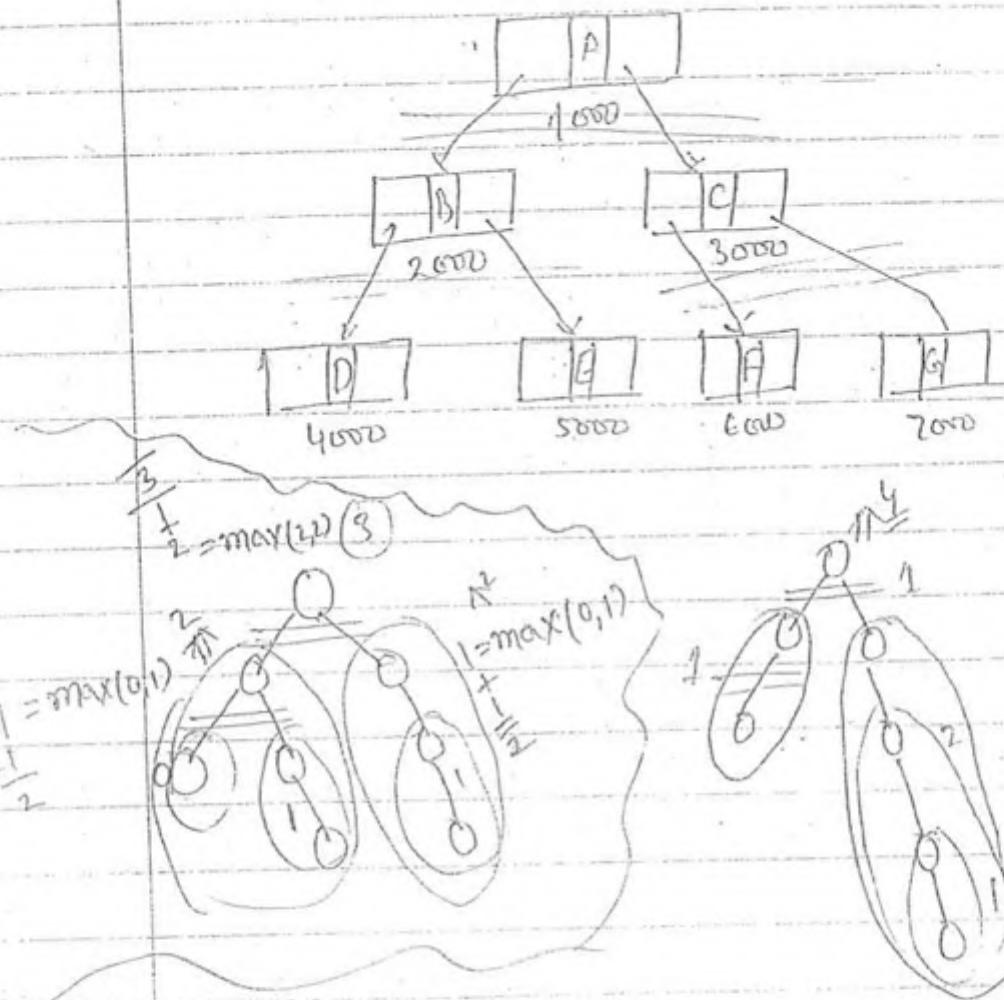
WACP to count total no of nodes in the given binary tree.



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 )
        }
    }
}
```

2 WAP to find height of a given binary tree.



Height of BT (struct BtNode *p)

```

if ( p == NULL )
    return 0;
else
    if ( p->lptr == NULL & p->rptr == NULL )
        return 0;
    else
        {
            if ( p->lptr != NULL & p->rptr != NULL )
                return ( max( Height(p->lptr), Height(p->rptr) ) + 1 );
            else
                return ( Height(p->lptr) + 1 );
        }
    }
  
```

else

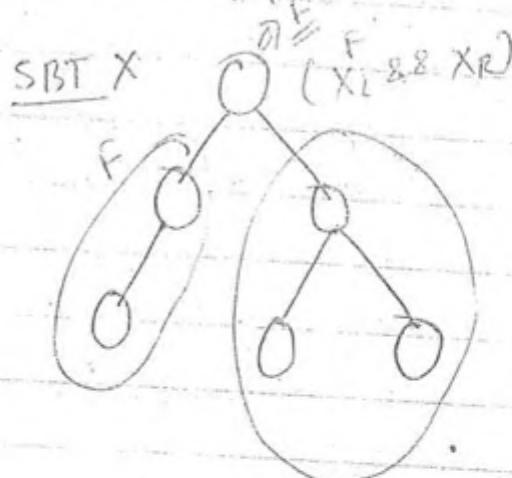
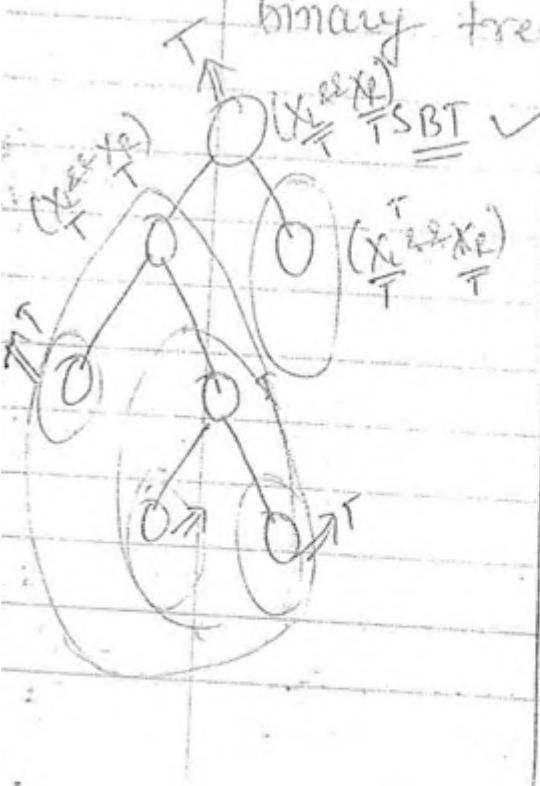
{

 $x_L = \text{Height of BT } (P \rightarrow l_p)$ $x_R = \text{Height of BT } (P \rightarrow r_p)$ if ($x_L < x_R$) return ($x_R + 1$)

else

return ($x_L + 1$){
}{
}{
}

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);
}
  
```

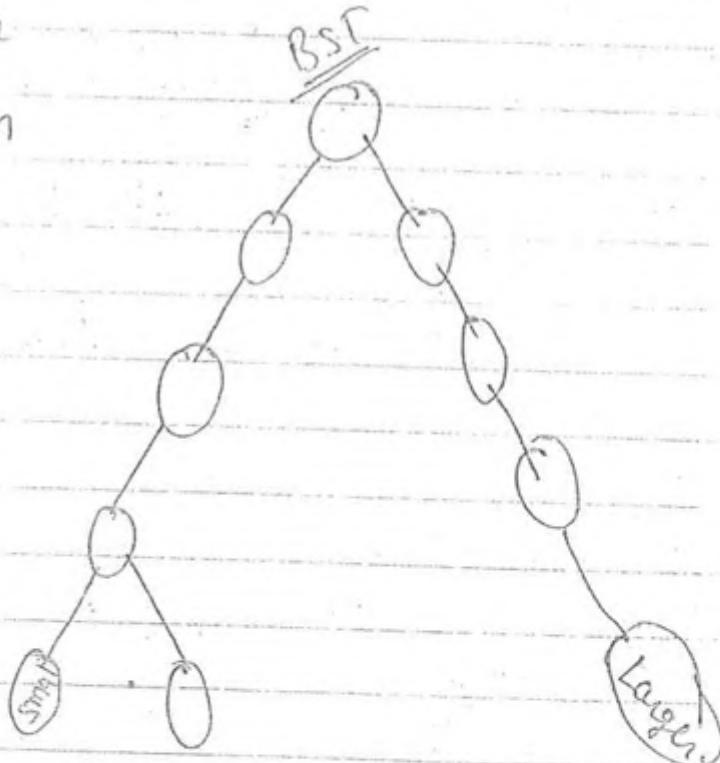
Sol.

B.S.T.



left most value

→ minimum

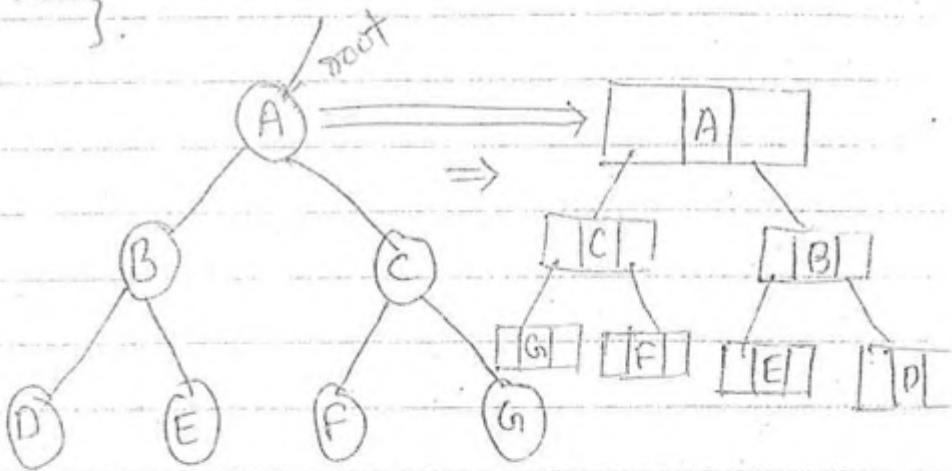
 $T.C \geq O(n)$ w.l.o.g worst case = $O(n)$ 

~~had it~~ ~~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

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.

- X_L = 0
- X_R = 3
- 1 + X_L + X_R

Struct Test

```

struct test {
    int info, mark;
    struct test *p, *q;
};

```

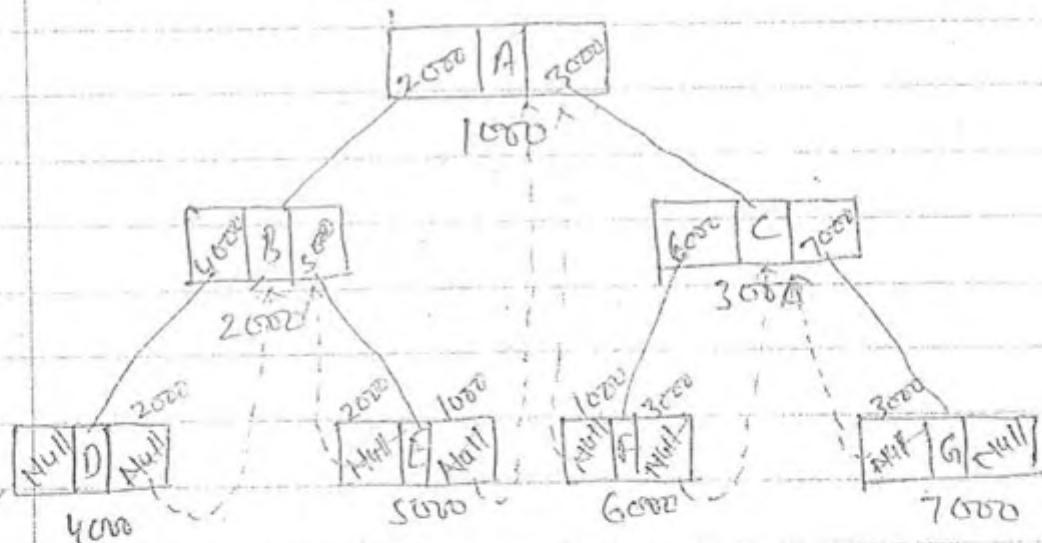
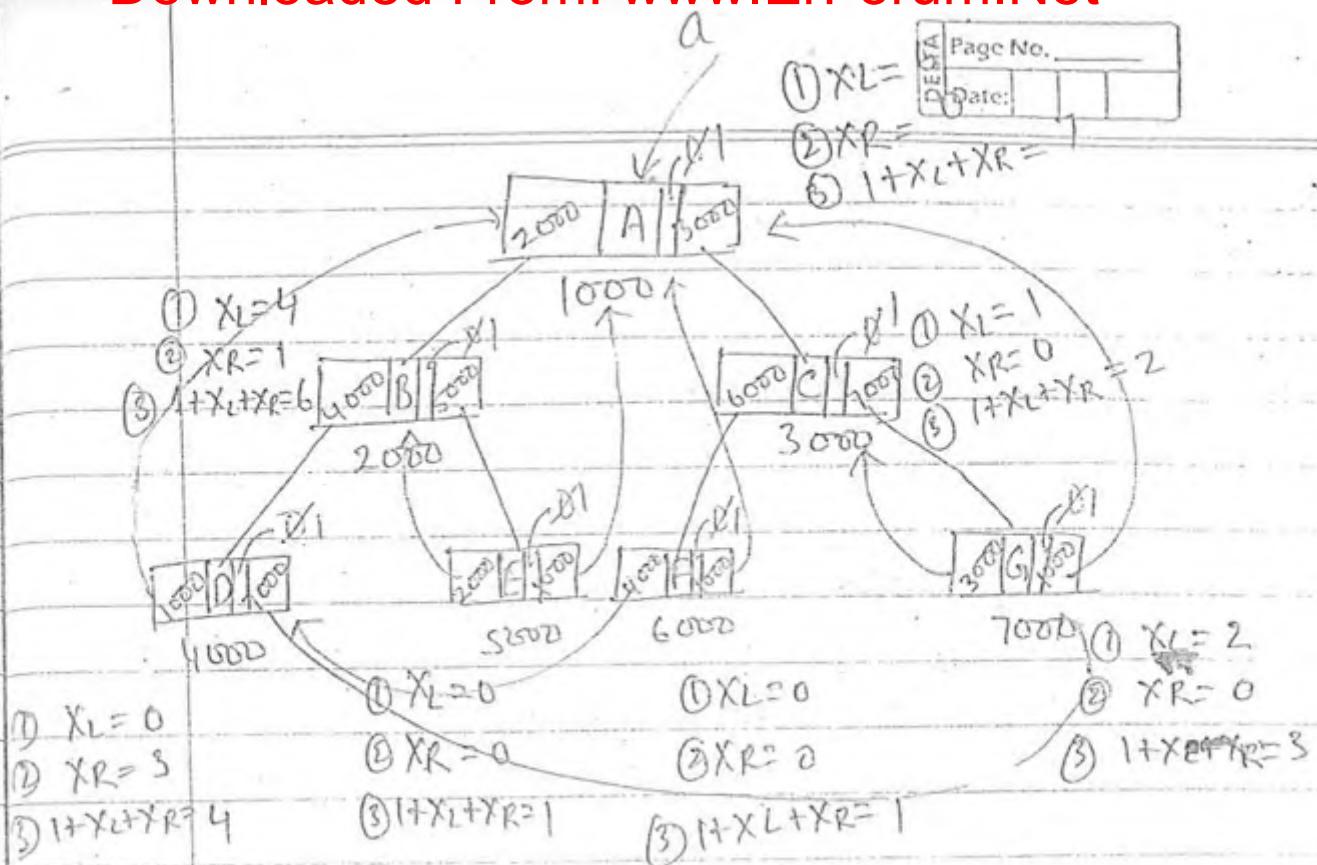
```
int nodeCount ( struct test *a )
```

```
{
    if ( a->mark ) return(0);
}
```

```
else
```

```
{
    a->mark = 1
    return( nodeCount ( a->p ) + node
```

```
count ( a->q ) + 1 );
}
```



Drawback of leaf nodes left pointer and right pointer always null.

We cannot use that space efficiently.

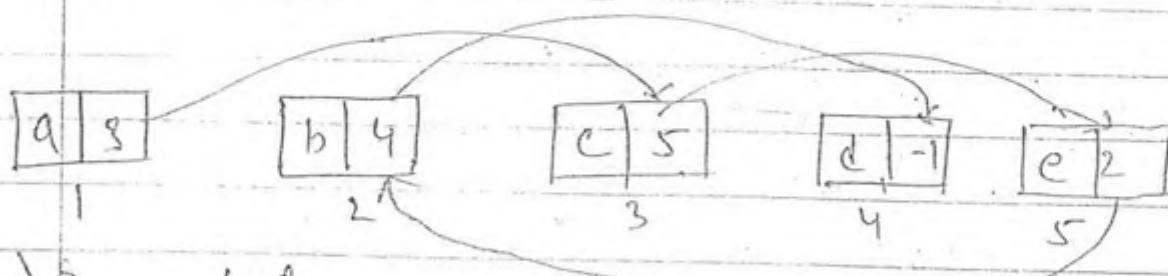
Inorder: - D B E A F C G

Inorder Threaded Binary tree.

Inorder Success \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



Starting node address 1
 temp $\leftarrow 3 \rightarrow 4 \rightarrow -1$

#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:- aceba

```
{ { printf("%d", data[temp])
temp = link[temp]
```

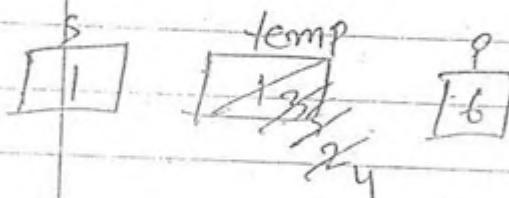
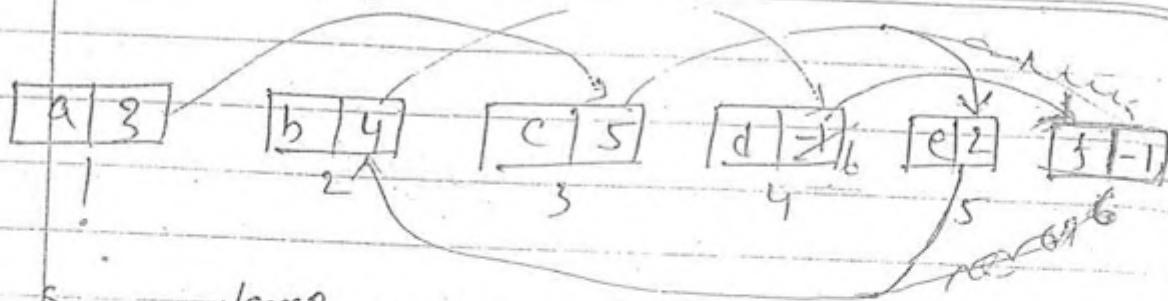
```
}
```

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]

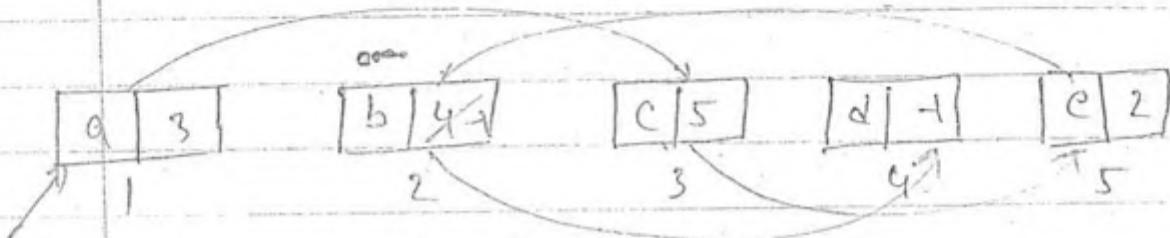
Link[temp] = p;

```
}
```

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) [empty list] 0-element
    }
    
```

```

    if (Link[t1] == -1) [1 element]
        s = -1
    }
    
```

while (Link[t1] != -1)

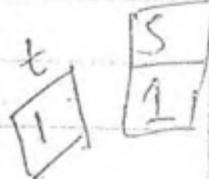
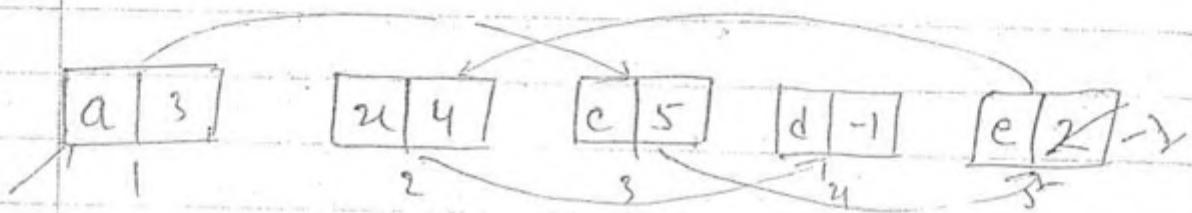
more than
one element

```

    {
        t2 = t1;
        t1 = link(t1);
        link[t2] = -1;
    }
    
```

WACP to delete a node which contain data as 'u' in the given linked list in the array representation.

data		link	
1	a	1	3
2	u	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] == u)
    {
    }
}
  
```

```

1-e
    {
        t = s;
        s = link[s];
        link[t] = -1;
    }
  
```

in the

else

```
{ while (link[t1] != u & link[t1] != -1)
```

```
{ t2 = t1
```

```
t1 = link(t1);
```

}

if

```
(data[t1] == u)
```

or

```
{ link[t2] = -1 link[t1]
```

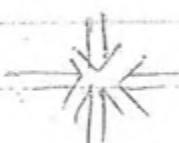
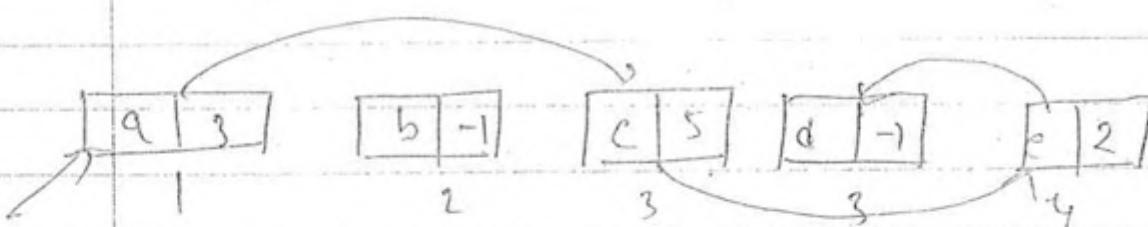
```
link[t1] = -1
```

}

else

u is not there.

3
3

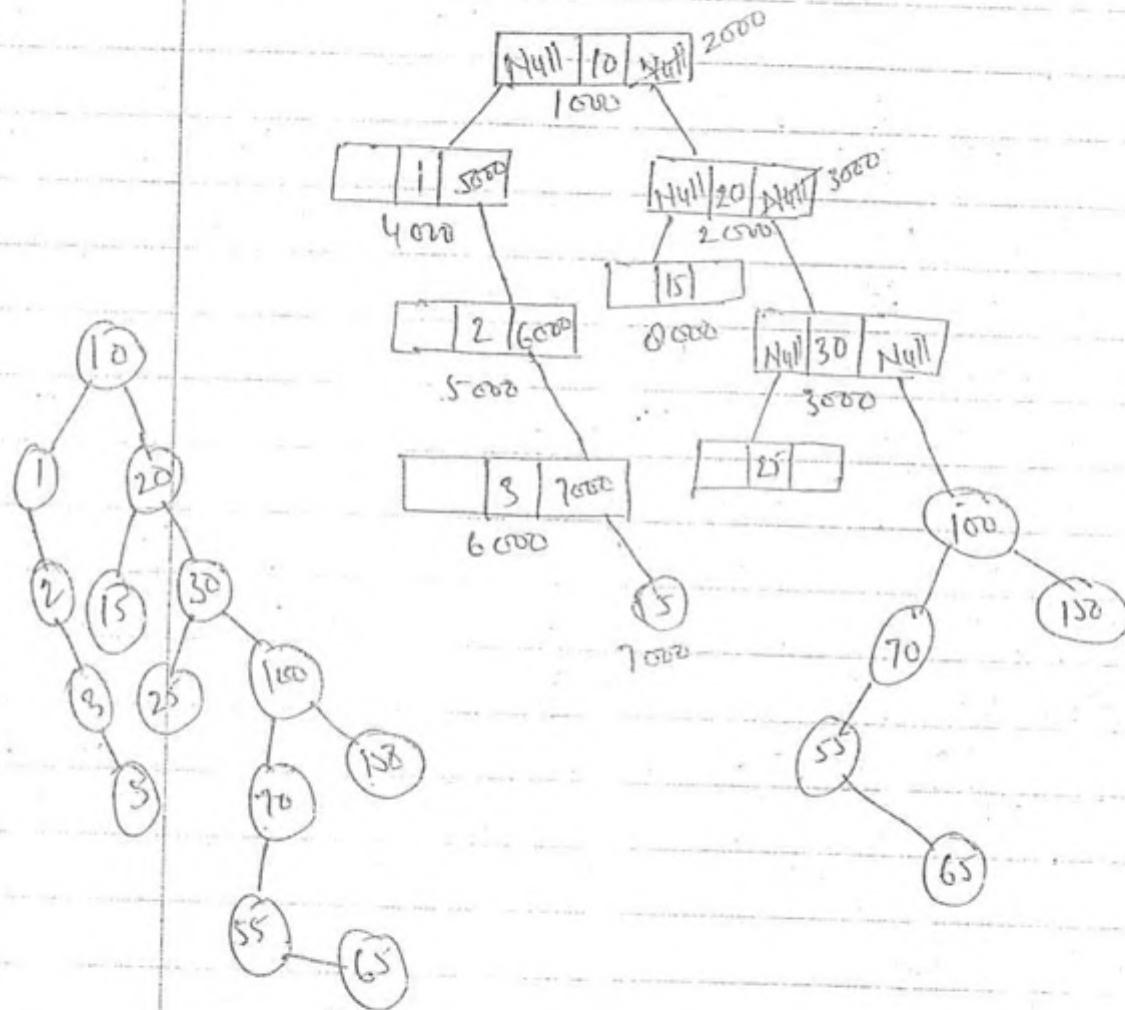


Binary Search Tree

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.

$$\Rightarrow O(1)$$

$$\overbrace{O(n), \mathcal{O}(1) \text{ or } O(\log n)}$$

Avg

Delete

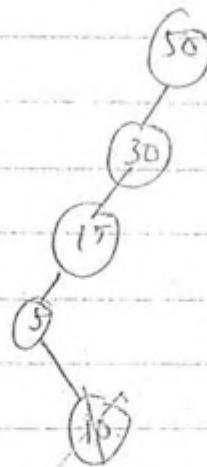
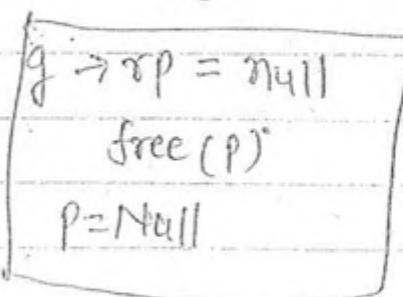
① 0 - children.

② 1 - children.

③ 2 - children.

$$O(1)$$

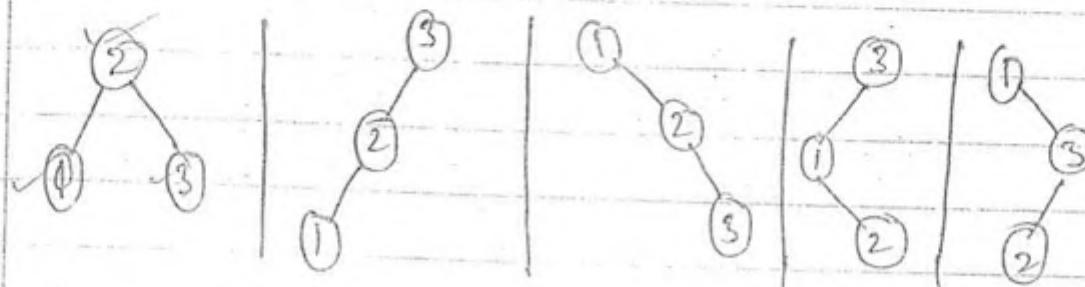
①



AVL Trees (Adelson, Velski , Landis)

- ① Binary Search Tree }
 ② Balanced property. } Maximum $\log_2 n$

Ex $n = 3 (1, 2, 3)$.



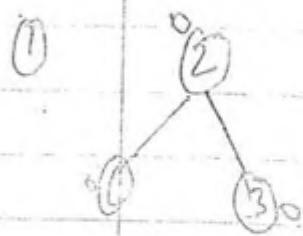
Balance factor :-
$$\left| H(LST) - H(RST) \right|$$

BST



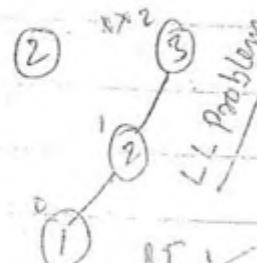
at every node

balance factor ($-1 \leq 0 \leq 1$)

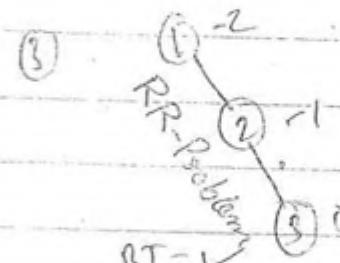


BST ✓
 $O(\log n)$

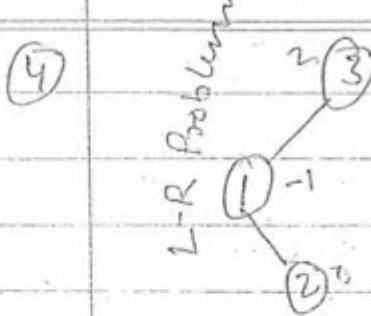
AVL ✓



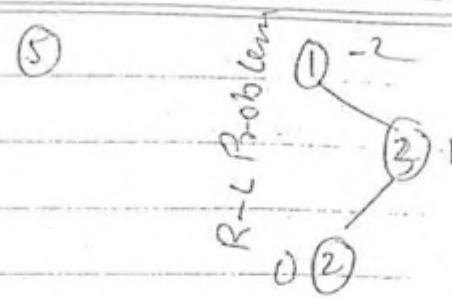
BST ✓
 $O(n)$



BST ✓
 $O(n)$

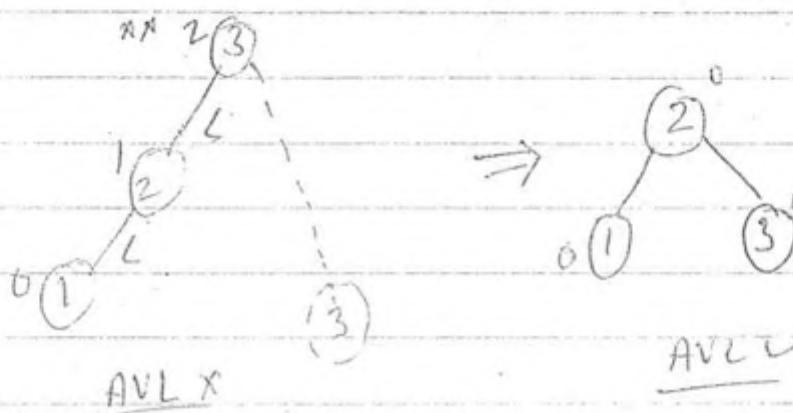


BT ✓
BST ✓
AVL X $O(n)$



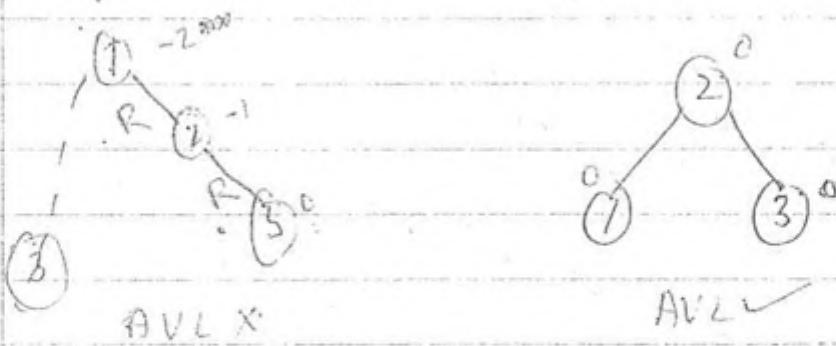
BT ✓
BST ✓
 $O(n)$ AVL X

(1) LL Problem (rotate Top half part right) forcefully.



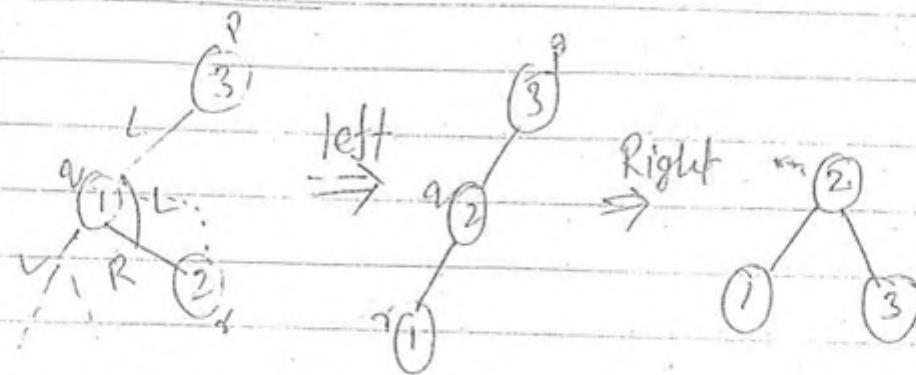
Inorder = 1, 2, 3 Inorder = 1, 2, 3.

(2) RR Problem (rotate Top half part left) forcefully.



Inorder = 1, 2, 3. Inorder = 1, 2, 3.

(3) LR Problem



$p \rightarrow \text{lp} = \gamma$

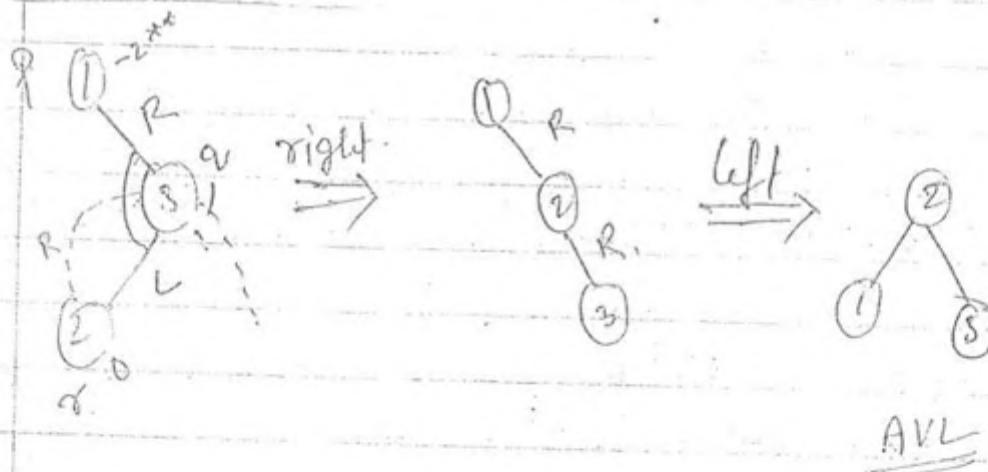
$r \rightarrow \text{lp} = q$

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

$r \rightarrow \text{rp} = \text{null}$

AVL

(4) RL Problem



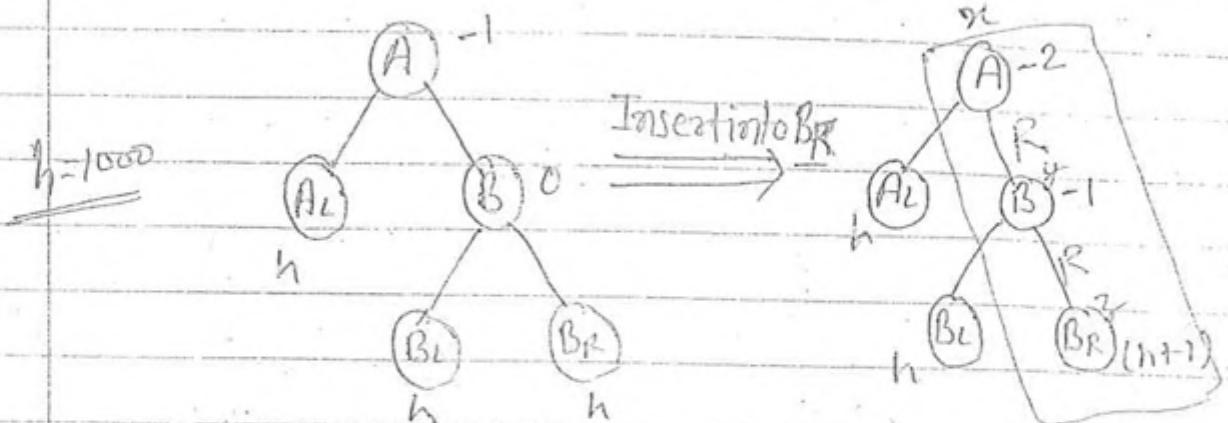
$p \rightarrow \text{rp} = \gamma$

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

AVLAll of the above four problem takes $O(1)$

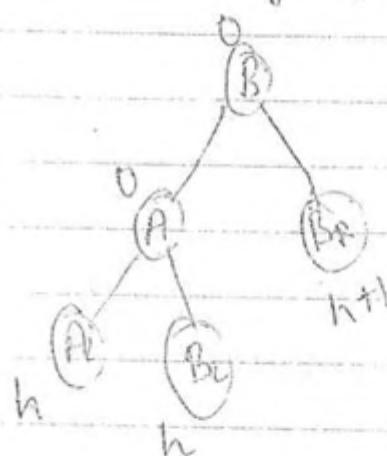
Insulation

R R- Problem



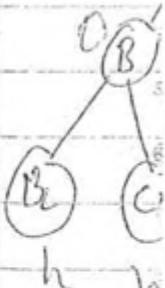
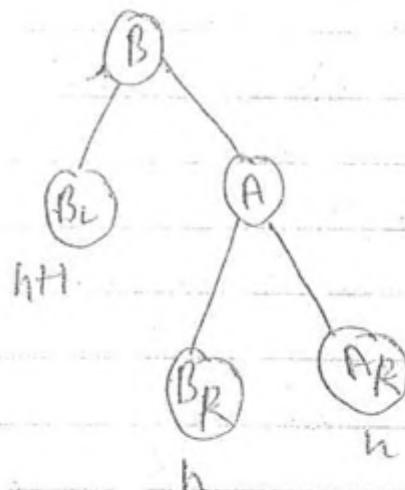
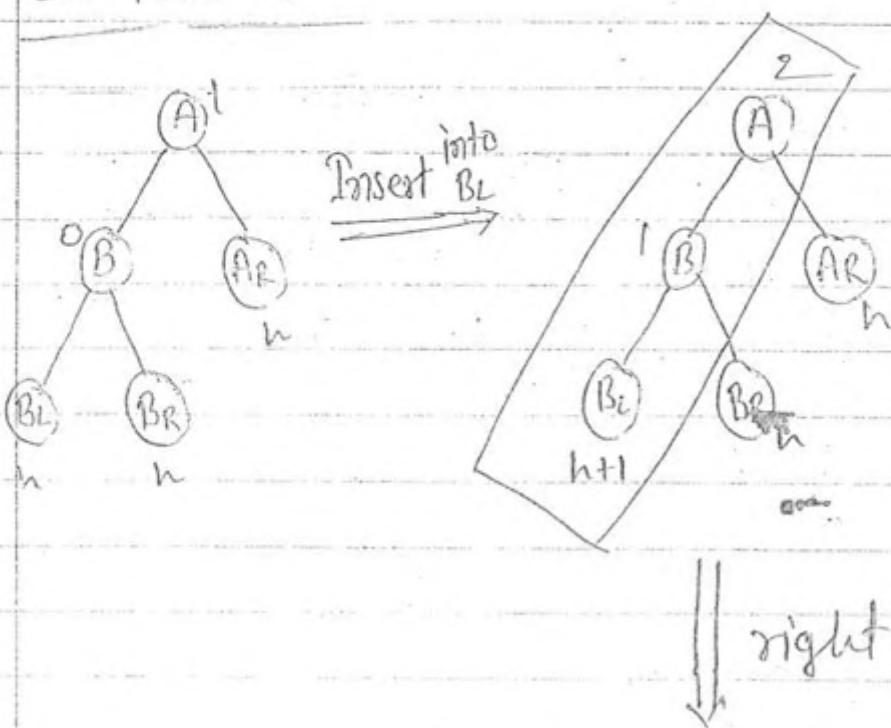
left

$$y \rightarrow p = n$$



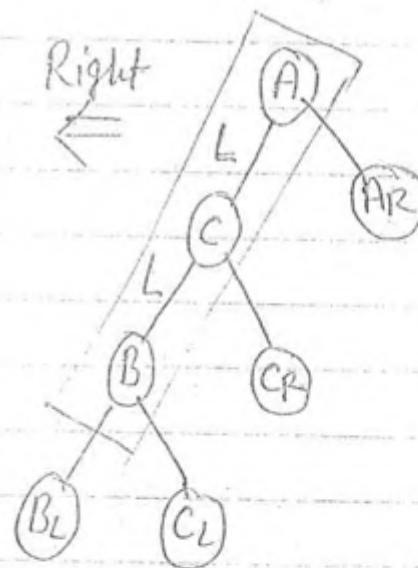
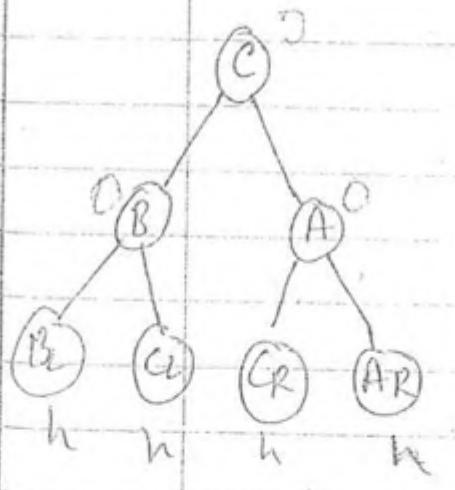
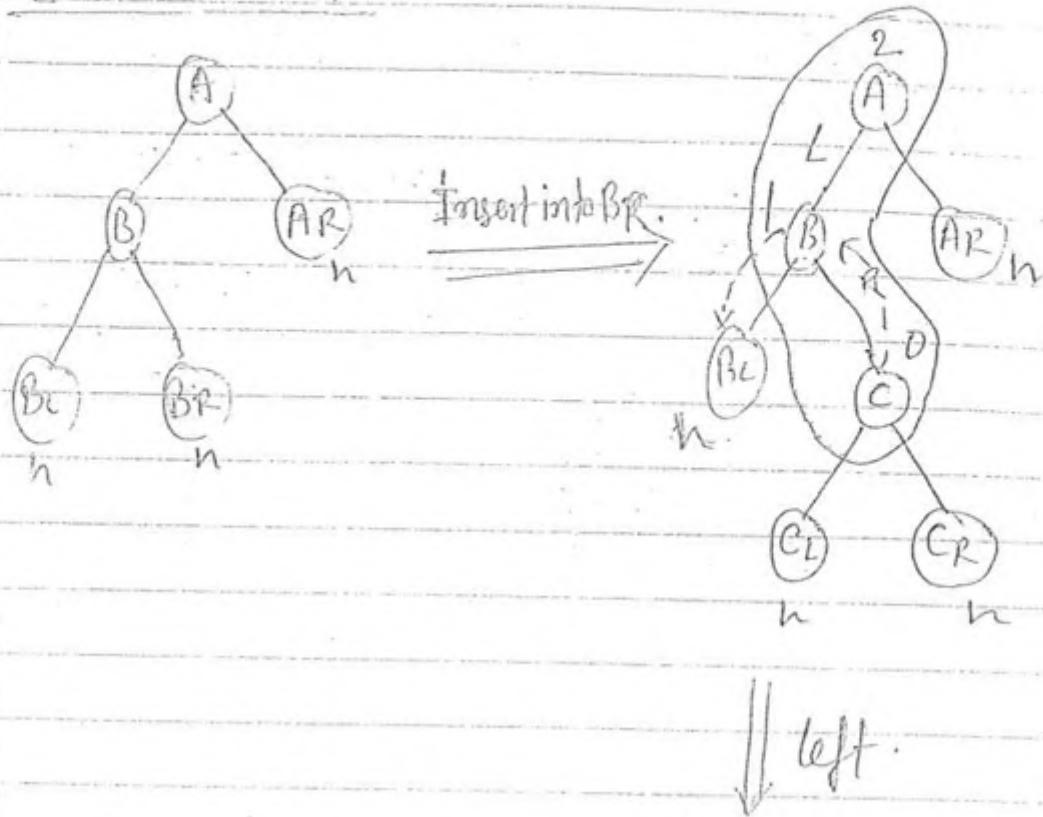
Insertion

LL Problem



Insertion

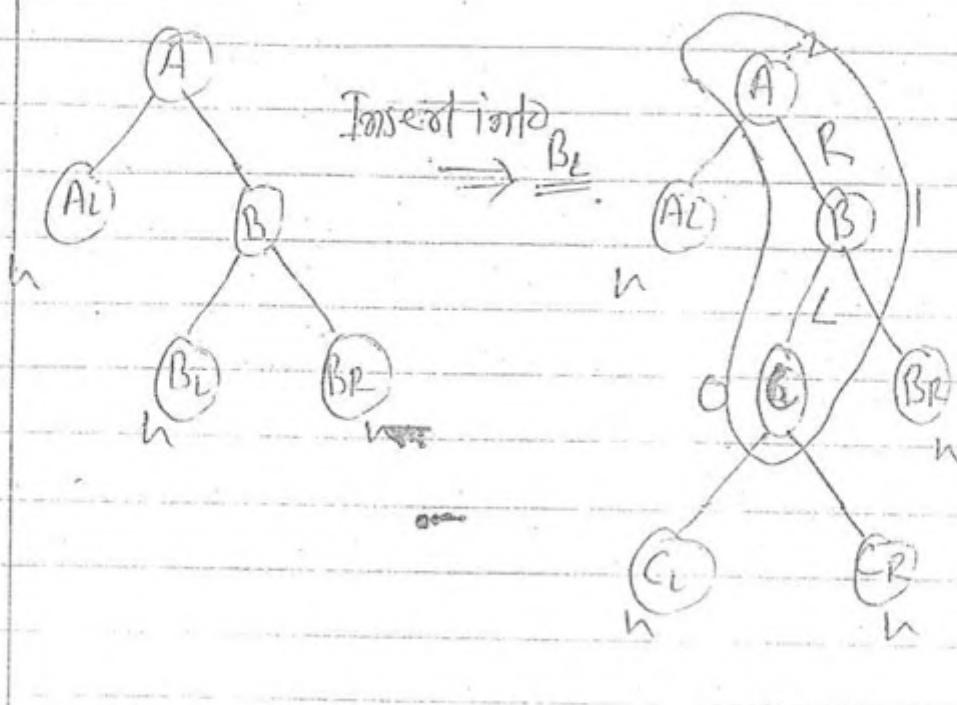
L R Problem.



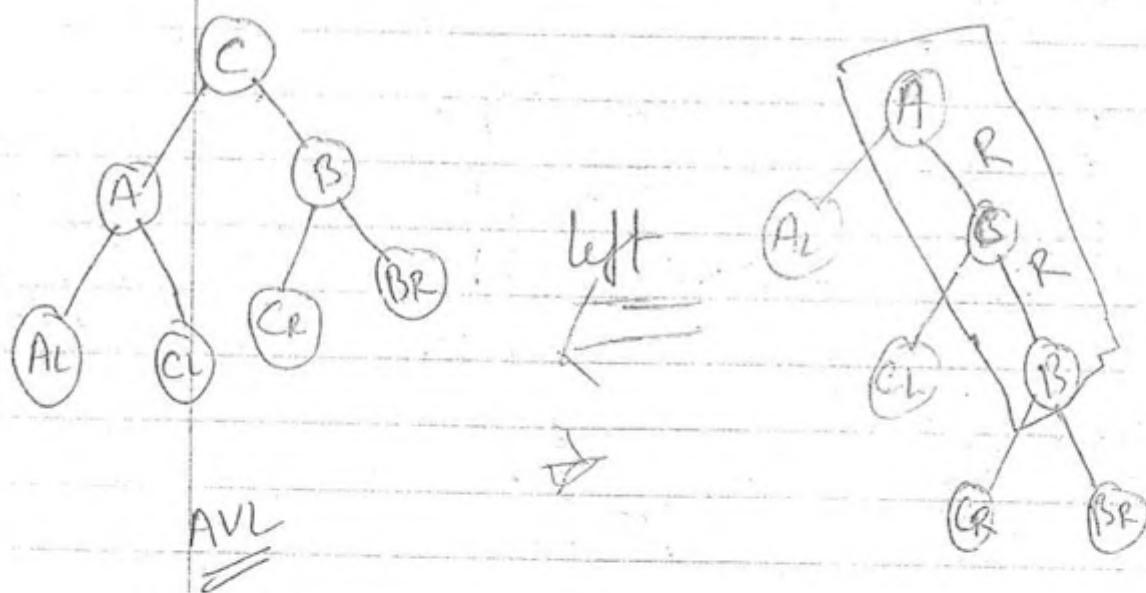
Rotation take $O(1)$

but if insertion take

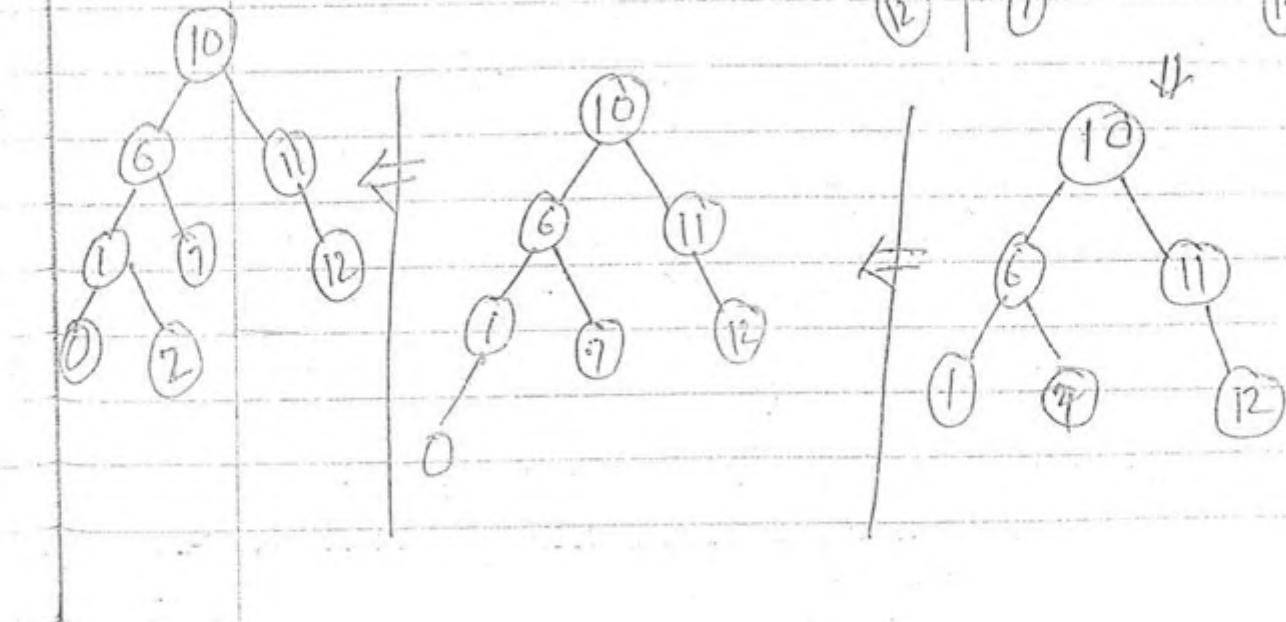
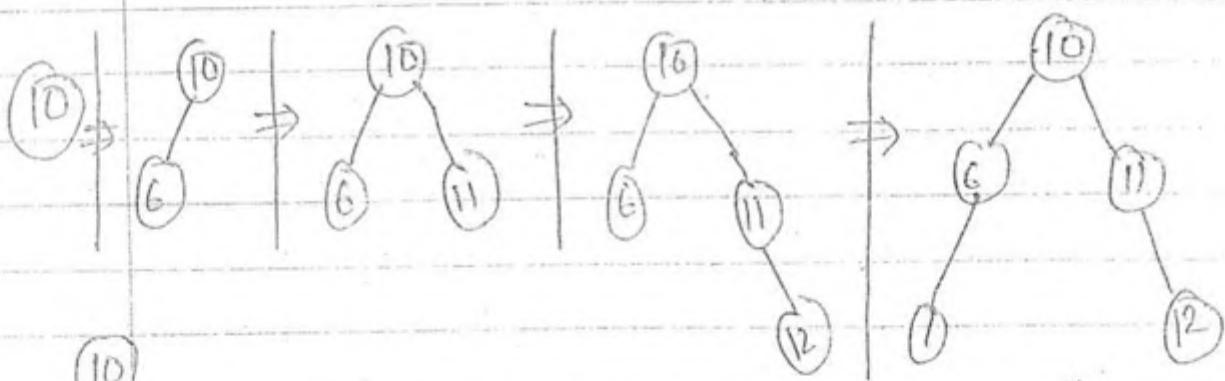
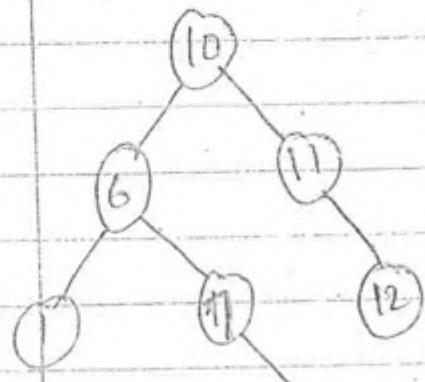
$O(\log n)$ time;

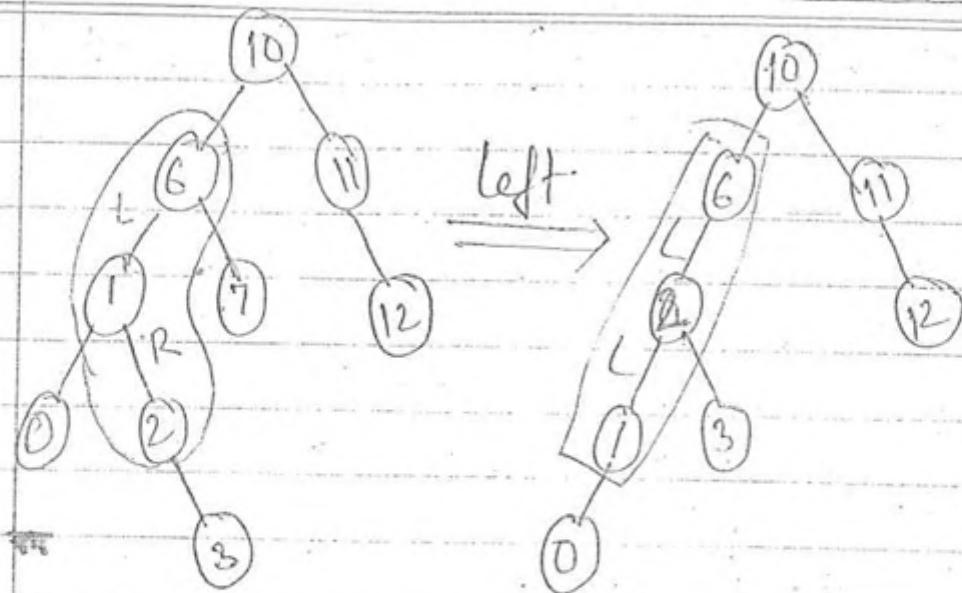
InsertionRL Problem

↓ Right.

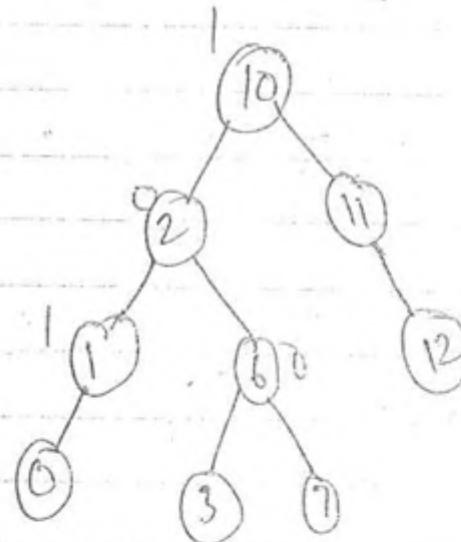


#9 What will be the resulting AVL tree if we insert 10, 6, 11, 12, 1, 7, 0, 2, 3 in that order.



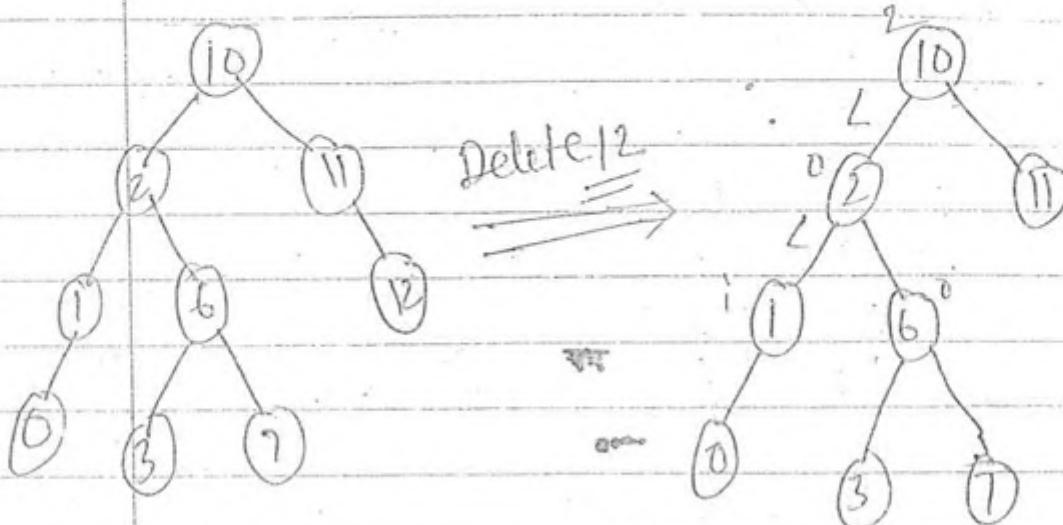


↓ Right

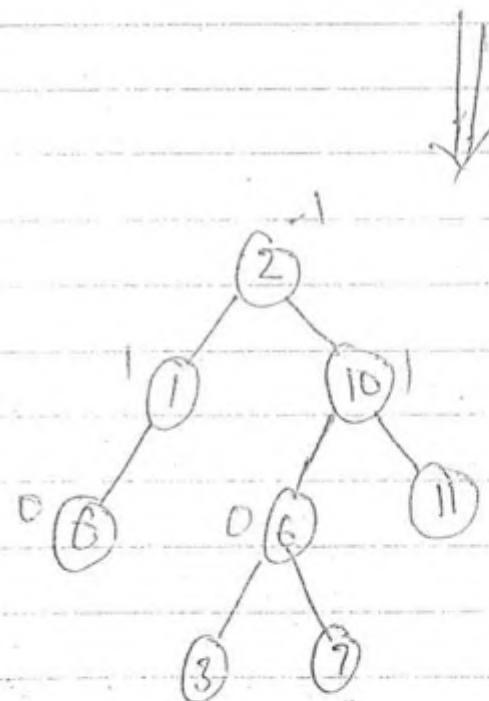


AVL Tree

Q What will be resulting AVL tree if we delete 12 from above AVL tree.



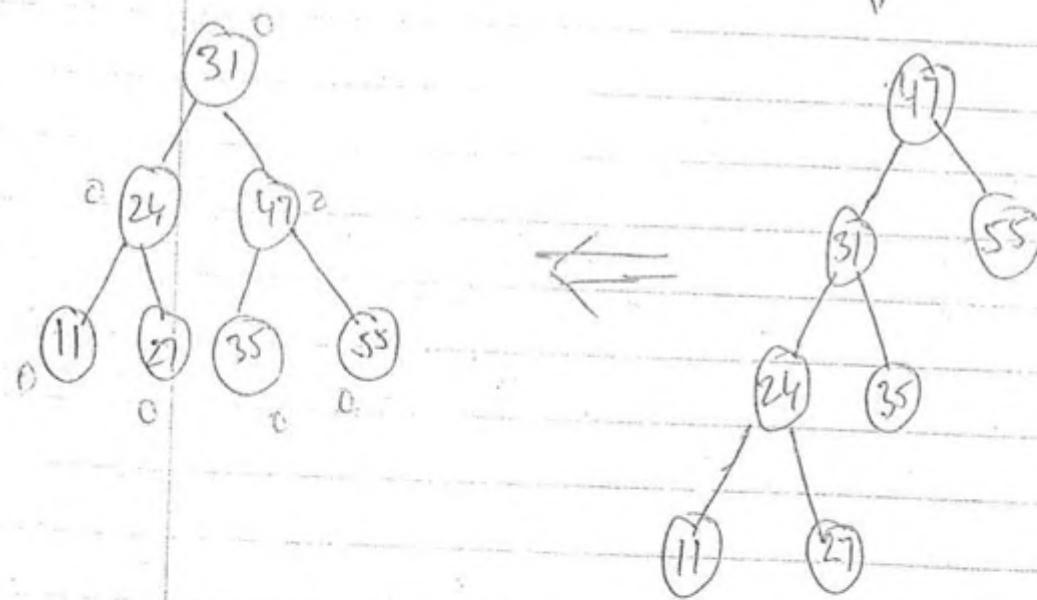
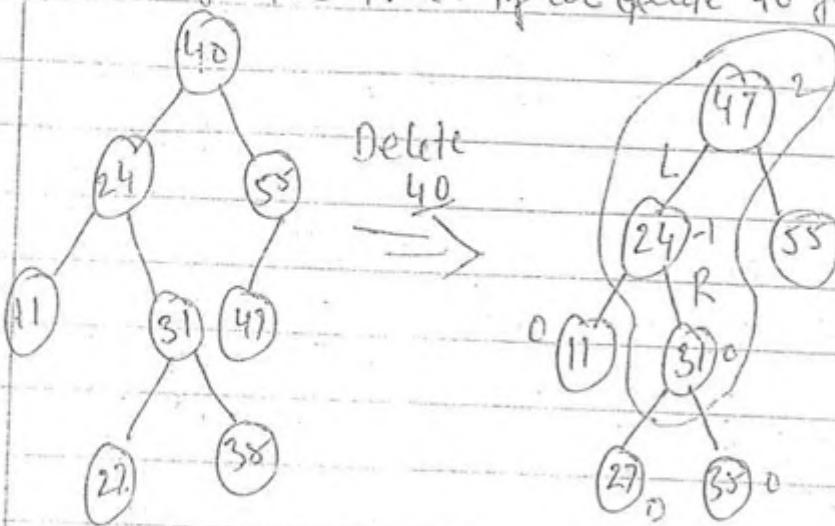
LL and LR got both problem but LL is better becoz LL has less time complexity we do 2 rotat



if we LL & LR problems we should go for LL.

#9

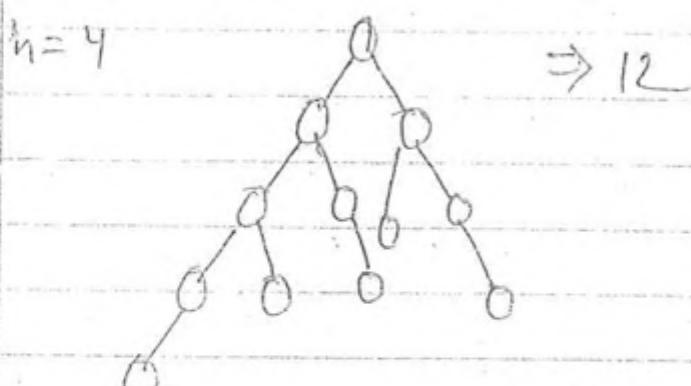
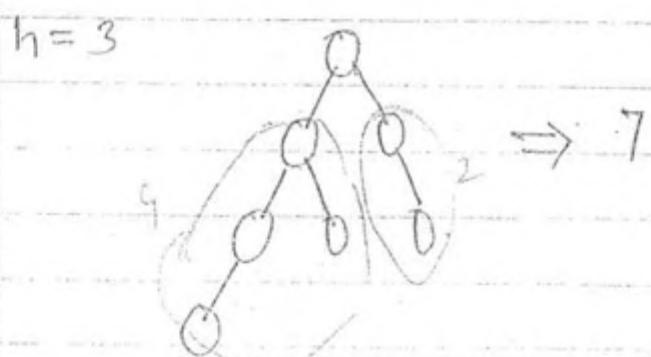
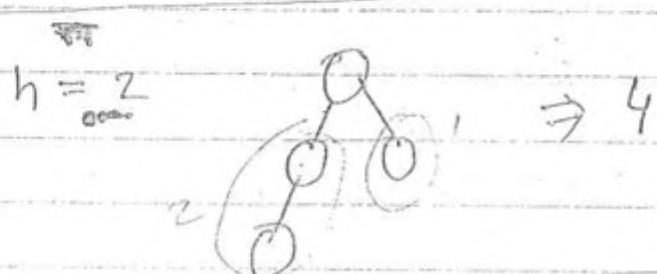
Consider the following AVL tree, what will be the resulting AVL tree if we delete 40 from above tree.



be the
vertex.

~~#8~~ What is the max^{!!} height of any AVL tree
with 7 nodes. ?

$$h=0 \quad \text{O} \quad \Rightarrow 1$$



Min^m no of nodes in height(h) AVL tree
is

$$\boxed{\text{Min}(h) = \text{Min}(h-1) + \text{Min}(h-2) + 1}$$

$$h=0 \Rightarrow 1$$

ie

$$h=1 \Rightarrow 2$$

$$h=2 \Rightarrow 4$$

$$h=3 \Rightarrow 7$$

$$h=4 \Rightarrow 12$$

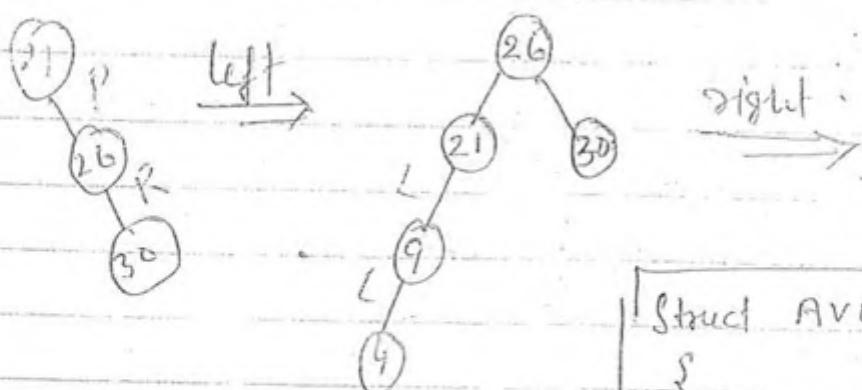
$$= 20$$

* 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.

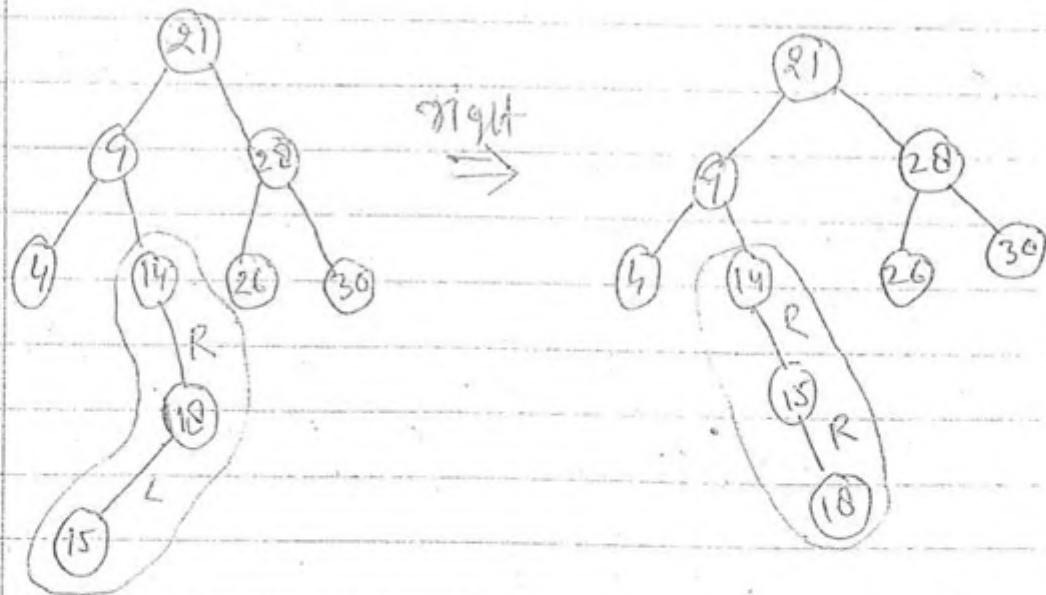
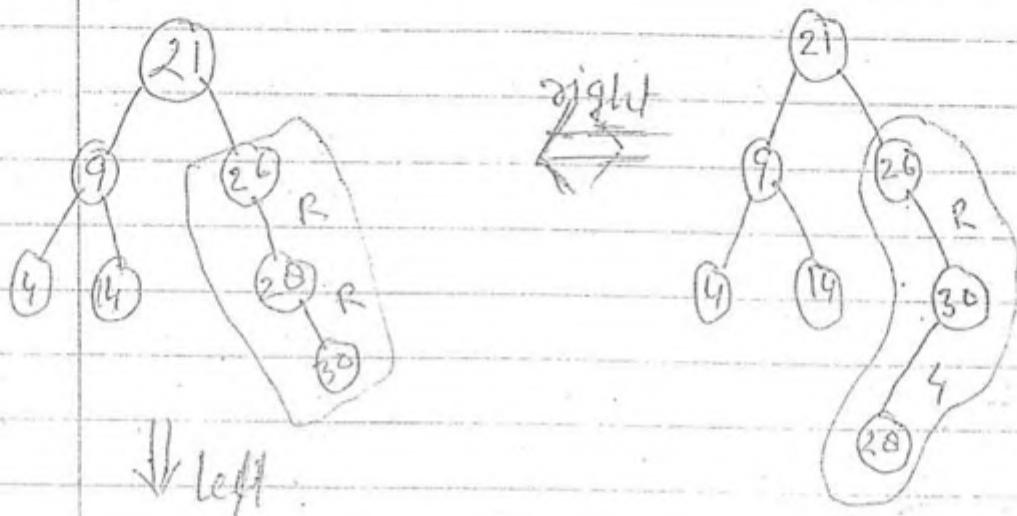
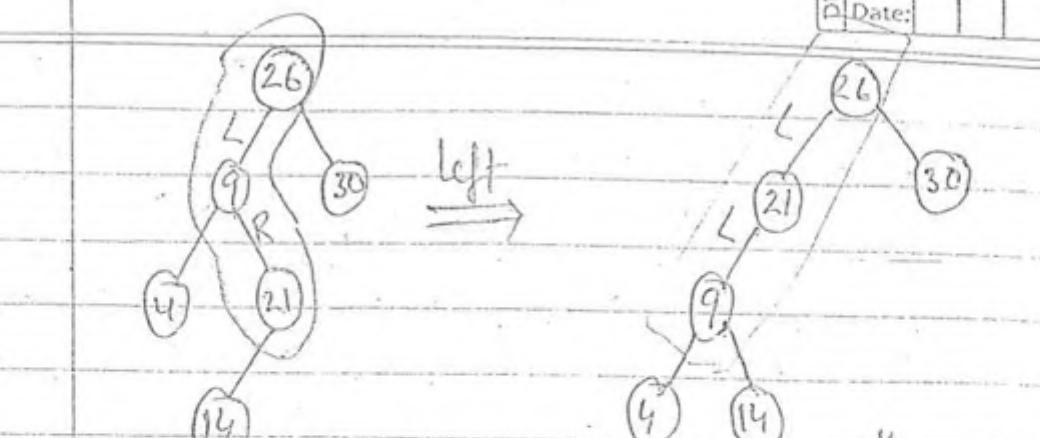


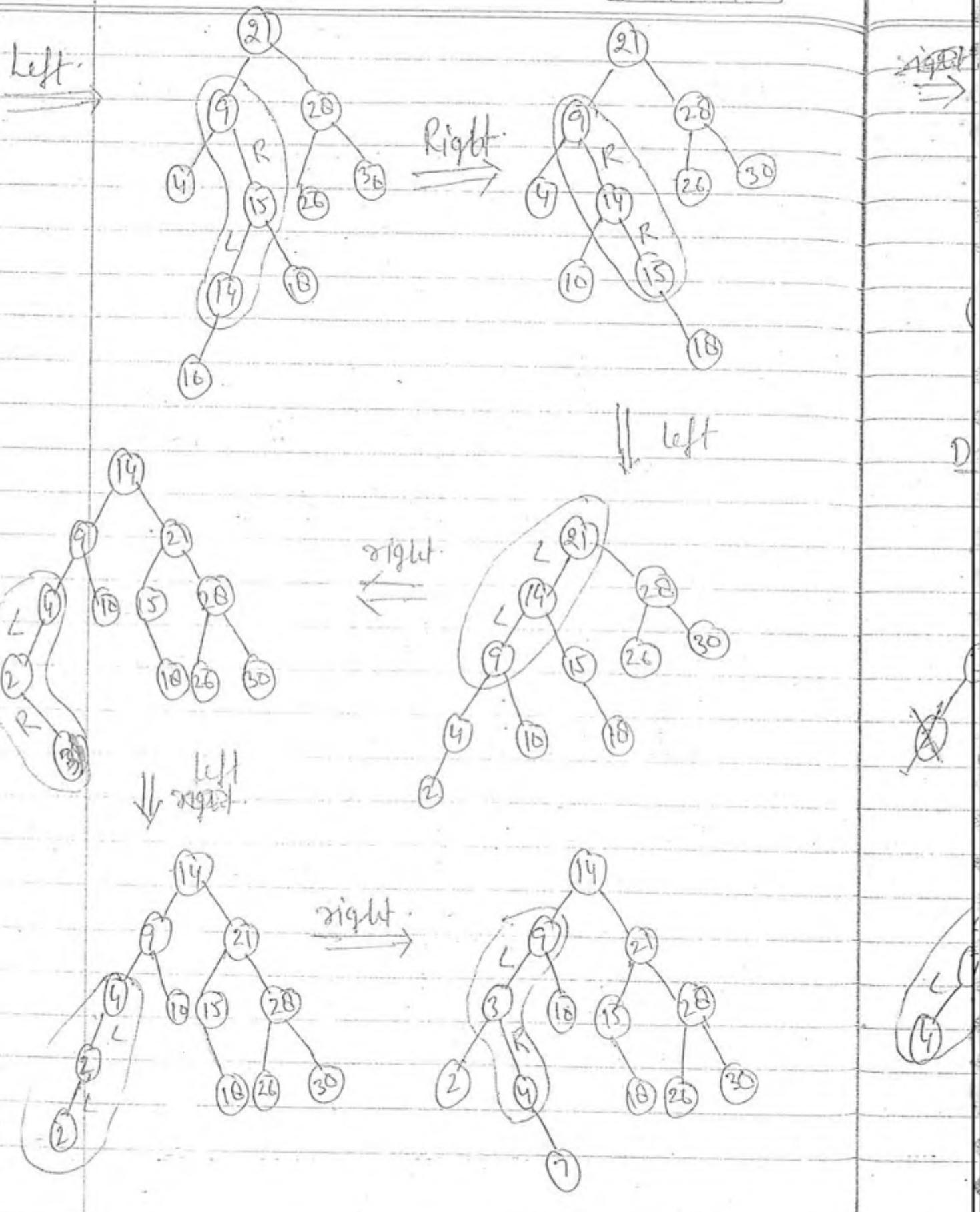
every node have height

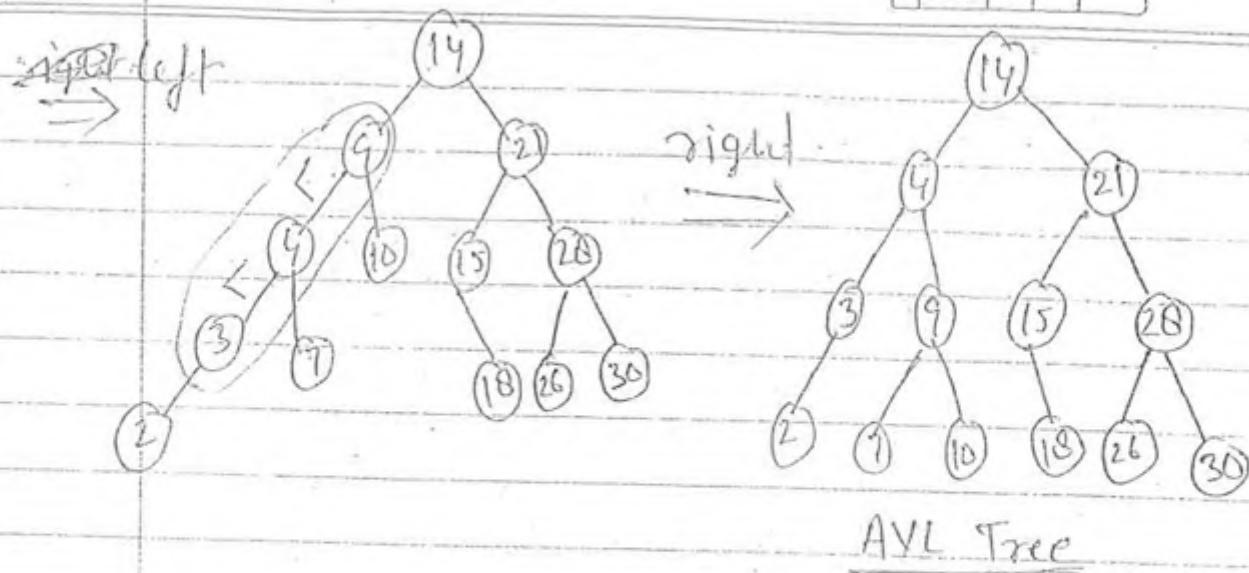
Struct AVL

{
int data;
}

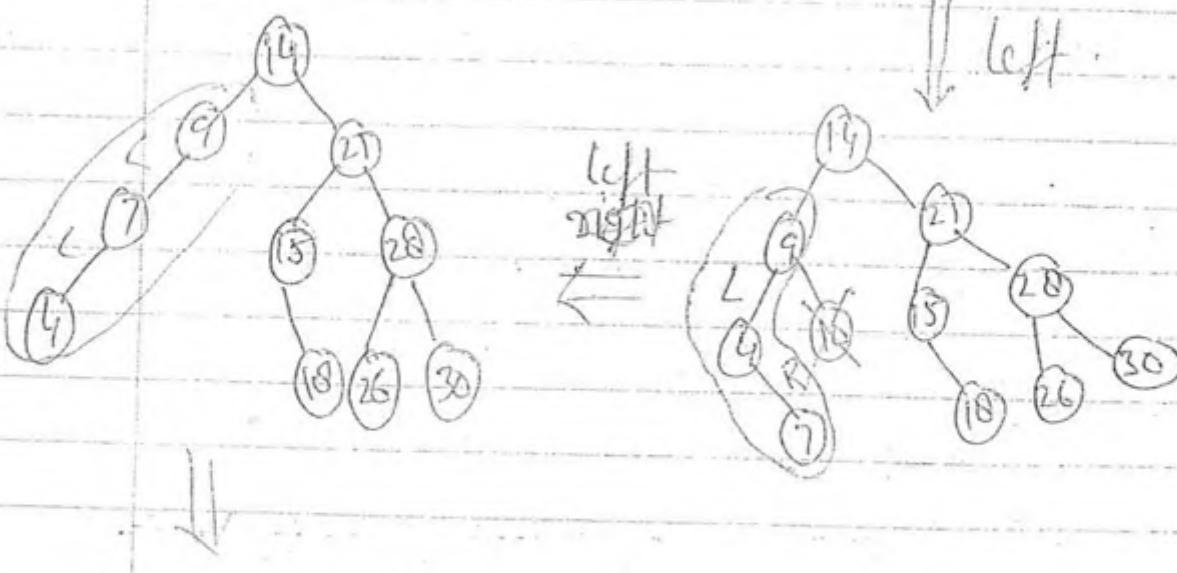
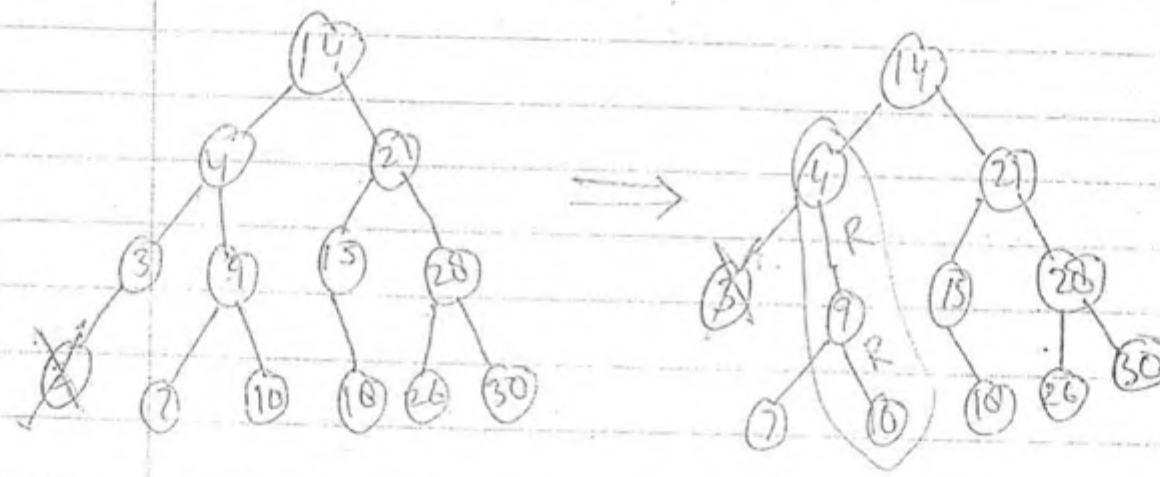
Struct AVL, *dp, **sp,
int height;
}

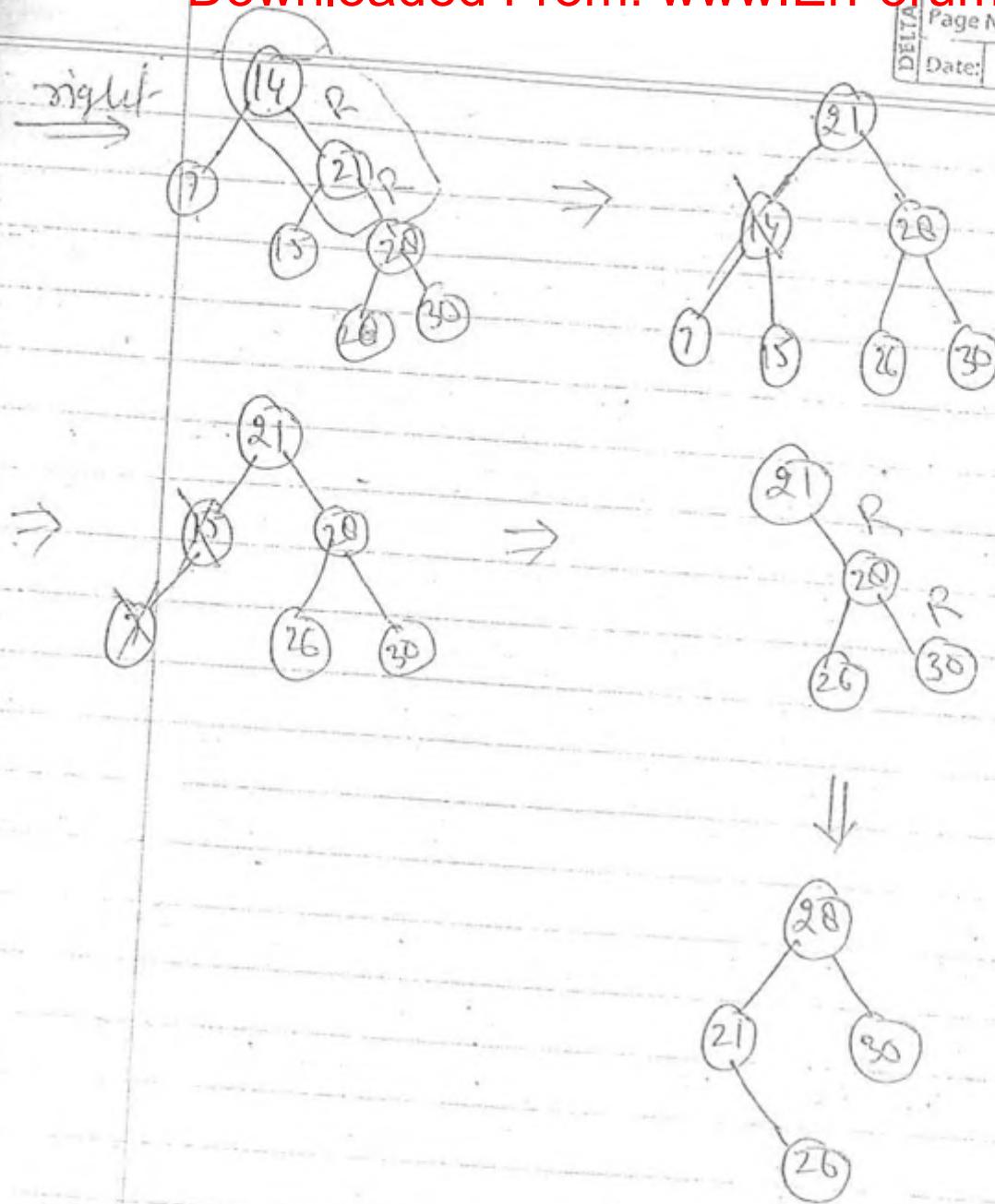






Delete $\Rightarrow 2, 3, 10, 18, 4, 9, 14, 7, 15.$





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.

$$= \Theta(\log_2 n)$$

③ Deleting an element from AVL tree the which already contain n elements.

$$= \Theta(\log_2 n)$$

④ In a AVL tree insertion or deletion takes $O(n)$ time. When AVL tree were no height.

~~A(P)~~

A(P)

{

if (P != Null)

{

if (P->data > (P->lP->data))

swap (P->data, P->lP->data)

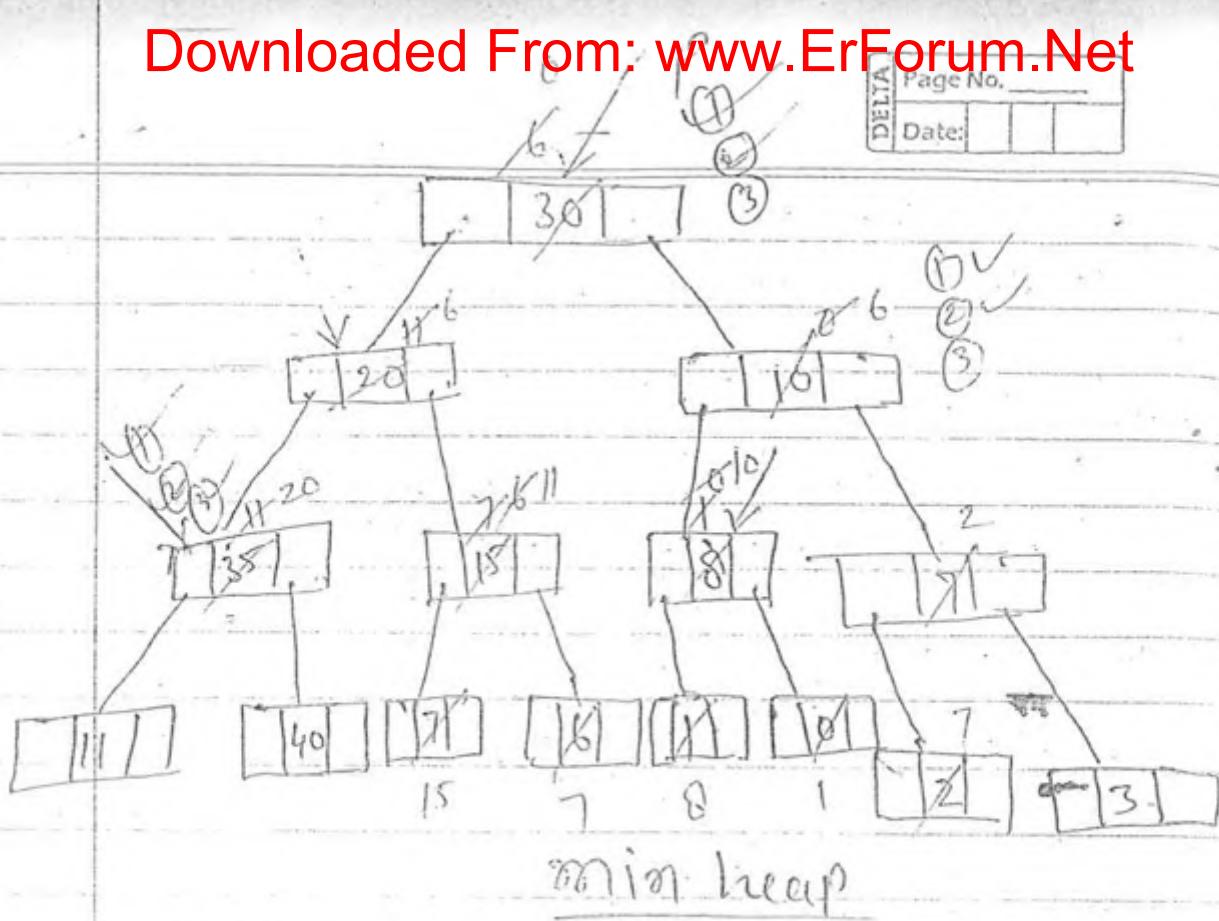
if (P->data > (P->rP->data))

swap (P->data, P->rP->data)

A(P->lP)

A(P->rP)

}



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)

- 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 (k keys) (max^m Keys)

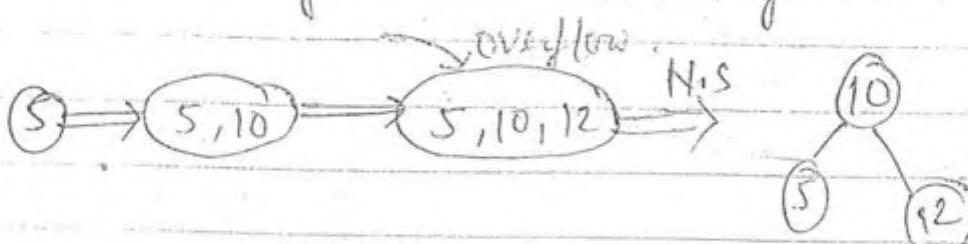
= $\lceil \frac{m}{2} \rceil$ (min^m children) \Rightarrow

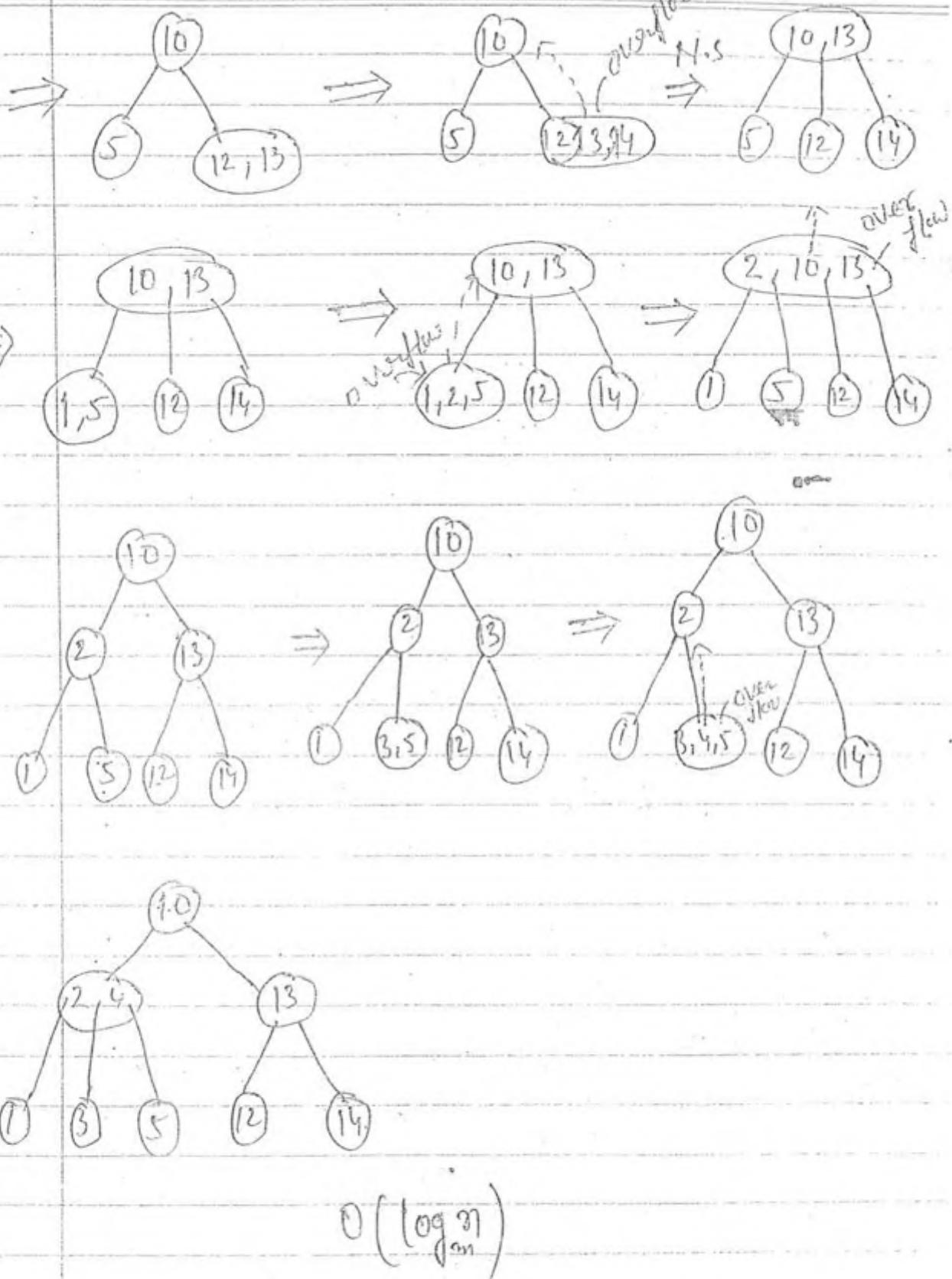
= $\frac{m}{2} - 1$ (min^m keys)

~~Ans~~ Consider the following elements 5, 10, 12, 13, 14,
1, 2, 3, 4, degree = 3

$$\text{max}^m \text{ children} = 2 \quad \text{min}^m \text{ children} = \frac{3}{2} = 1.5 = 2 \Rightarrow$$

$$\text{max}^m \text{ keys} = 2 \quad \text{min}^m \text{ keys} = 2 - 1 = 1$$





#

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

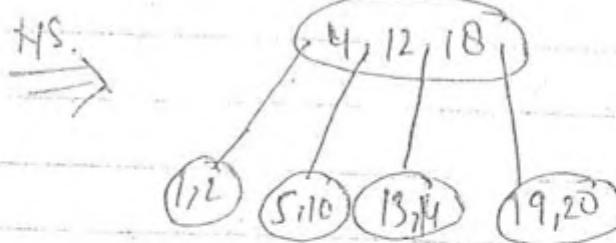
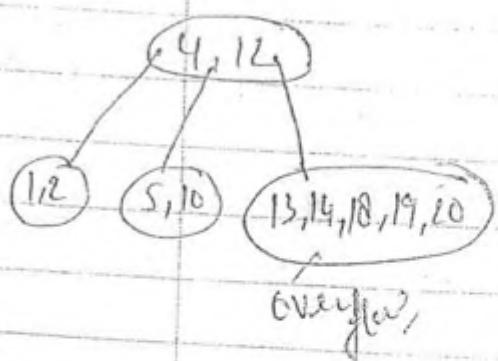
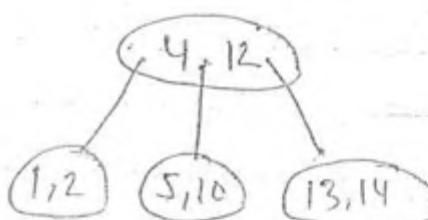
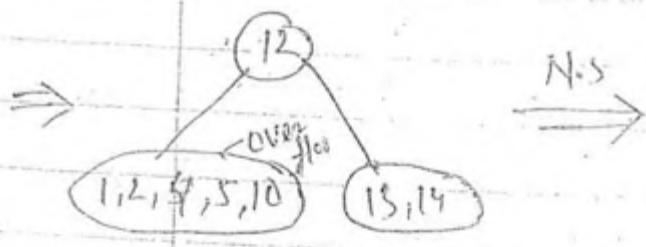
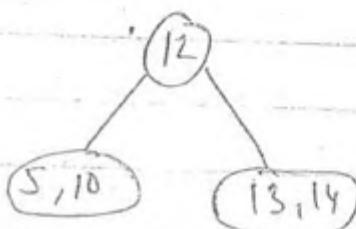
Min^m children = 3

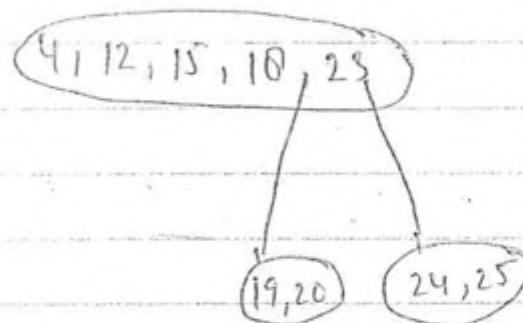
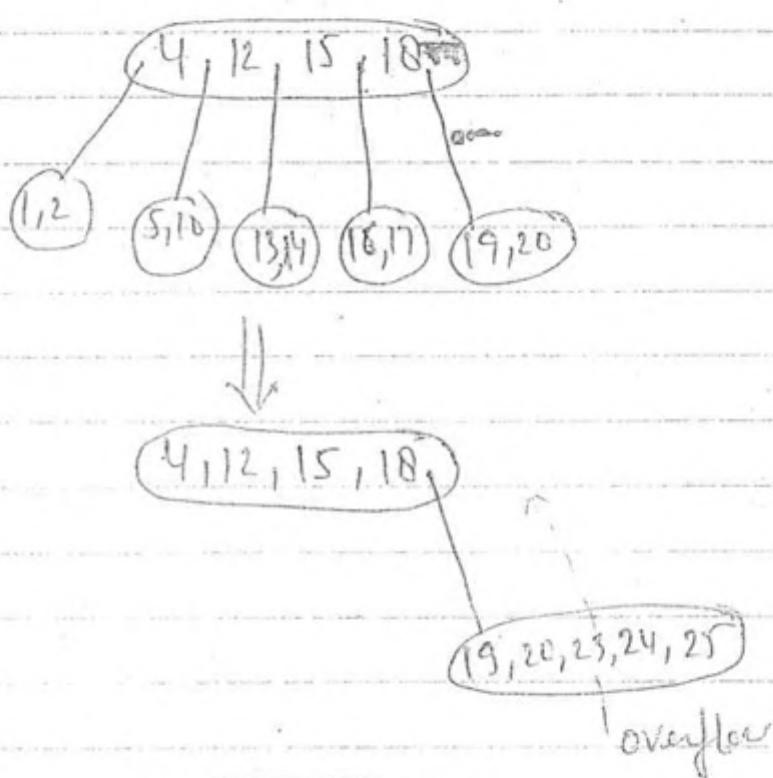
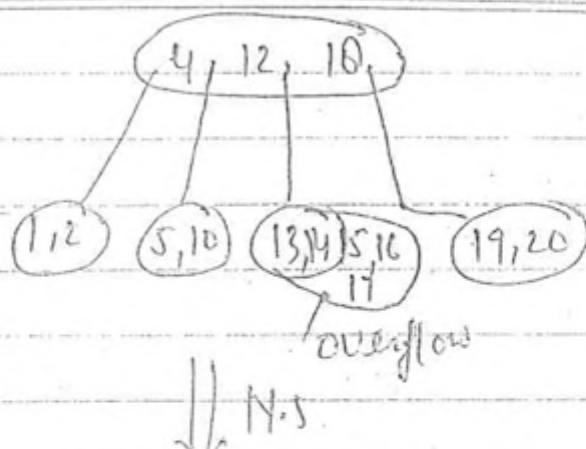
Keys = 2.

overflow

5, 10, 12, 13, 14

N.S.

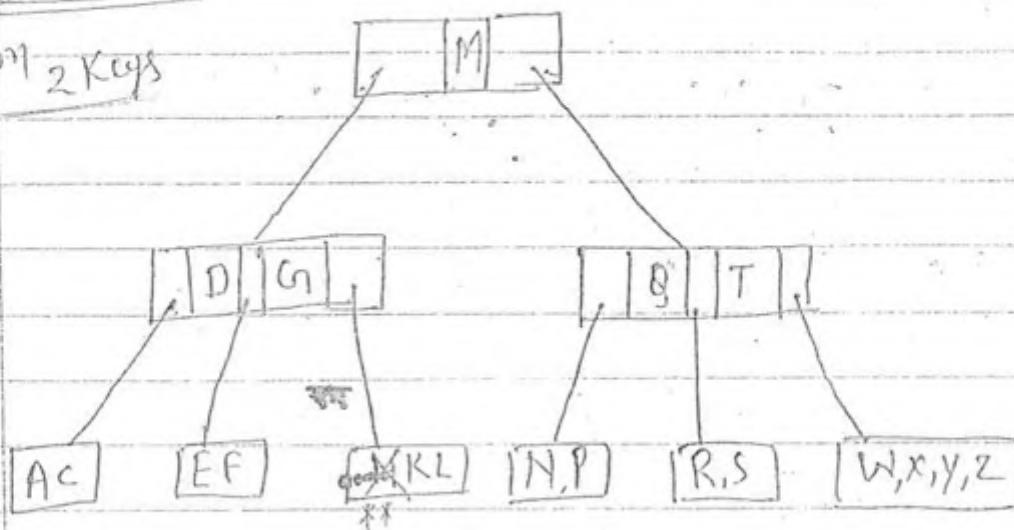




DELM	Page No.			
	Date:			

Consider the following B+ tree.
Deletion

Min 2 Keys



$$D = 6$$

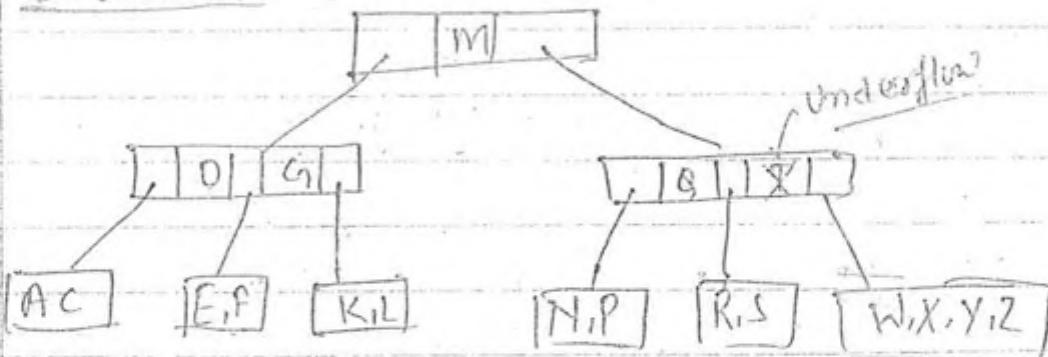
$$\text{max}^m \text{ch} = 6$$

$$K = 5$$

$$\text{min}^m \text{ch} = 3$$

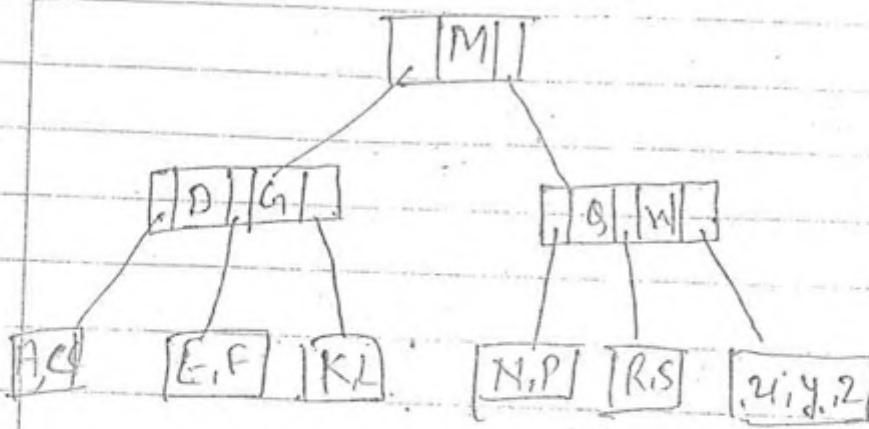
$$\text{Key} = 2$$

Delete-H

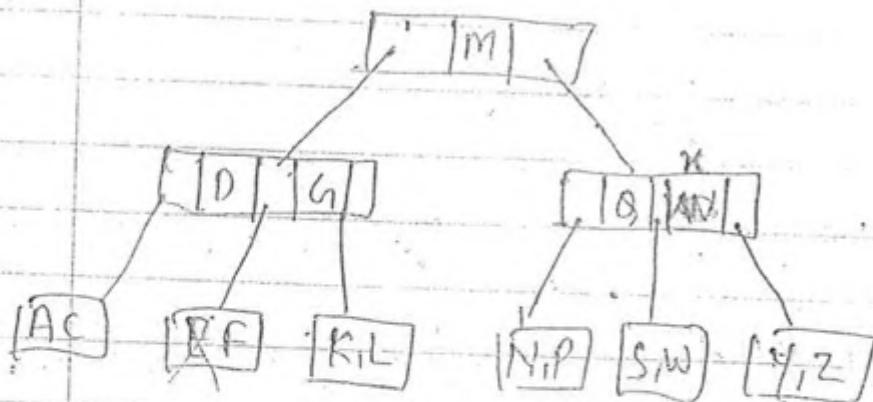
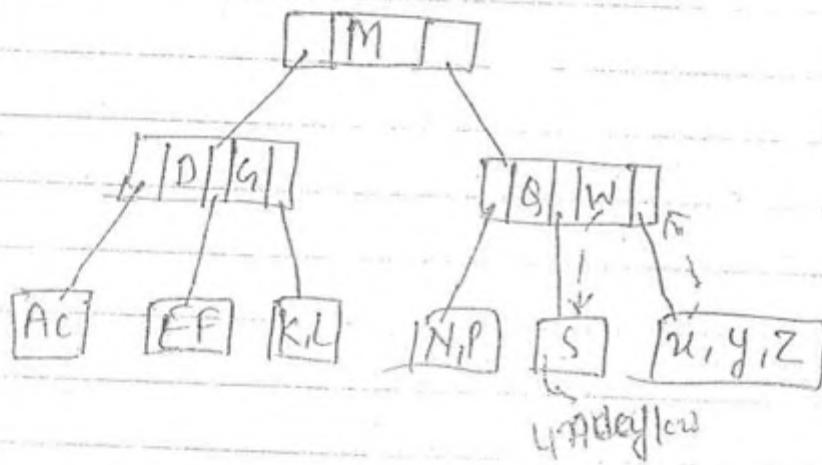


In the above tree after deleting T that node is in underflow but one of the children is having more than the required keys. So it will send

Delete - T

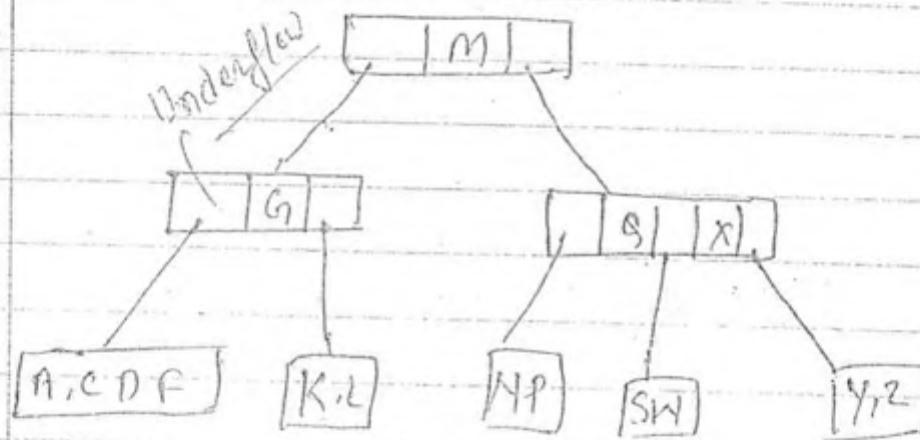
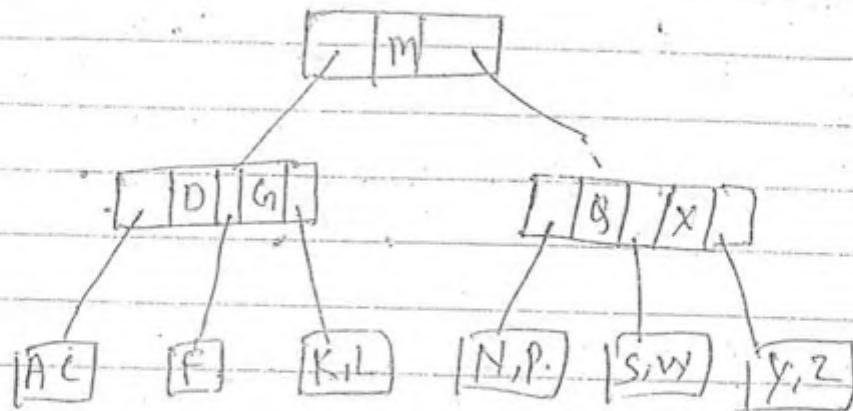


Delete - R



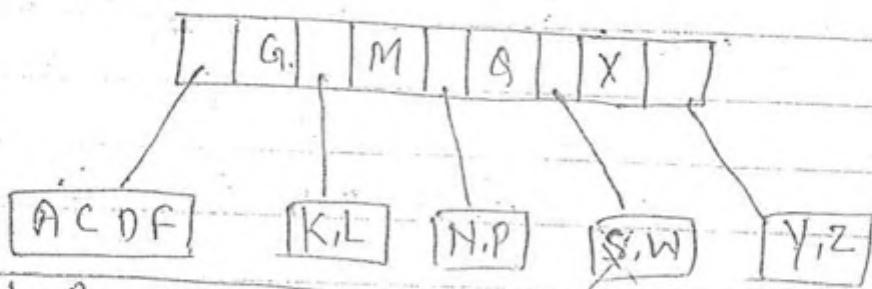
If
replace
of H
so
comb
is C

Delete E



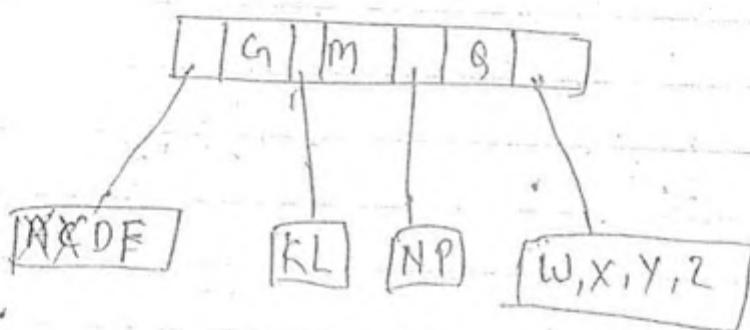
If you delete element E from the above 18-tree replacement is one of the Keys from any one of the Sibling. but if simply it not ready. so final step is made combining. Node combining nothing but grouping two sibling including parent.

Now the node G is underflow. No sibling is ready to help.



Delete A and C. No Problem

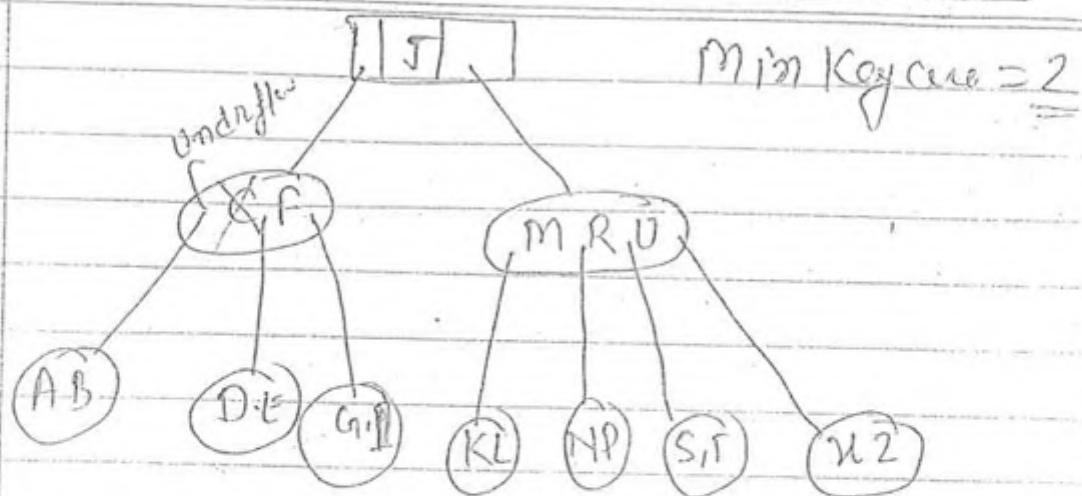
Delete S



The min^m condition should be satisfied other than the root entry.

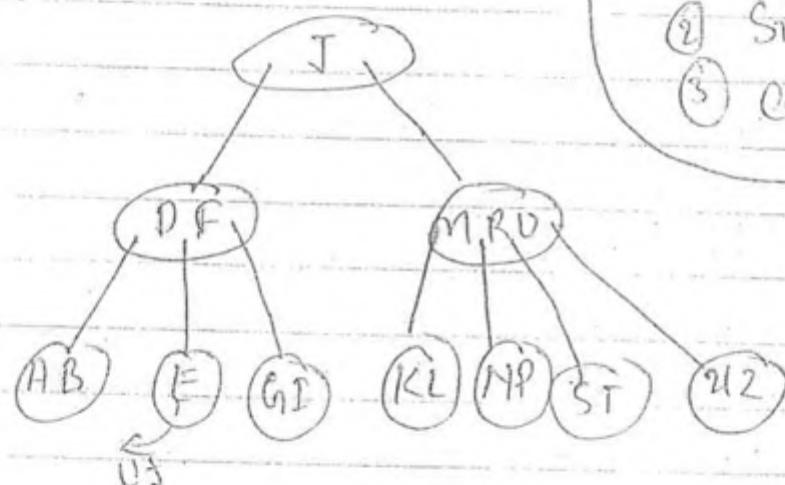
DETA	Page No.
	Date:

#9



Sq^n

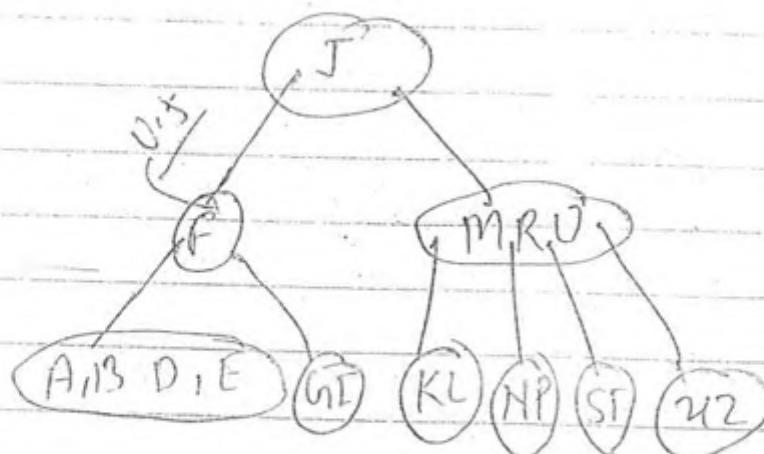
Delete

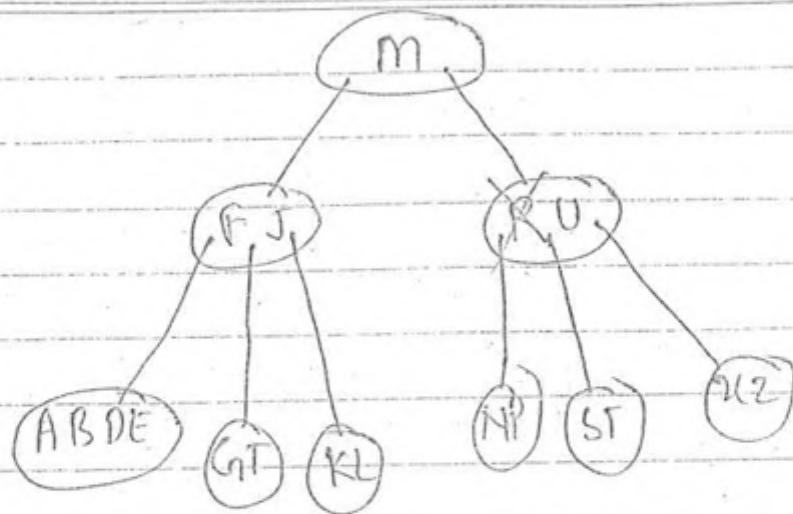


(1) Children

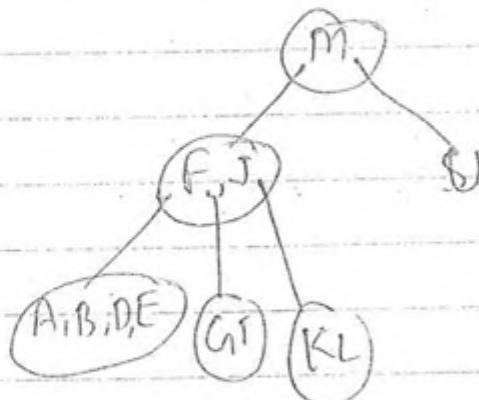
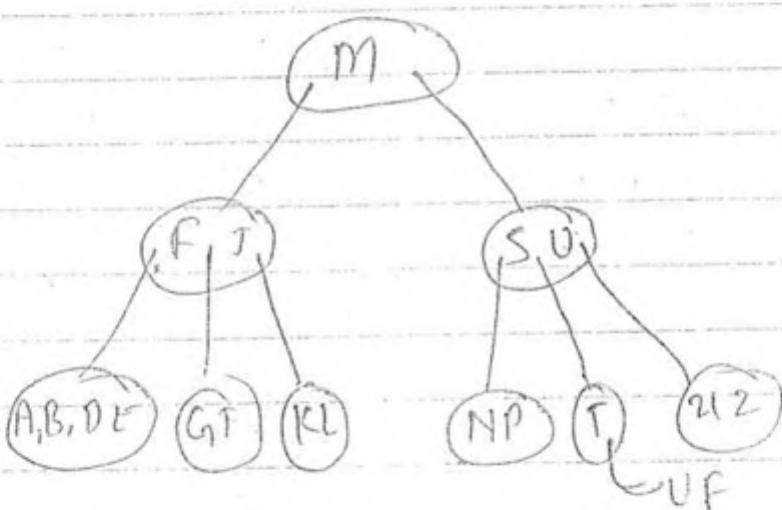
(2) Sibling

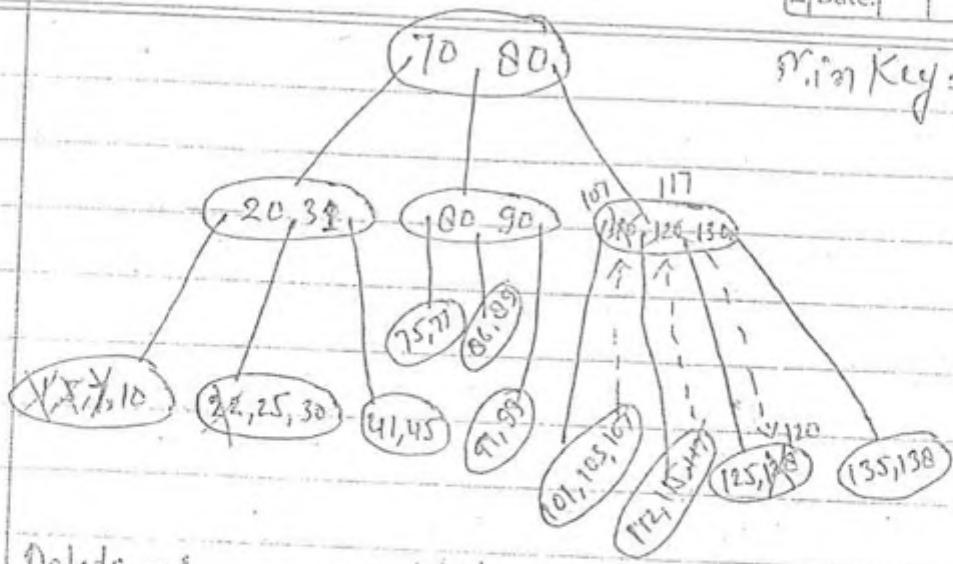
(3) Combined





Pelite R





Delete - 1 — delete directly . . . no effect

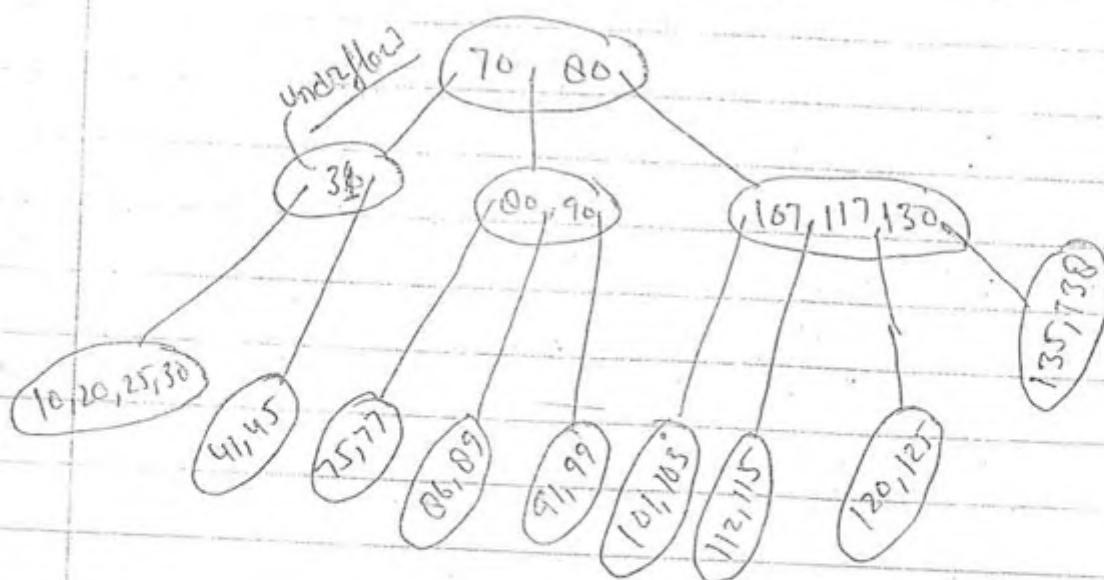
Delete - 22 —

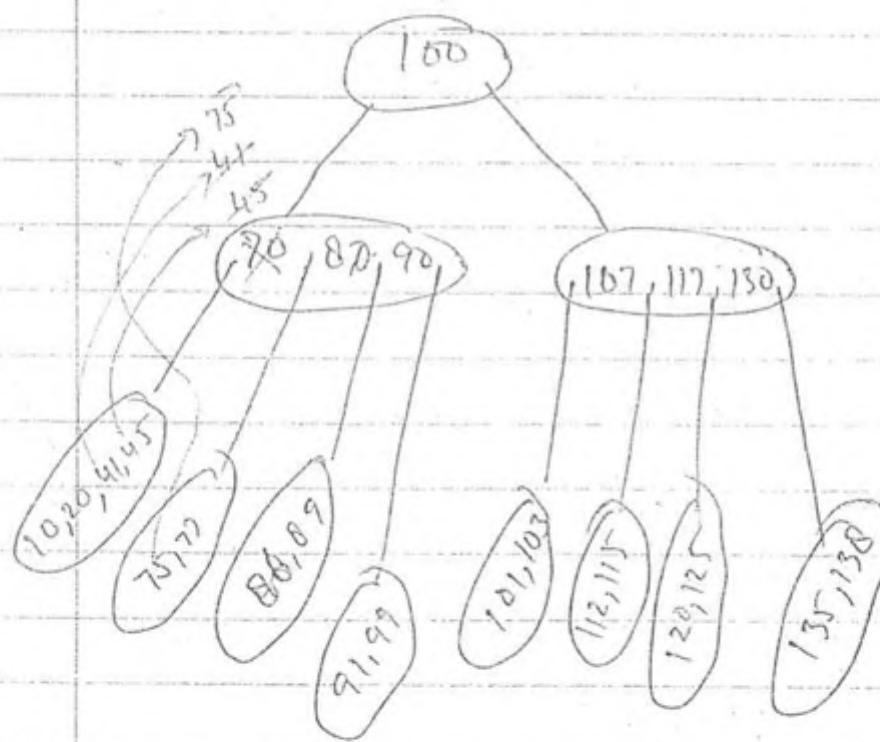
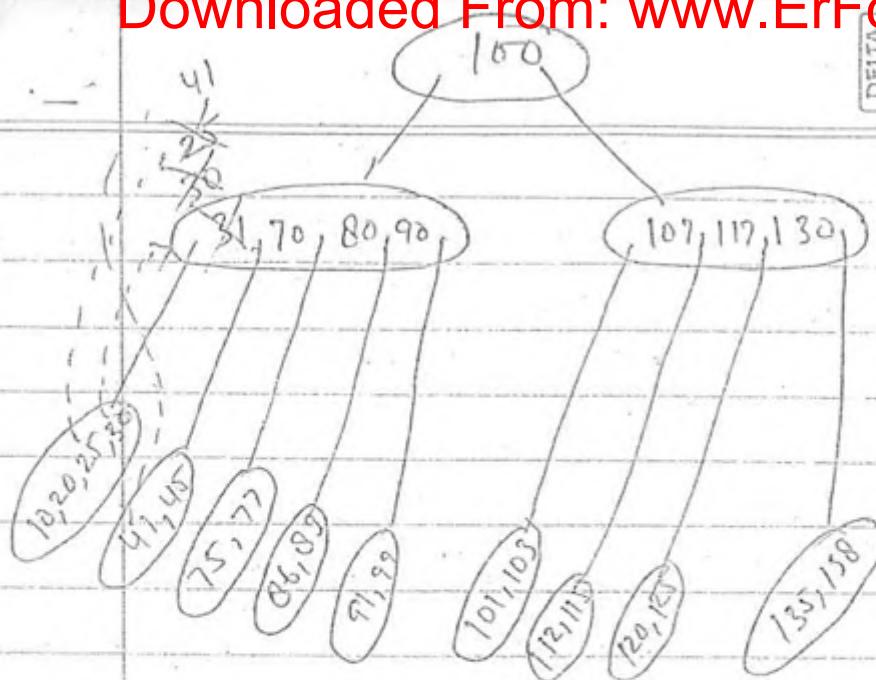
Delete - 5 —

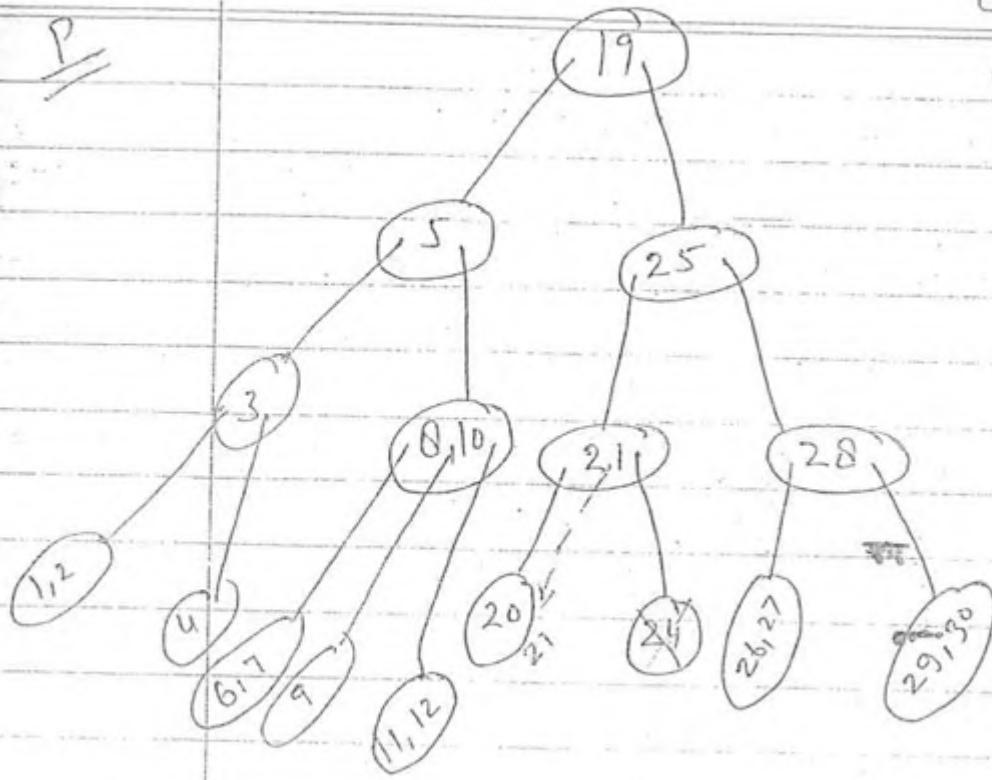
Delete 120 —

Delete 110

Delete 7 after deleting (7)

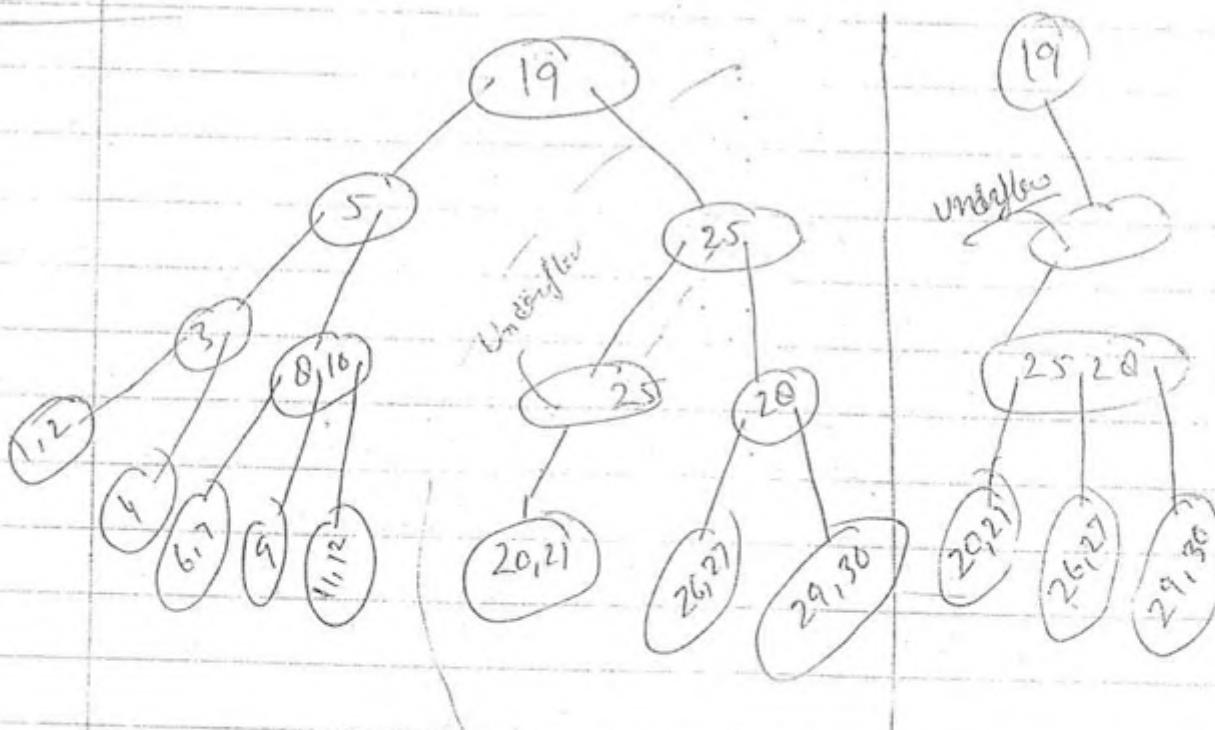


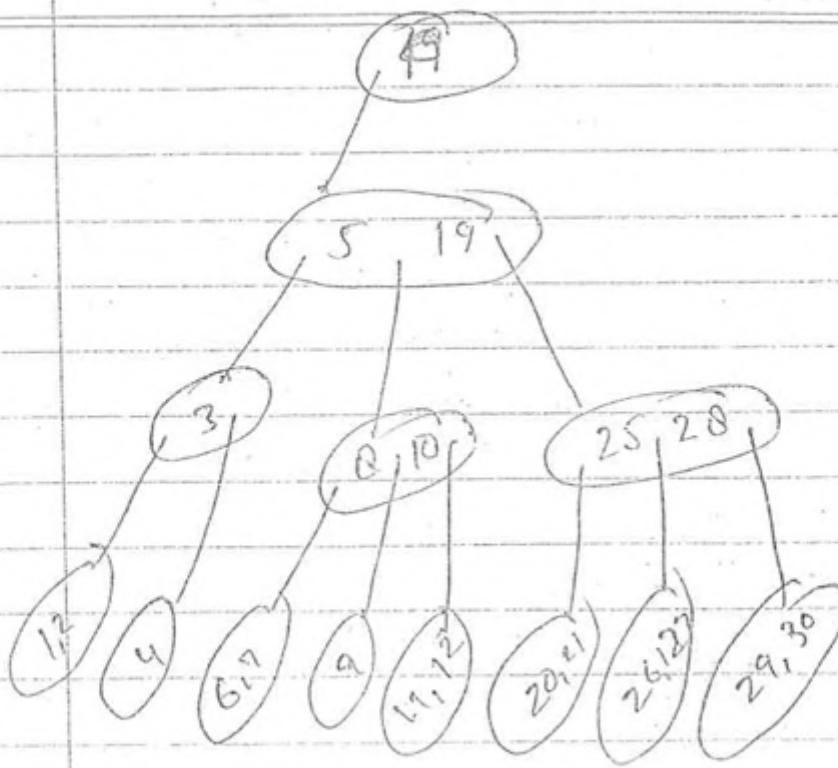




Min Key = 1

Delete 24





B tree of order 32, m = 32, then.

- (1) Find the max^m key that can accommodate in level 3 B-tree
- (2) find out min^m no key that can accommodate in level 3 B-tree,

Max^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

Let T be a B-tree of order m of height h .
 If no total key = ?
 $n = ?$

Let T be a B-tree of order m and height h .
 if n is the number of key elements in
 ~~T~~ then the max^m value of n is:

level
1 $m-1 \rightarrow m$

2 $m(m-1) \rightarrow m^2$

3 $m^2(m-1) \rightarrow m^3$

4 $m^3(m-1) \rightarrow m^4$.

$$m-1 + m(m-1) + m^2(m-1) + m^3(m-1) + \dots + m^h(m-1)$$

$$m-1 [1 + m + m^2 + m^3 + \dots + m^h]$$

$$m-1 [m^0 + m^1 + m^2 + \dots + m^h]$$

$$m-1 \left[1 \left(\frac{m^{h+1} - 1}{m-1} \right) \right] \Rightarrow \boxed{m^{h+1} - 1}$$

DRDO

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.

Level	nodes	Key	Children
1	1	1	2

2	2	$2 \times (\frac{d-1}{d})$	$2 \times d$
---	---	----------------------------	--------------

3	$2 \times d$	$2 \times d(\frac{d-1}{d})$	$2d^2$
---	--------------	-----------------------------	--------

$$\begin{aligned}
 &= 1 + 2(d-1) + 2d(d-1) + 2d^2(d-1) + \dots + 2d^{h-1}(d-1) \\
 &= 1 + 2(d-1) \left[d^0 + d^1 + d^2 + \dots + d^{h-1} \right]
 \end{aligned}$$

~~$$\begin{aligned}
 &\approx 2(d-1) \cdot 1 + 2(d-1) \left[1 \left(\frac{d^h - 1}{d-1} \right) \right]
 \end{aligned}$$~~

$$= 2d^h - 2 + 1$$



$$2d^h - 1$$

8

B-tree

Degree 4

4

max^m children = 4

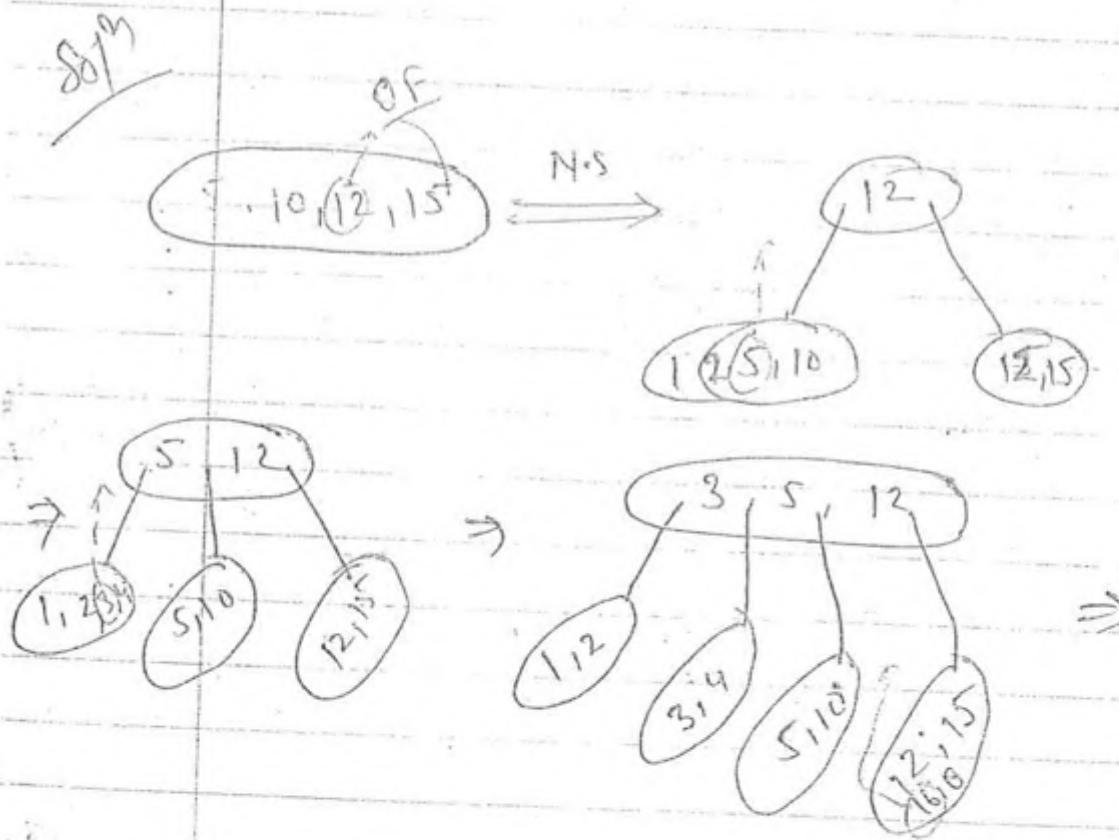
Key = 5

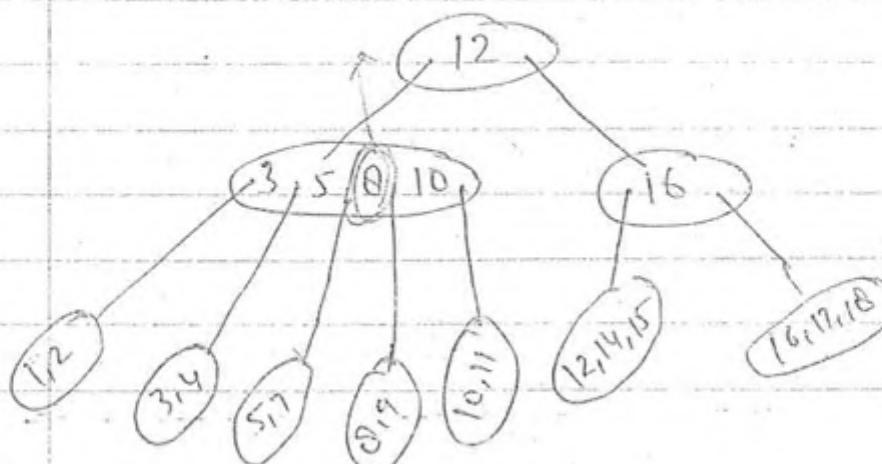
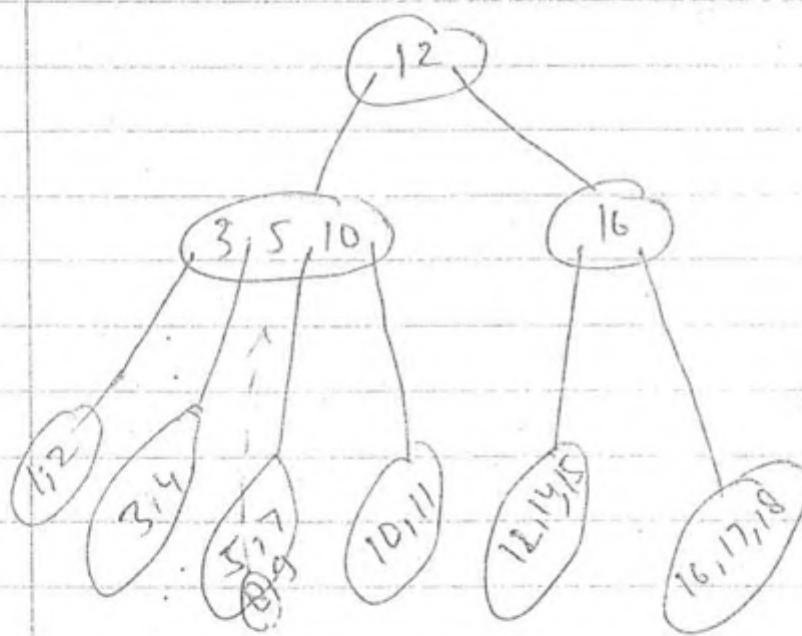
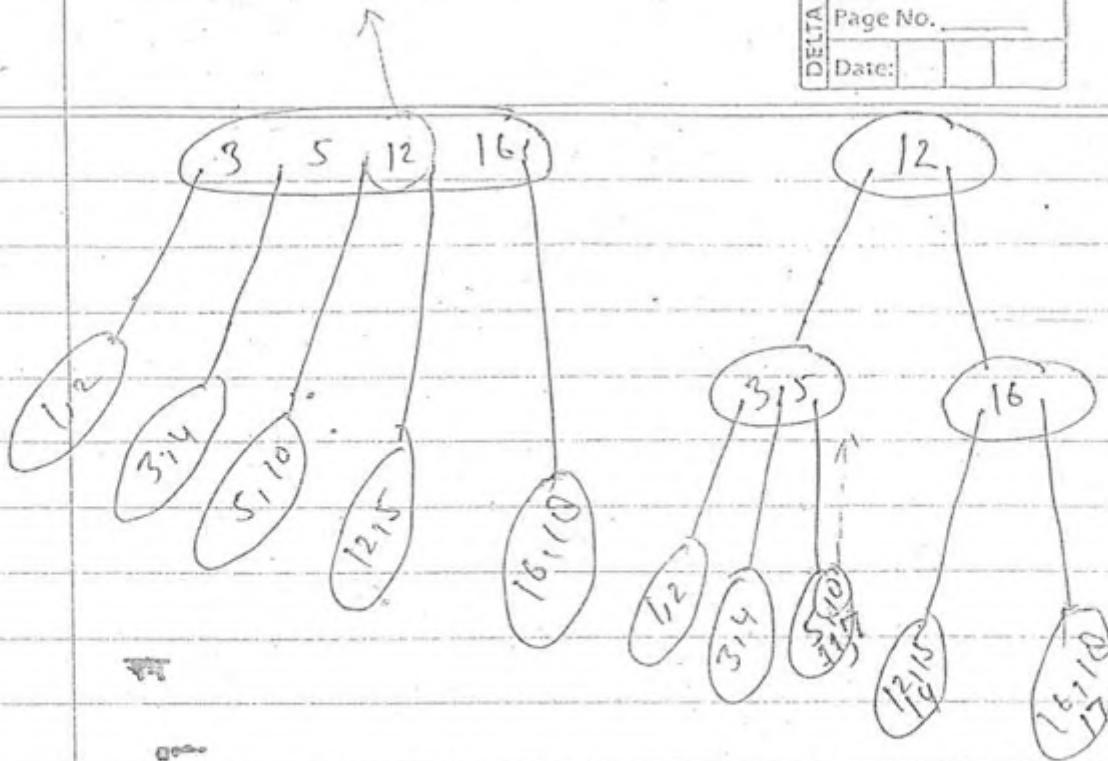
min children = 2

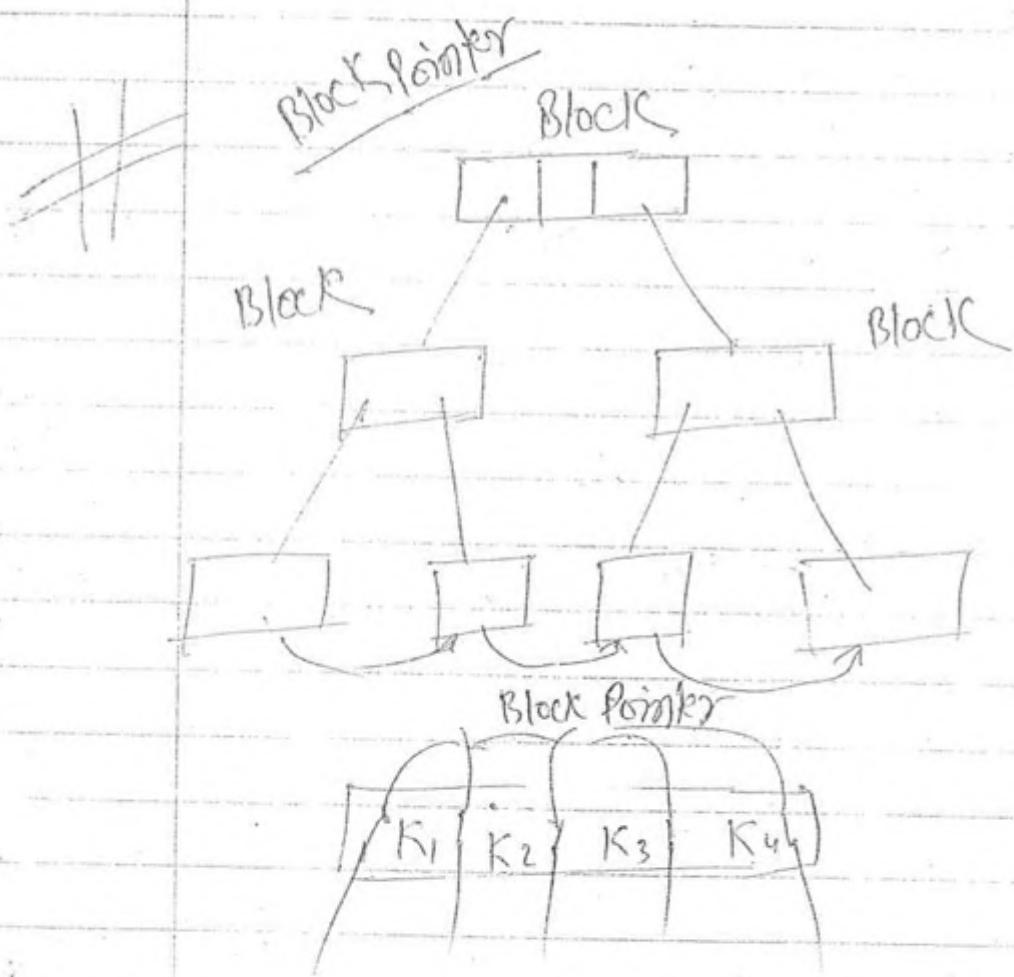
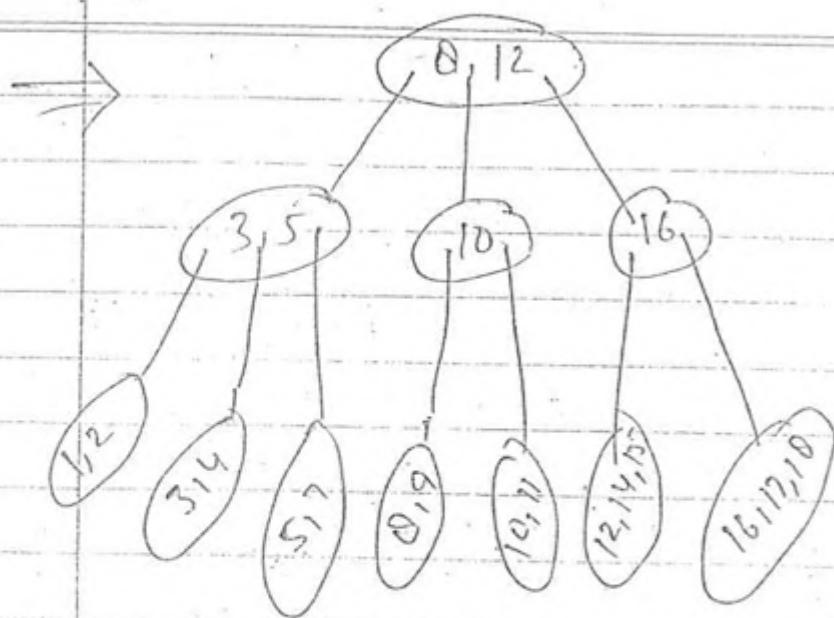
Key = 1

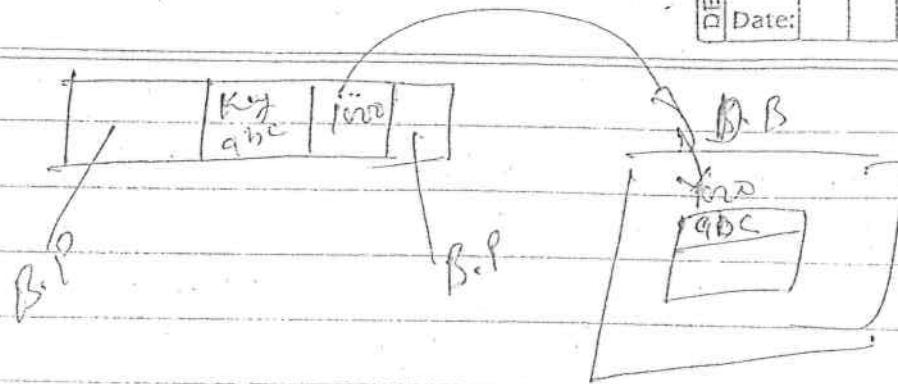
Consider 'B-tree' for the following :

5, 10, 12, 15, 11, 2, 3, 4, 16, 18, 17, 19, 11, 7, 8, 9







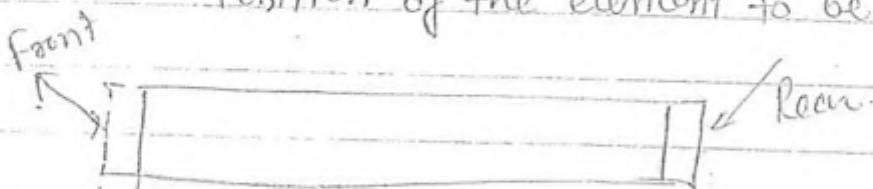


- Queue

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 Queue.

Enque (Queue , element) = Queue .

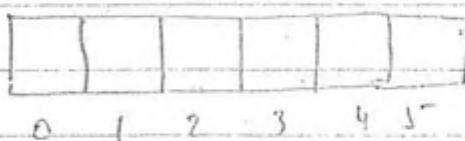
Deque (Queue) = element .

Isfull (Queue) = Boolean .

IsEmpty (Queue) = Boolean .

Enque $N = 6$

Q.

 $F = -1$ $R = -1$

(all time both will be -1)

Void Enque(Q, N, F, R, u)

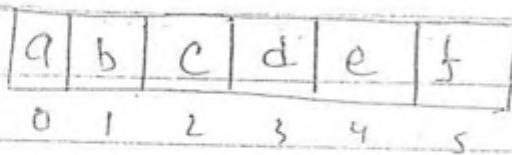
```
{
    if (R == N-1)
        Print("Queue is full");
        exit(1);
    else
}
```

```
{
    if (R == -1)    } → first Insertion
        F = R = 0;
    else
}
```

```
else
{
    R++; } → remaining
    Q[R] = u;
```

Queue (It is a operation)
 $N=6$

Q.



$$F=0 \\ R=5$$

Dequeue (Q, N, F, R)

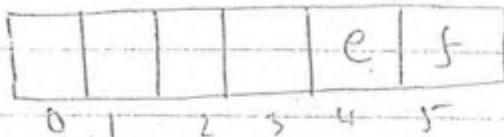
```
{
    int y;
    if (F == -1)
    {
        printf("queue is empty");
        exit(1);
    }
}
```

Is Empty
O(1)

```
else
{
    y = Q[F];
    if (F == R) } Del deletion
    F = R - 1;
```

```
else
    F++; } 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 because an element because R can not go back. So Linear Queue is not efficient implementation of queue.

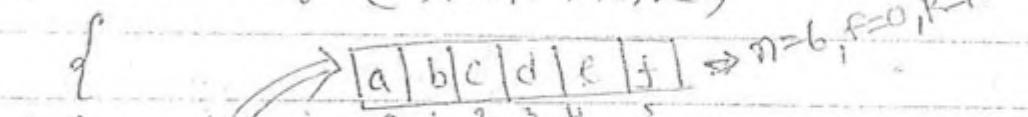
So we are going for circular Queue in which R can go back also.

Circular Queue

In circular queue: $(R < F)$

Circular Queue Insertion

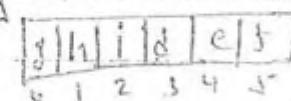
Circular Queue(Q, N, F, R, X)



$\{ \text{if } ((R=N-1 \& F=0) \text{ || } (R=F-1))$

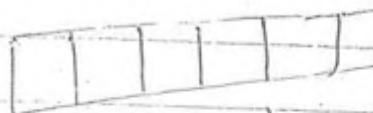
 $N=6$

$\} \text{Pointf ("queue is full")}$
exit();

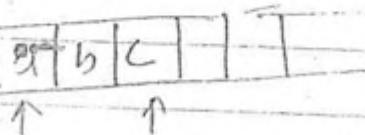
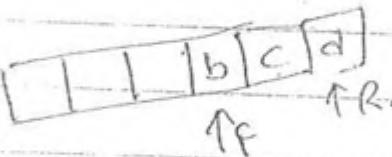
 $F=3$
 $R=2$

else

{

if ($R == -1$) $F = R = 0;$  $F = R = -1$

else

if ($R == N-1$) $R = 0$ else $R++$ { }
}

Circular Queue Deletion

Cdeletion(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

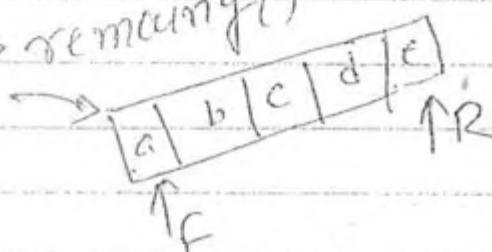
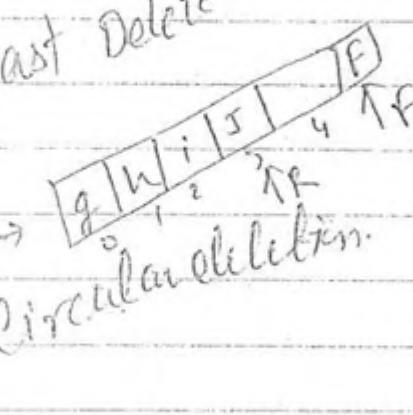
if ($f == n-1$)
 $f = 0$

else

$f++$

\Rightarrow remaining()

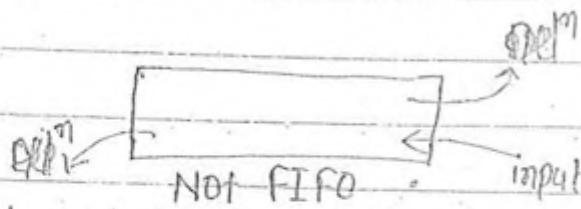
return (g);



Types of Queue

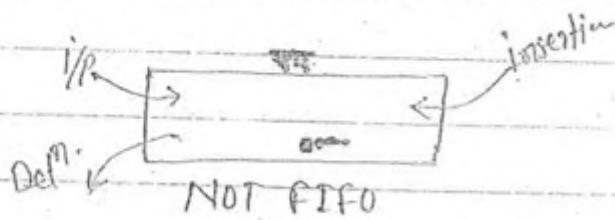
- (1) Input restricted Queue.
- (2) Output restricted Queue.
- (3) Double ended Queue.
- (4) Ascending priority queue.
- (5) Descending priority queue.

✓ Input Restricted Queue :-

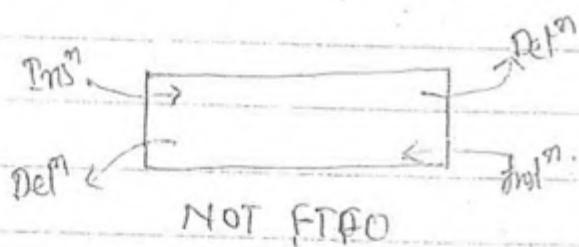


Input will be done only one side but O/P will be done both sides.

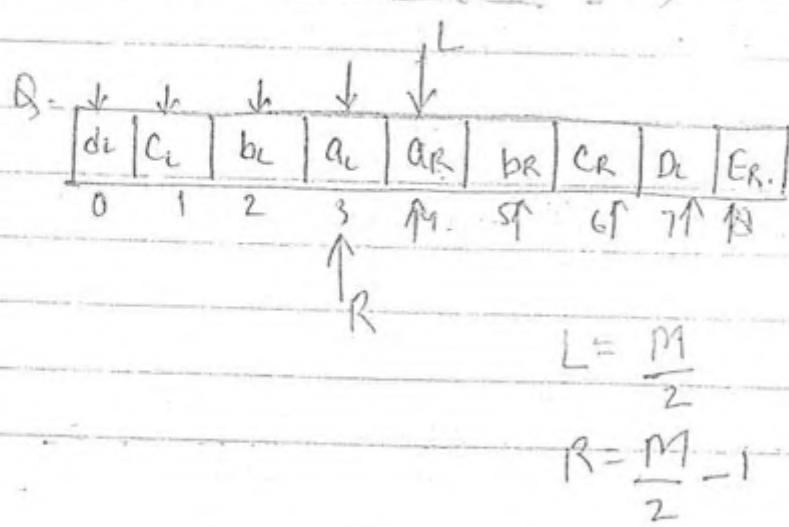
✗ Output Restricted Queue :-



✓ Double Ended Queue (DEQ) \Rightarrow It is data structure.



Double Ended Queue (Operation)



Insert leftif ($l == 0$) full

else

```
{
    L--
    Q[L] = u
}
```

Deletion leftif ($l == \frac{M}{2}$) empty

else

```
{
    Y = Q[L]
    L++
}
return(Y);
```

Insert rightif ($R == M - 1$) full

else

```
{
    R++
    Q[R] = u
}
```

Deletion rightif ($R == \frac{M}{2} - 1$) empty

else

```
{
    Y = Q[R]
    R--
}
return(Y);
```

Efficient way of implementing Double Ended Queue

 $M = 9$ $L = 1 \quad L \quad L \quad L$

q_L	b_L	c_L	d_L	e_L	d_R	c_R	b_R	q_R
0	1	2	3	4	5	6	7	8

 $R = M$

DELTA	Page No.	
	Date:	

Insert Leftif ($R == L+1$) full)

else

{
 $L++$ $Q[L] = u;$

3

 \dots Insert Rightif ($R == L+1$) full)

else

{
 $R--$ $Q[R] = u;$

3

Delete Leftif ($L == -1$) empty

else

{

 $y = Q[L];$ $L--;$

3

return(y);

Delete Rightif ($R == m$) empty

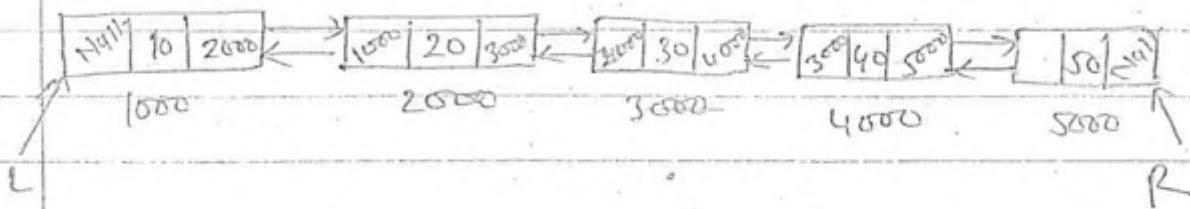
else

{

 $y = Q[R];$ $R++;$

3

return(y);

Delete Left① if ($L == \text{NULL}$) empty② if ($L == R$) free(L) $L = R = \text{NULL}$ ③ $L = L - \&p$ free($L \rightarrow L_p$) $L = L_p = \text{NULL}$ Delete Right① if ($R == \text{NULL}$) empty② if ($L == R$) free(L) $L = R = \text{NULL}$ ③ $R = R + \&p$ free($R \rightarrow R_p$) $R = R_p = \text{NULL}$ Insert Left $P \rightarrow \&p = L$ $L = L_p = P$ $L = P$ $P = \text{NULL}$ Insert Right $R \rightarrow \&p = P ;$ $P \rightarrow L_p = R ;$ $R = P ;$ $P = \text{NULL} ;$

Priority Queue

Insertion and Deletion can be done at any place depending upon Priority.

Ascending Priority:- We always delete min^m element each tim.

50	45	15	10	25	60
50	45	15	10	25	60

↓
Ascending Priority Queue

10, 15, 25, 45, 50, 60
(min).

Descending Priority:- We always delete max^m element each tim.

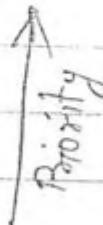
50	45	15	10	25	60
50	45	15	10	25	60

H

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

Insertion $O(1)$ Deletionfind min in the given array of n -element and deletefinding: \downarrow min $\approx O(n) + O(n) \approx$ $O(n)$

sifting

Sorted Array.

60, 50, 90, 5, 25, 30, 1, 15, 90

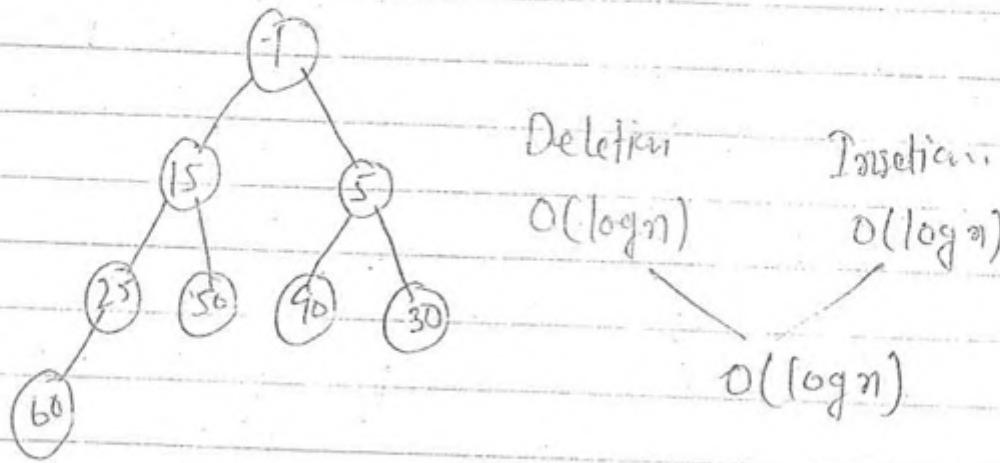
1	5	15	25	30	50	60	70	90
0	1	2	3	4	5	6	7	8

Insertion $O(n)$ Deletion $O(1)$ [using marker] $O(n) + O(1)$ \downarrow $O(n)$

Priority

Min heap

60, 50, 90, 5, 25, 30, 1, 15



B⁺-Tree

Insertion $\Rightarrow \underline{O(\log n)}$

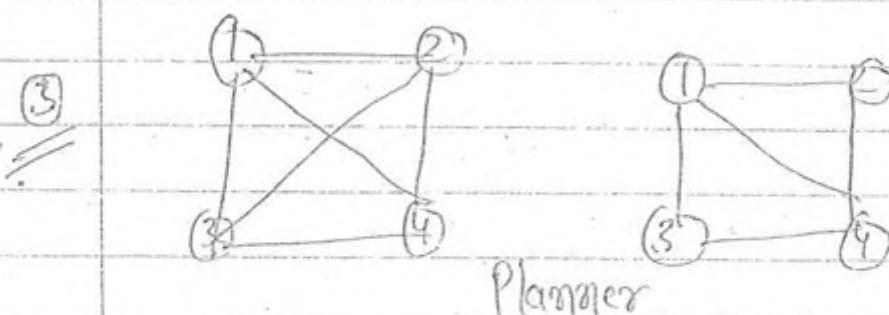
Deletion $\Rightarrow \underline{\log(n)}$
 $\underline{O(\log n)}$

2nd Time

Insertion $\underline{O(\log n)}$

Deletion = $O(1)$

$\log n$

Work BookAug 4
10Aux₄ =

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & & \\ q_{31} & q_{32} & & \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}$$

300.

11

$$A_{24 \times 4} = \begin{bmatrix} q_{11} & 0 & 0 & 0 \\ q_{21} & q_{22} & 0 & 0 \\ q_{31} & q_{32} & q_{33} & 0 \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}$$

$$B_{24 \times 4} = \begin{bmatrix} b_{11} & 0 & 0 & 0 \\ b_{21} & b_{22} & 0 & 0 \\ b_{31} & b_{32} & b_{33} & 0 \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

$$B^T = \begin{bmatrix} b_{11} & b_{21} & b_{31} & b_{41} \\ 0 & b_{22} & b_{32} & b_{42} \\ 0 & 0 & b_{33} & b_{43} \\ 0 & 0 & 0 & b_{44} \end{bmatrix}$$

$$C = \begin{bmatrix} a_{11} & b_{11} & b_{21} & b_{31} & b_{41} \\ a_{21} & a_{22} & b_{22} & b_{32} & b_{42} \\ a_{31} & a_{32} & a_{33} & b_{33} & b_{43} \\ a_{41} & a_{42} & a_{43} & a_{44} & b_{44} \end{bmatrix}$$

$$b[3,1] = c[1,4]$$

i, 5 j; i+1

(11)

i=4, 5, 6

$$a[4+1] = a[5] \quad \text{side effect}$$

$$a[6] = 6$$

(12)

$$n = 1000 \Rightarrow 10 \text{ ms}$$

$$1000 \times \log_{10} 1000 C \Rightarrow 10 \text{ ms}$$

$$3000 C = 10 \text{ ms}$$

$$C = \frac{10}{3000} \text{ ms}$$

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) $\hookrightarrow 2^{\Sigma^*}$ contains every possible language, (Regular, CFL, CSL...)
but, it is Uncountable.

4) $\hookrightarrow L_{RE} \xrightarrow{\text{means}}$ Set of all RE language and It is "Countable Infinite".

5) \hookrightarrow Set of all Turing Mc is Countable Infinite and discrete.

for ex:-

$$Q = \{q_0, q_1, q_2\}$$

$$\Gamma = \{a, b, \square\}$$

$$\{L, R\}$$

6) LRE, Lfg, Lsg, LREC all are Countable Infinite.

(CANTOR 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.

Which one Countable Infinite and Uncountable Infinite.

~~only,~~ ~~or~~ 2^{Σ^*} and LnotRE 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 RE and REC Languages :-

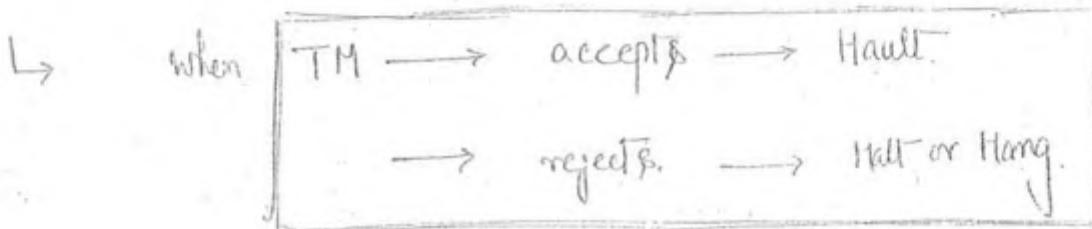
RE
 ↳ defn → A language is RE if and only if it accepts or. There

↳ exist a Turning machine.

↳ It is also called as semi-decidable.

REC
 ↳ defn → A language is REC if and only if there exist a
 i.e. (TM)
 Turning machine, and which halts on every
 string 'w' where, $w \in \Sigma^*$.

↳ It is also called as decidable.

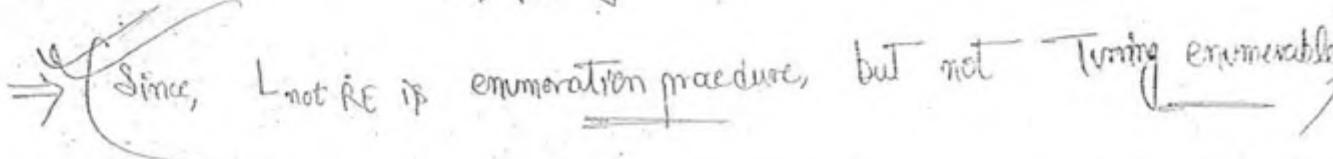


Limited Version of RE is REC.

- 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

} all are Turing enumerable,


 Since, L not 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.

→ If L is RE, then L^c may or may not be RE, because of closure property.

→ If L is REC, then L^c is REC.

\hookrightarrow If L and L^c are complementary language,

Case I: Both L and L^c is REC i.e. not RE

Case II: Both L and L^c

L is RE and not REC, $\therefore L^c$ 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 Undecidability.

L_1 is RE

L_2 is RE

Is $L_1 \cup L_2$ is RE? \Rightarrow decidable

RE language \rightarrow Regular ?

\hookrightarrow Using RICE'S theorem, above one is Undecidable.

Such question is nonTrivial question.

Non Trivial \rightarrow Undecidable.

Trivial Question } \rightarrow decidable.

$L_1 \rightarrow$ which RE \rightarrow Regular ?

$L_2 \rightarrow$ 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) Hailing Problem (HP)
- 2) Blank Tape Hailing 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 $P_1 \leq P_2$~~
 $P_1(D) \Rightarrow P_2(D)$
 $P_2(D) \Rightarrow P_1(D)$

HP \leq BTHP
HP \leq SEP
HP \leq PCP
RE M \leq MPCP (UD) \leq (UD)

4) PCP :- A PCP Solution is Undecidable,
 when, $(u_i, v_j, u_k, \dots) = (v_i, v_j, v_k, \dots)$

~~But~~ Undecidable or Semi-decidable,
 ↳ decidable for only Unary alphabet $\{1\} = \Sigma$

$G_1 = G_2$ i.e. equivalent of grammar is undecidable,

$$\text{for eg:- } \begin{pmatrix} u_1 & u_2 & u_3 \\ 10, 00, & 110 \end{pmatrix} \rightarrow \text{Set 1}$$

PCP solution exist?

$$U_2 \rightarrow \infty \quad \begin{matrix} U_1 \\ 10 \end{matrix} \quad \left\{ \text{Therefore} \quad \begin{matrix} U_2 & U_3 \\ 0 & 10 \end{matrix} \right. \quad \begin{matrix} 0.0 & 10 \\ V_2 & V_3 \end{matrix} \quad \left. \downarrow \right.$$

It means there exist
a PEP solution

Thus, ϵ -undecidable,

MPCP: - A solution is said to be MPCP,

$$u_i \cdot u_j \cdot u_k = v_i \cdot v_j \cdot v_k$$

e.g.:-(RE but not REC) universal language. binary code

Q A language is RE but not REC $\Rightarrow L_u = \{ T \mid \overline{e(T)} \in L(T) \}$

\Leftarrow : A language is not RE $\Rightarrow \exists T \in L_d = \{T \mid e(T) \notin L(T)\}$

eg of { diagonalisation
language

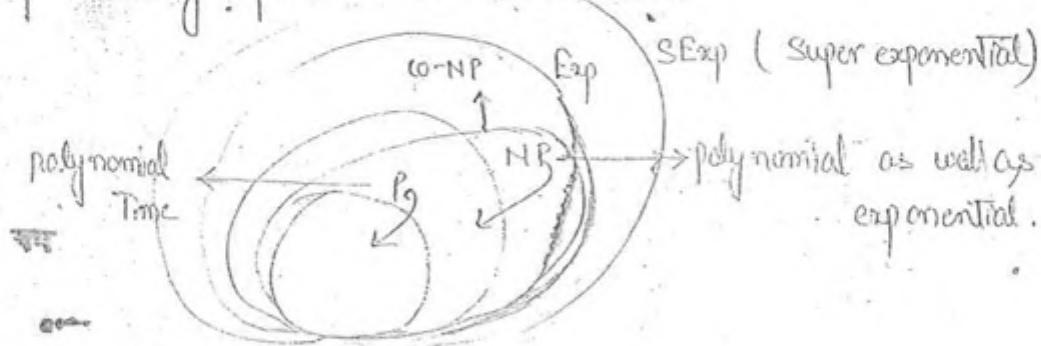
↳ decidable for only unary alphabet, i.e. $\Sigma = \{1\}$

Computational Complexity

↳ Decidable

→ In, this topic only Speed matter.

↳ A Halting problem is Undecidable.



$$\Rightarrow P \subseteq NP$$

$\Rightarrow P \subseteq NP$? → It is not sure | prove till now

$P = NP$? → 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, $L' \in \text{NP}$

$$\& L' \leq_p L$$

→ 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. $NP \in P$, then $P = NP$.

Closure property

↳ 'P' is closed under ($U, \cap, L^c, \cdot, *$)

↳ 'NP' is closed under ($U, \cap, \cdot, *$)

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 $NP = Co-NP = P \rightarrow \text{True}$

$P \subseteq NP \cap Co-NP \rightarrow \text{True}$

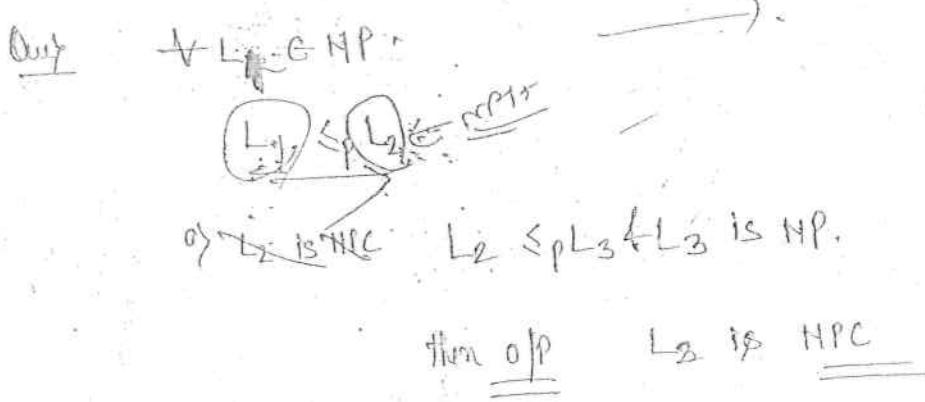
If $L_1 \leq_p L_2 \& L_2 \leq_p L_3 \Rightarrow L_1 \leq_p L_3$

~~↳ 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 :-

Bounded Tiling

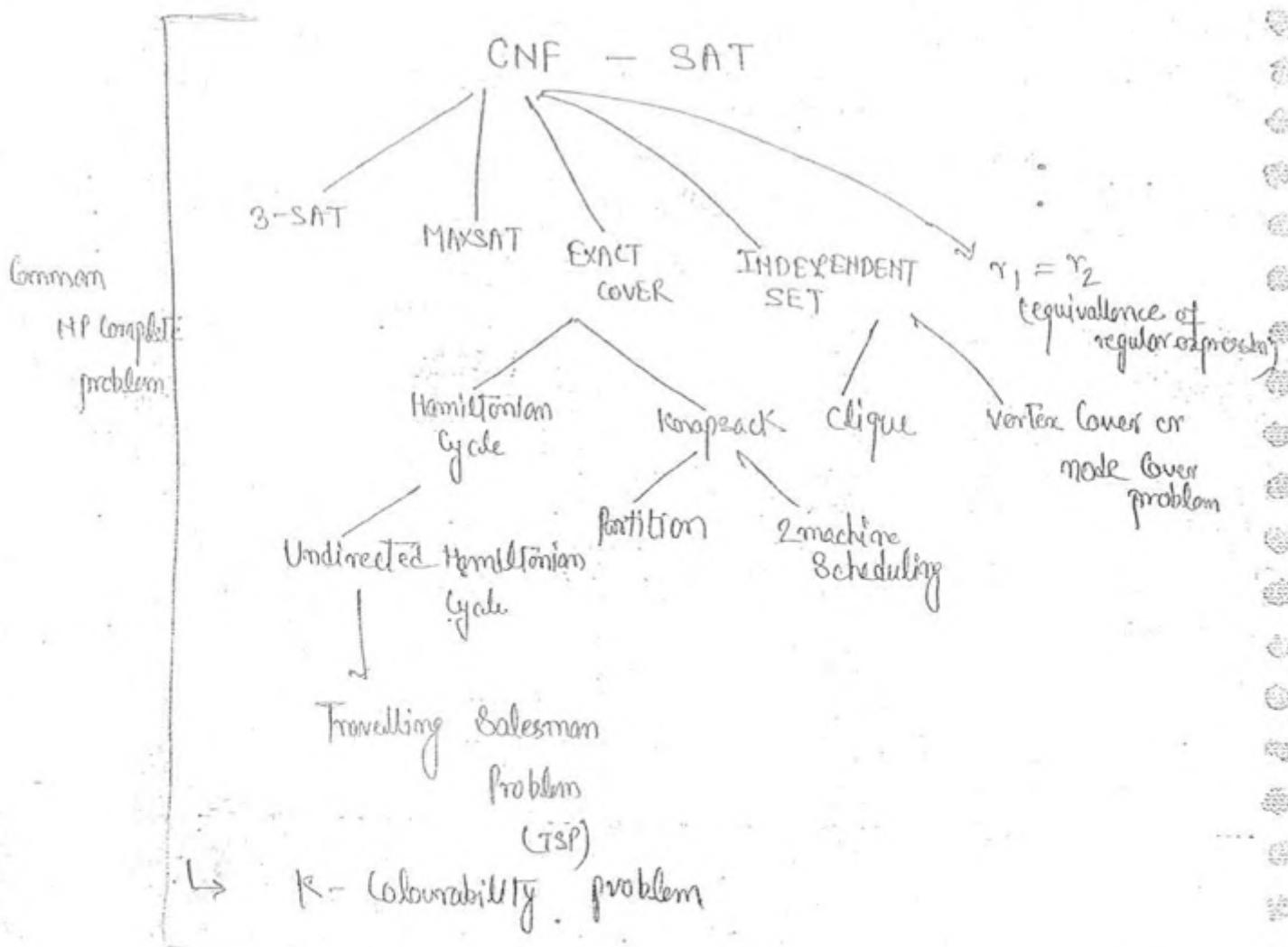
CNF-SAT | satisfy Conjuctive normal form

[Tautology + Contradiction \Rightarrow SAT (satisfy)]

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. $M_1 = M_2$.

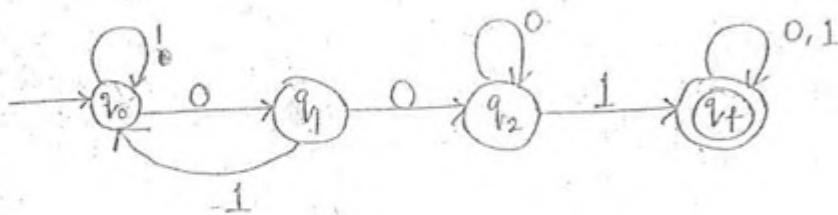
and $\eta_1 = \eta_2$.

↳ MAXCUT.

THE END

Double

i) String that "containing the substring 001".



$$R.E. \Rightarrow (1 + \underline{001} + 001)^*$$

(True or False)

ii) By Using Arden's Theorem,

$$R = Q + R^P \Rightarrow R = Q \underline{P^*}$$

$$q_0 = q_0 1 + q_1 1 + \lambda \quad \text{--- (1)}$$

$$q_1 = q_0 0 \quad \text{--- (2)}$$

$$q_2 = q_1 0 + q_2 0 \quad \text{--- (3)}$$

$$q_f = q_2 1 + q_f (0+1) \quad \text{--- (4)}$$

Sol} from eq (4)

$$q_f = q_2 1 + q_f (0+1)$$

$$R = Q + R^P \Rightarrow R = Q \underline{P^*}$$

$$\boxed{q_f \geq q_2 1 (0+1)^*}$$

from eq (3)

$$q_2 = q_1 0 + q_2 0$$

$$\rightarrow \boxed{q_2 = q_0 0^*}$$

$$\left\{ \begin{array}{l} q_f = q_1 0 0^* 1 (0+1)^* \\ q_0 = q_0 1 + q_1 1 + \lambda \end{array} \right.$$

$$\left\{ \begin{array}{l} q_f = q_1 0 0^* 1 (0+1)^* \\ q_0 = q_0 1 + q_1 1 + \lambda \\ \text{Since, } q_1 = q_0 0 \end{array} \right.$$

Application of Nfa & Dfa.

Date:- 13 Sep. 2010

- 1) Lexical Analysis { 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 M/c.

↳ Sequential Circuit design.

or

Digital Circuit design.

Variation of dfa | mfa.

$$\delta(q_0, a) = \{ (q_1, b), (q_2, a) \}$$

L → left
R → Right

1) $2 \text{ dfa, nfa} \xrightarrow{\text{(Variation of dfa, nfa)}} \text{dfa, nfa + R or L}$

2) $\text{dfa, nfa + R} \leftrightarrow L + R/w \equiv TM$.

3) $\text{dfa, nfa + stack} \rightarrow PDA$

$\hookrightarrow nfa + stack \xrightarrow{\text{(equivalent)}} NPDA$.

$\hookrightarrow dfa + stack \xrightarrow{\text{(equivalent)}} DPDA$.

4) $\text{dfa} / \text{nfa} + 2 \text{ stack} \xrightarrow{\text{(equivalent)}} TM$

5) $\text{dfa} / \text{nfa} + 2 \text{ counter} \equiv TM$

Counter \rightarrow "It is nothing but a stack.

i.e. : $N = \{ z, \}^*$ stack symbol
 \downarrow counter

6) $\boxed{\text{dfa/nfa} + 1 \text{ counter}} < \boxed{\text{dfa/nfa} + 1 \text{ stack}}$.
 \downarrow less powerful. \downarrow more powerful.

(nfa + 1 counter is better than nfa.)



while

$nfa + 1 \text{ counter} < \boxed{(nfa + 1 \text{ stack}) \rightarrow CFL}$

$nfa + 2 \text{ counter} \equiv nfa + 2 \text{ stack}$

~~nfa + 1 stack~~

Note:-

nfa & dfa < 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, $CFG \rightarrow L$
 $L \rightarrow CFL$
- 2} Derivation, L.M. Derivation & R.M. Derivation.
- 3} Derivation Tree, Ambiguity of Grammar & Languages, Partial derivation tree, Sentential form,
- 4} PARSING: Membership algorithm,
Complexity, Programming language & TOC, $O(n)$
Linear time Compiler, S grammar, LL grammar,
LR grammar.
- 5} Algorithm of CFG's, 1) algorithm, \Rightarrow Removal of λ -prod
2) Removal of unit production;
3) " " useless production;
4) " " LR \Rightarrow Remove ambiguity.
5) " " L factoring \Rightarrow Remove ambiguity.

6) $\text{cfg} \rightarrow \text{cnf}$

7) $\text{cfg} \rightarrow \text{gnf} \rightarrow \text{pda.}$

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 e.g:-

$a^n b^n : n \geq 0$

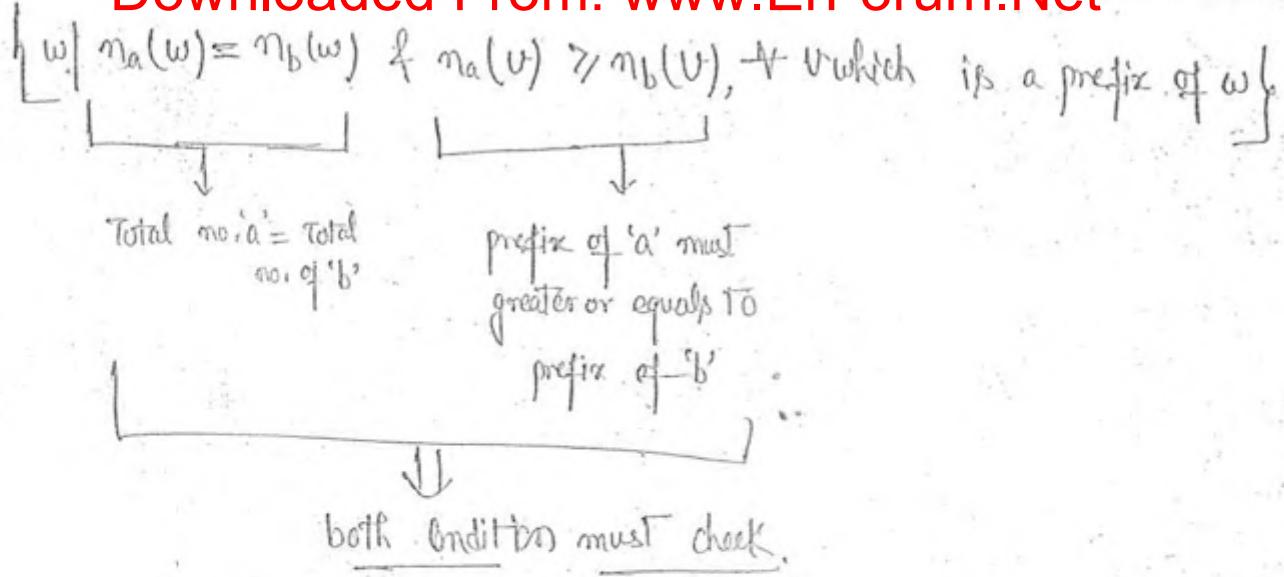
\Rightarrow Grammar: $S \rightarrow aSb \mid \lambda$
↳ very restricted in nature

$w | m(w) = n_b(w)$
called balanced parenthesis

$S \rightarrow \underline{aSb} \mid bSa \mid SS \mid \lambda$

$\underline{S} \rightarrow \underline{aSb} \mid SS \mid \lambda$
properly balanced parenthesis

$w w^R$
 $\rightarrow w(a tb) w^R$
odd palindrome
 $S \rightarrow aSa \mid bSb \mid a \mid b$
even palindrome
 $S \rightarrow aSa \mid bSb \mid \lambda$



~~Note:~~ (properly balanced ~~w~~ 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.

$$V \rightarrow T^* V + V T^* + \overbrace{T^* V T^*}^{\text{Cfl}}$$

→ $w w^R$ is always in the form of ~~w~~ a palindrome;

$$L = \{ww^R \mid w \in (a;b)^*\} \quad \rightarrow \text{Even palindrome}$$

$$L_2 = \left\{ w(a+b)w^R \mid w \in (a,b)^* \right\} \rightarrow \text{odd palindrome.}$$

$$L_3 = \{ w \mid w = w^R \} \rightarrow \text{All palindrome.}$$

(even + odd)

Thus,

$$L_3 = L_1 + L_2 \text{ or } L_1 \cup L_2.$$

四
四

$$a^m b^m ; \quad m > 1$$

Gramm. :- $s \rightarrow \text{asb} \mid \text{ab}$

294

$$a^n b^{2n}, \quad n \geq 1$$

Girren :-

$\hookrightarrow \text{asbb} \setminus \lambda$

eg. \tilde{w}_j

$$a^m b^n, \quad m > 1$$

Grammar

Grammar $S \Rightarrow aasb \mid \lambda$

09.12.15

$$h(a^m b^n); \quad m > n$$

$$A_n = 5n + 1$$

$$\{ a^m b^{5n+1} \mid m \geq 0 \}$$

We always take
Condition by
default in m...
 ≥ 0
if not given.

for e.g:-

$$a^n b^m ; n > m ; n, m \geq 0$$

$$S \rightarrow AS_1$$

$$S_1 \rightarrow aS_1 b | \lambda$$

$$A \rightarrow aA | \lambda \rightarrow a^*$$

$S \rightarrow a^* a^n b^n \rightarrow$ This is nothing but,

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

$$S \rightarrow AS_1 | S_1 B$$

$$S_1 \rightarrow aS_1 b | \lambda$$

$$A \rightarrow aA | a \quad X \quad X \quad a$$

$$B \rightarrow bB | b \quad X \quad b \quad \lambda$$

Condition to check

Sol

$$S \rightarrow AS_1 \rightarrow A a S_1 b$$

$$\rightarrow a A a S_1 b$$

$$\rightarrow \underline{\underline{aaaab}}$$

$$\rightarrow \underline{\underline{a^n b^m}}$$

Thus, $[a^n b^m]$

- 1) $n > m$ or $n < m$
- 2) $n \neq m$

Ques. $S \rightarrow AS_1 \rightarrow n-m = 2k$

$S_1 \rightarrow aS_1 b \mid \lambda \rightarrow n=m$

$A \rightarrow aaA \mid aa$

Soln

$S \rightarrow AS_1 \rightarrow aaAS_1 \rightarrow AS_1$

$\rightarrow A aS_1 b \rightarrow A aS_1 b b \rightarrow A a a S_1 b b$

$\rightarrow \underline{aaaab} \underline{b}$

Final O/P $\Rightarrow (a^n b^m; n-m=2k); k \geq 1$

Ques. $n_a(w) = 2n_b(w)$

$S \rightarrow aasb \mid asbb \mid ss \mid \lambda$

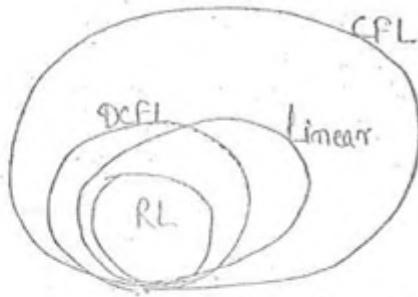
Possible
Combination
or order

$S \rightarrow \overbrace{aS}^s \overbrace{aS}^s b$
asbsa
bSas a | λ

for eg:- $[w \# w^R] \quad w \in \{a, b\}^* \quad \# \notin \{a, b\}^*$

Grammar

$S \rightarrow aSa \mid bSb \mid \#$



~~✓~~ from the above fig, every DCFL and Linear grammar contains Regular grammar and all three are under Context free language.

Linear grammar

$$N \rightarrow T^* + VT^* + T^*V + T^*VT^*$$

- ↳ Every DPDA can do linear comparison.
- ↳ Push and Pop is not possible in DPDA.
- ↳ while, Push and Pop is possible in NPDA.

for eg:- $\frac{aab}{\text{push}} \frac{baa}{\text{pop}}$.

$\frac{a}{\text{push}} \rightarrow a$
 $\frac{b}{\text{push}} \rightarrow b$
 $\frac{a}{\text{pop}} \rightarrow b$
 $\frac{b}{\text{pop}} \rightarrow a$

$\left[\begin{array}{l} \text{If } a \rightarrow a \text{ comes push} \\ \text{If } a \rightarrow b \text{ comes pop} \end{array} \right]$

$\Rightarrow S \rightarrow aSb \mid \lambda \Rightarrow$ Linear grammar.

$\Rightarrow S \rightarrow aSb \mid bSa \mid SS \mid \lambda \Rightarrow$ Non linear grammar.
but
DCFL.

e.g:- if wwR .

$S \rightarrow aSb \mid bSb \mid \lambda$ → It is CFL but not DCFL.

ii) $w \mid m_a(w) = m_b(w) \rightarrow$ It is not linear, but it is DCFL.

iii) $a^n b^m ; n > 0 \rightarrow$ 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.

$S \rightarrow aSb \mid aSbb \mid aSbbb \mid \lambda$

$\rightarrow S \rightarrow aSb \rightarrow aaSbb \rightarrow \underline{\underline{aabb}}$

$\rightarrow S \rightarrow aSb \rightarrow aasbbb \rightarrow \underline{\underline{aabbb}}$

Ques. find a language correspond to the grammar.

$$G \Rightarrow \left\{ \begin{array}{l} S \rightarrow aaB \\ A \rightarrow aaBa \\ B \rightarrow bbAa \\ A \rightarrow \lambda \end{array} \right.$$

$L = ?$

$$\rightarrow S \rightarrow aaB \rightarrow aabbAa \rightarrow aabbaaBa$$

$$\begin{array}{ll} \rightarrow aabb \\ \rightarrow aabbba \\ \rightarrow aabbaba \\ \rightarrow aabbbaabbAa \\ \rightarrow aabbbaabbbaBa \\ \rightarrow aabbbaabbbaabbBa \end{array} \rightarrow \begin{array}{l} aabbaabbAaa \\ aabb \boxed{aa} \boxed{bb} \boxed{aa} \end{array}$$

$$S \rightarrow xSy \mid z$$

$$\rightarrow x^n z y^n \quad [S \xrightarrow{n} \text{remove by } z]$$

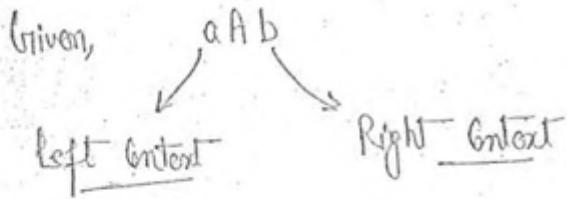
$$S \rightarrow ab \underbrace{(bbba)}_P^n \underbrace{ba}_Q \underbrace{(ba)}_R^m ; n \geq 0$$

\rightarrow Taking a common

$$\therefore \rightarrow B \rightarrow (bbba)^n bba (ba)^m$$

$$(PQ)^n P = P \cdot (QP)^n$$

$$\Rightarrow \{ abbb (aabb)^n (ab)^m a ; n \geq 0 \}$$



→ If the grammar is Context Sensitive, then we can write

The grammar as :-

$$xAy \rightarrow xA'y$$

for eg:- i) $aAb \rightarrow aAab$: $A \rightarrow Aa$.

ii) $bAc \rightarrow bBc$.

$$\text{iii) } aA \xrightarrow{\text{Left Context}} xA^y \xrightarrow{\text{Right Context}} xAa$$

$$\text{iv) } aA \rightarrow xA^y$$

Note:-

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. $S \rightarrow A B \quad (aa^* b^*) \quad S \rightarrow AB$

$$\begin{array}{l} A \rightarrow aaA \quad | \quad \lambda \rightarrow aa^* \\ B \rightarrow bbB \quad | \quad \lambda \rightarrow b^* \end{array} \quad \begin{array}{l} \rightarrow aaAB \\ \rightarrow aaA bb \\ \rightarrow \underline{aab} \end{array}$$

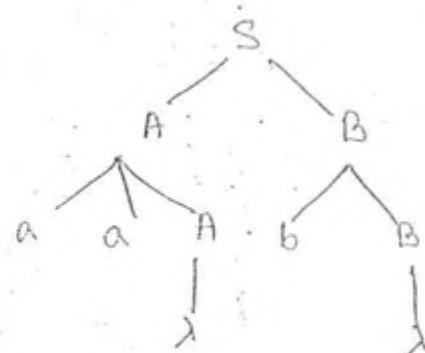
$aab \in L(G)$

$$\underline{\text{L.M.D.}} \Rightarrow S \Rightarrow A B \xrightarrow{\text{(left most)}} aaAB \rightarrow aaB \rightarrow aabB \rightarrow \underline{aab}$$

$$\underline{\text{R.M.D.}} \Rightarrow S \Rightarrow AB \xrightarrow{\text{(right most)}} abb \rightarrow Ab \Rightarrow oAb \rightarrow \underline{aab}$$

Derivation $\Rightarrow S \Rightarrow AB \rightarrow aaAB \rightarrow aaA bb \rightarrow aabB \rightarrow \underline{aab}$
 (neither LMD nor RMD)

(Derivation Tree)



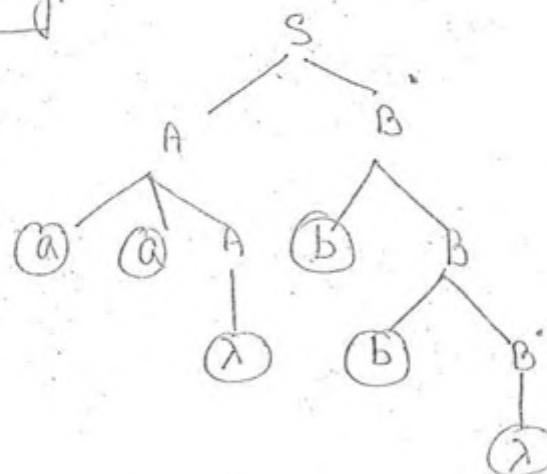
↳ A derivation tree must always be unique.

↳ 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 $\Rightarrow a a \underline{b} b \lambda$

Notes

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)

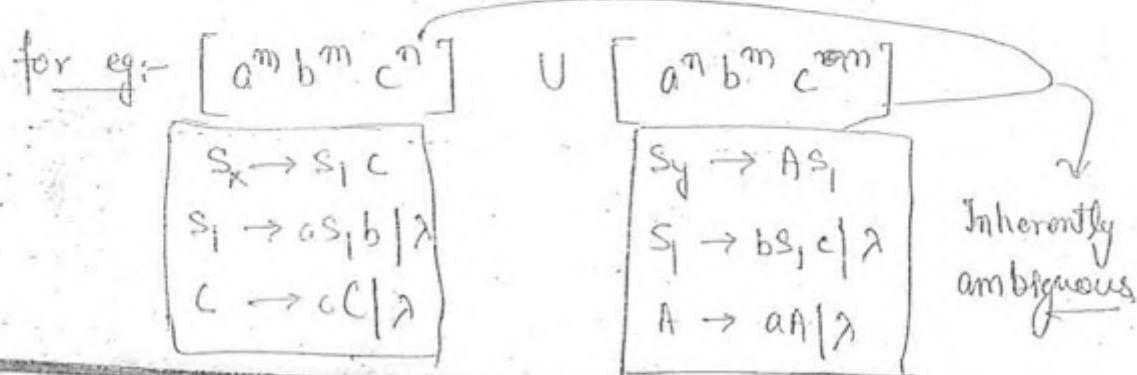
↳ While Root of partial derivation tree can be any one in the tree, not necessary of 'S' (Root node)
i.e. (Root must be anywhere in the tree)

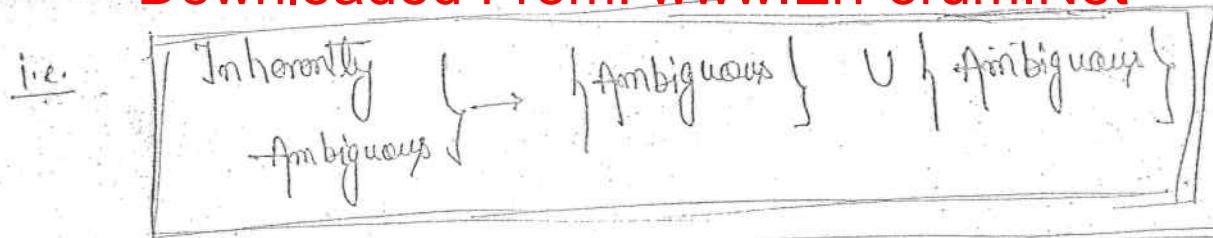
- ↳ A grammar ' G ' is unambiguous if and only if
 - $\forall w \in L(G)$, w must have exactly one derivation tree,

- ↳ 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 language ' L ' is inherently ambiguous if and only if every grammar that generates ' L ' is ambiguous.





~~for eg:-~~

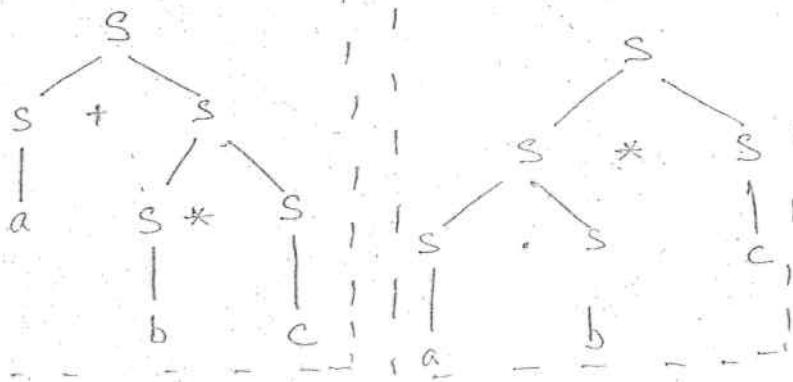
Date:- 14/09/10.

→ Ambiguous grammar :-

$$\text{eg:- } S \rightarrow S+S \mid S*S \mid a \mid b \mid c.$$

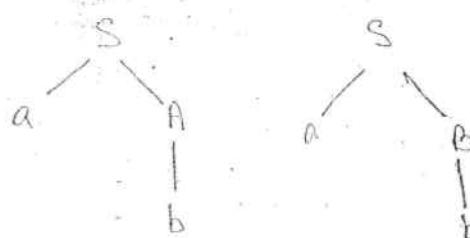
$a+(b*c)$

$(a+b)*c$

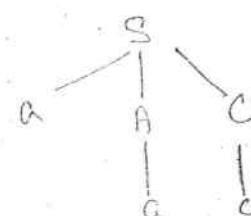
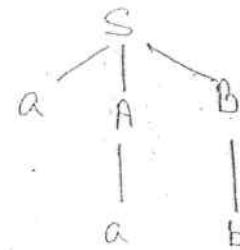


→ Left factoring Ambiguity :-

$$\text{eg:- } S \rightarrow aA \mid aB \quad \text{or, } S \rightarrow aAB \mid aAC \\ A \rightarrow b \quad \text{or, } S \rightarrow Aa \mid Ab. \\ B \rightarrow b$$



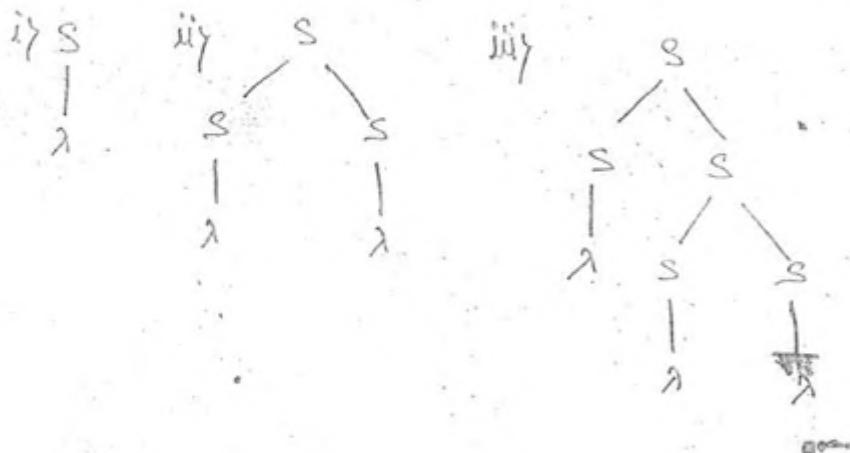
$$\begin{array}{l} A \rightarrow a \\ B \rightarrow b \\ C \rightarrow c \end{array}$$



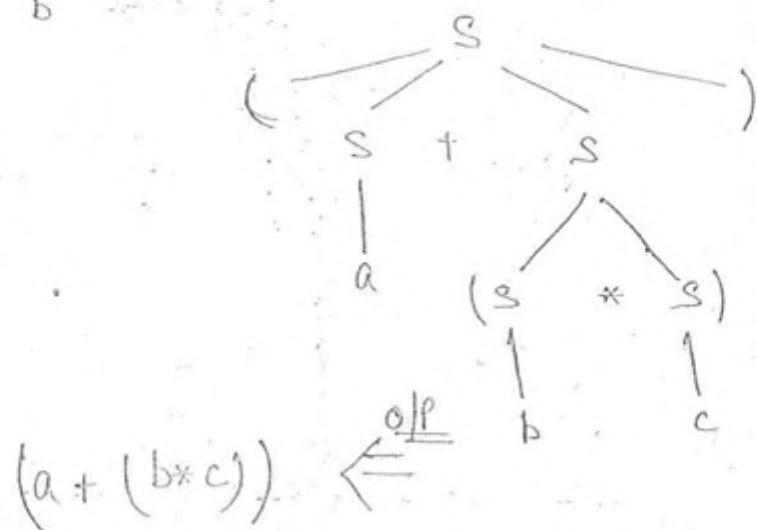
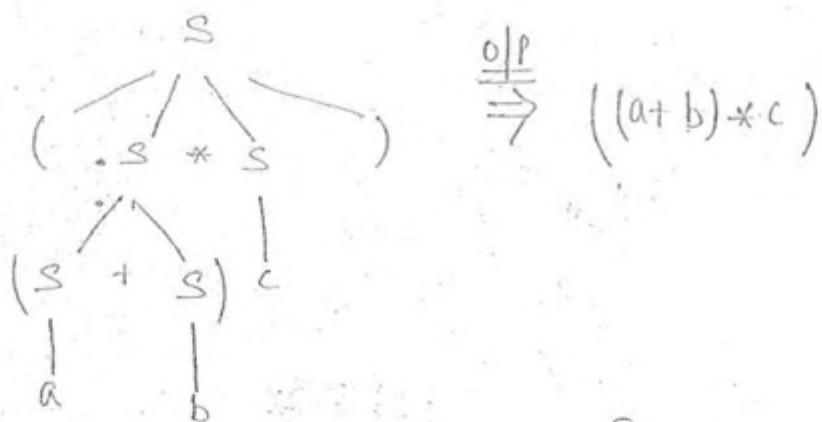
→ If two SS is given, then there is more chance of Ambiguity.

for eg:- $S \rightarrow a \mid b \mid SS \mid \lambda$ O/P \Rightarrow Ambiguous

derivation Tree



eg 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".]

↳ A 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 (lambdai)]

BFP :- Brute force Parsing.

It will only work after removing 'λ' & "Unit" production.

→ Complexity of Brute force Algorithm. is

$$O(K^n) \rightarrow \text{In worst case.}$$

$$\hookrightarrow |P| + |P|^2 + |P|^3 + \dots + |P|^{2n}$$

It mean, it hold $2n$ rounds

Thus, the no. of step required :-

$$\text{no. of steps} = \frac{|P|(|P|^{2n} - 1)}{|P| - 1}$$

↪ (BFP is a membership alga.)

$$\Rightarrow |P|^{2n}$$

↪ It comes under NP problem. $\Rightarrow [|P|^2]^n$

NP problem $\xrightarrow{\text{implies}}$ exponential + polynomial $\Rightarrow K^n$

CYK Algorithm

⇒ The best membership algorithm for CFL is of

Time Complexity = $O(n^3)$

Complexity of UK Algo $\Rightarrow O(n^3)$

proper subset

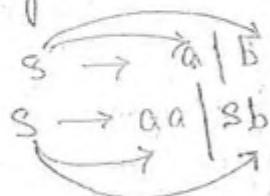
Time complexity of $O(n)$

```

graph TD
    PS[proper subset] --- S[S-grammar]
    S --- DCFL[DCFL]
    DCFL --- CFL[CFL]
    S -- "may" --> LL[LL-grammar]
    LL -- "generate" --> DCFL
    LL -- "may" --> CFL
    LL --- LR[LR-grammar]
    LR -- "exactly generate" --> DCFL
  
```

(S-grammar is the Super Set of
Regular grammar)

Regular grammar \subseteq S-grammar \subseteq DCFL.



$$\text{LHS} = s \rightarrow a \wedge b$$

$$\begin{array}{l} LL(k) = ? \\ LR(k) = ? \end{array} \rightarrow \text{Int} \Rightarrow (\text{Read in compiler subject})$$

→ 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 eg:- $S \rightarrow aSb \mid SS \mid a \mid b$

O.P.:- Neither, LL(K) nor LR(K).

(Backup - NAFR form)

$\langle \text{if-statement} \rangle ::= \text{if} \langle \text{statement} \rangle \text{then} \langle s \rangle$
 $\qquad\qquad\qquad \text{else} \langle s \rangle$

(Type 'E' grammar)

$I \rightarrow aSbScs$

$\langle \text{Term} \rangle ::= \langle \text{Term} \rangle * \langle \text{factor} \rangle$

$$\begin{array}{l} T \rightarrow T * F \\ F \rightarrow F + T \end{array} \quad \left[\begin{array}{l} T \rightarrow \text{Term} \\ F \rightarrow \text{factor} \end{array} \right]$$

↳ 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. $LR(K) \xleftrightarrow{\text{gives}} DCFL$,

Since, every Regular language is a DCFL,

Thus, Every regular language is unambiguous,

But, $Every\ LR(K) \xrightarrow{\text{can't generate}} \text{Regular language}$

\hookrightarrow Some Regular grammar is ambiguous, and some are unambiguous.]

4) for any ~~CFG~~ 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) & LR(1)

for any DCFL $\xrightarrow{\text{with prefix}} DCFL\ LR(0)$
 $\xrightarrow{\text{without prefix}} LR(1)$

7) LR(0) grammar exist only if and only if for DCFL with prefix property.

→ ~~L~~ language ' L ' is said to be prefix property if and only if no proper prefix of $w \in L$ must be in ' L '.

$$L = \{aab, ab\} \\ \{a, aa, \lambda, a, \lambda\} \rightarrow LR(0)$$

$$L = \{aab, \lambda, a\} \rightarrow \text{only } LR(1) \text{ not } LR(0).$$

~~L~~ λ is a proper prefix of any word, ^{or string} except ' λ '

ii). $a^n b^n; n \geq 0 \Rightarrow LR(0)$ not exist,

iii). $a^n b^n; n \geq 1 \Rightarrow LR(0)$ exist,

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.

$$L = \{ @ab\$, @a\$ \} \Rightarrow LR(0)$$

\rightarrow If L is not a DFL then in order to make $LR(0)$
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 DFL, L must produce $LR(0)$ or
 $L\$$ must produce $LR(0)$.

ALGORITHM.

\hookrightarrow 1. Removal of λ (Null) production:

$A \rightarrow \lambda \Rightarrow$ called null (λ) production.

e.g:-

$$\begin{array}{l} A \rightarrow BC \\ B \rightarrow c\lambda \\ C \rightarrow \lambda \end{array} \left\{ \begin{array}{c} A \xrightarrow{*} \lambda \\ \text{Nullable} \end{array} \right.$$

Steps:-

1) $N =$ all Nullable Variable.

2) put all nullable variable to null in
every possible permutation.

for eg:-

$$S \rightarrow ABaC$$

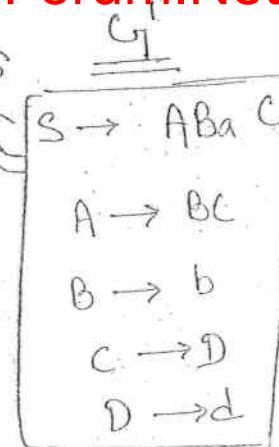
$$A \rightarrow BC$$

$$B \rightarrow b | \lambda$$

$$C \rightarrow D | \lambda$$

$$D \rightarrow d$$

$\Rightarrow G$



Step 1}

$$N = \{B, C\}$$

$$S \rightarrow BaC | AaC | ABa | aC | a$$

Ba | a

Round 2}

$$N = \{B, C, A\}$$

$$A \rightarrow c | B$$

Round 3}

$$N = \{B, C, A\}$$

(After combining we get).

$$S \rightarrow ABaC | BaC | AaC | ABa | aC | Aa | Ba | a$$

$$A \rightarrow BC | C | B$$

$$B \rightarrow b$$

$$c \rightarrow D$$

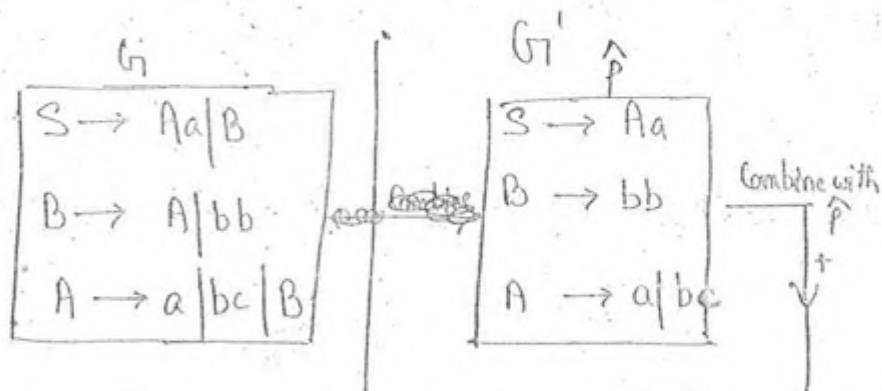
$$D \rightarrow d$$

\Rightarrow final O/P.

2. (Removal of Unit production)

→ Always give first preference to null production then Unit production, then Use less prod.

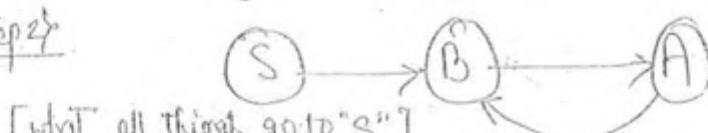
for eg:-



Step 1 make a unit production Dependency graph.

→ make a node for every Variable,

Step 2



[what all things goto "S"]

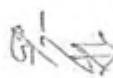
↑ some

$$S \xrightarrow{*} B$$

$$A \xrightarrow{*} B$$

$$B \xrightarrow{*} A$$

$$S \xrightarrow{*} A$$



~~$$S \rightarrow Bb \mid Ba$$~~

$$S \rightarrow bb$$

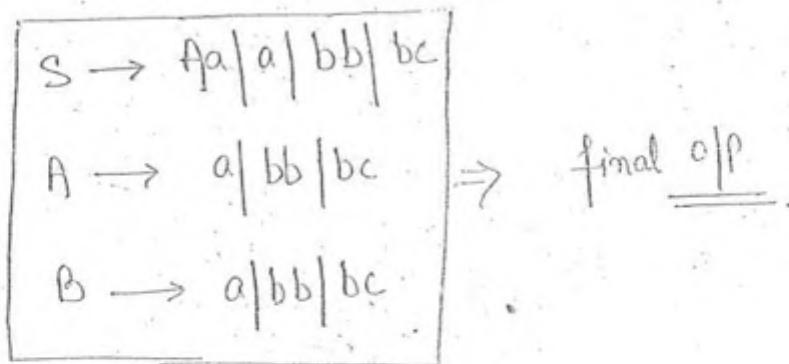
$$S \rightarrow a \mid bc$$

$$A \rightarrow bb$$

$$B \rightarrow a \mid bc$$

This,

$G^1 \Rightarrow$



3) (Removal of useless production)

↳ A production containing useless variable either on left or right called useless production.

\Rightarrow Any problem solve by Algorithm is "decidable"

Useful Variable :- \Rightarrow It must generate Terminal symbol.

for eg:- $S \rightarrow aA|ab$ \Rightarrow If any variable specified, Variable / Terminal must be given in 'S'.

$A \rightarrow a|bb \rightarrow$ useful

Useless Variable \leftarrow $B \rightarrow b$ \Rightarrow (Converse of it is useless Variable)

def.n :- (Usefull V = { X | $s \xrightarrow{*} uxv \xrightarrow{*} w$
Variable})

$uv \in (TUV)^*$;

$w \in L(G)$ }

for e.g:-

$$S \rightarrow aS | A | c$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow acb$$

Step1 We go through with ^{all} Usefull Variable and delete Variable that not generate any Terminal

Step2 Contains Terminal (T^*) and Usefull Variable

1)

$$U = \{ A, B \}$$

$$U = \{ A, B, S \}$$

$$U = \{ A, B \}$$

2)

$$S \rightarrow aS | A$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

Graph



$$U = \{ S, A \}$$

$$\text{Now, } S \rightarrow aS | A$$

$$A \rightarrow a$$

$$\text{Thus, } S \rightarrow aS | a$$

$\rightarrow [aa^*]$ final op

Date:- 15 Sep, 2010

Removal of Left Recurssion.

for eg:- $G_1 = A \rightarrow A\alpha_1 | A\alpha_2 | A\alpha_3 | \dots | A\alpha_m | B_1 | B_2 | B_3 | \dots | B_n$

$$\text{In this step, } G_1^1 = A^1 \xrightarrow{\quad} \alpha_1 A^1 \left| \alpha_2 A^1 \left| \alpha_3 A^1 \right| \dots \right| \alpha_n A^1 \left| \lambda \right.$$

In this we not stop with β , we stop with by ' λ '

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \mid \dots \mid \beta_n A'$$

$$\text{for eg:- } S \rightarrow \frac{S+S}{\alpha_1} \mid \frac{S \times S}{\alpha_2} \mid \frac{a}{\beta_1} \mid \frac{b}{\beta_2} \mid \frac{c}{\beta_3} \quad \left. \right\} \rightarrow G$$

$$G \Rightarrow S \hookrightarrow +SS' \quad | \quad *SS' \quad | \quad \lambda$$

$$S \rightarrow aS' \quad | \quad bS' \quad | \quad cS'$$

Removal of Left factoring

$$A \rightarrow \alpha_x | \alpha_y | \alpha_z$$

$$\therefore \alpha_{\alpha} \in (\text{NUT})^*$$

$$\left. \begin{array}{c} A \rightarrow abc \mid abD \mid aba' \\ \quad \quad \quad \rightarrow aAC \mid aAB \mid aAa' \end{array} \right\}$$

Thus, it can be written as:-

$$A \rightarrow ab A^n$$

$$A'' \rightarrow C|D|A'$$

Algorithm.

CFG \rightarrow CNF {Chomsky Normal form}

λ , UNIT

(Remove) from Set of rules

e.g:- Remove the null, Unit & ~~rules~~ 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,

$$C \rightarrow B \mid \text{Unit production}$$

$$S \rightarrow aA \mid aBB \mid a$$

$$A \rightarrow aaA \mid aa$$

$$B \rightarrow bB \mid bbC$$

$$C \rightarrow bb \mid bbC$$

Step 3} Removal of Useless :-

B & C are Useless,

thus, $S \rightarrow aA \mid a$
 $A \rightarrow aaA \mid aa.$

$a|aa|^*$,

thus, $S \rightarrow aaa(aa)^* \mid a$

final off $[a^{2n+1}; n \geq 0]$

CNF

Thus, CNF of this is:-

let, $D_a \rightarrow a$

This

$S \rightarrow DaA \mid Da$
 $A \rightarrow DaDaA \mid DaDa.$

$D^1 \rightarrow DaA.$

Ques $S \rightarrow ABa$
 $A \rightarrow aa\ b$. $B \rightarrow Ac$ } \Rightarrow Convert it into CNF.

Soln Let $D_a \rightarrow a$.
 $D_b \rightarrow b$
 $D_c \rightarrow c$

Thus, $S \rightarrow ABD_a$
 $A \rightarrow D_a D_a D_b$
 $B \rightarrow AD_c$

$S \rightarrow AD'$ $D' \rightarrow BD_a$ $A \rightarrow D_a D''$ $D'' \rightarrow D_a D_b$ $B \rightarrow AD_c$ $D_a \rightarrow a$ $D_b \rightarrow b$ $D_c \rightarrow c$	\Rightarrow <u>Final off</u> .
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------

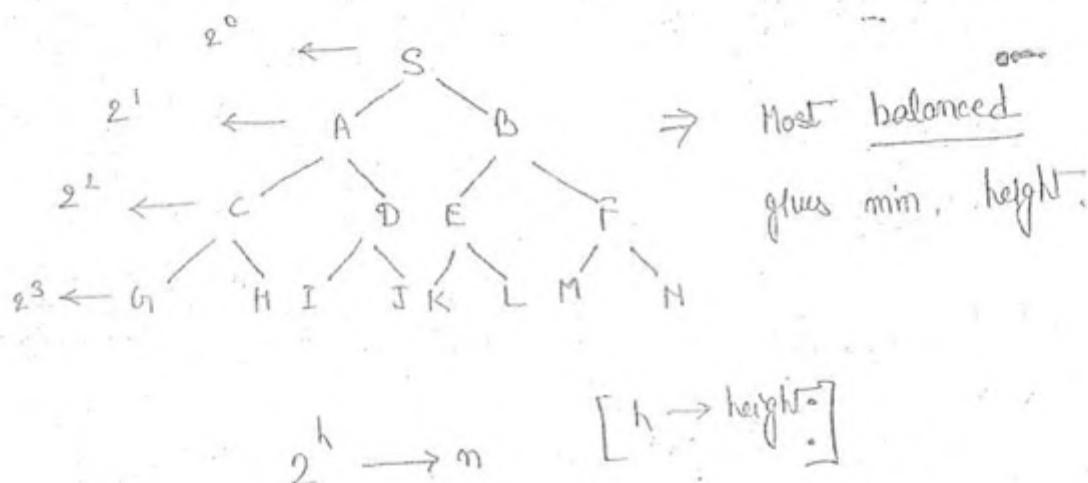
(Theorem of Chomsky Normal form)

Derivation of any string of length n , when G is
 CNF has $(2n-1)$ steps

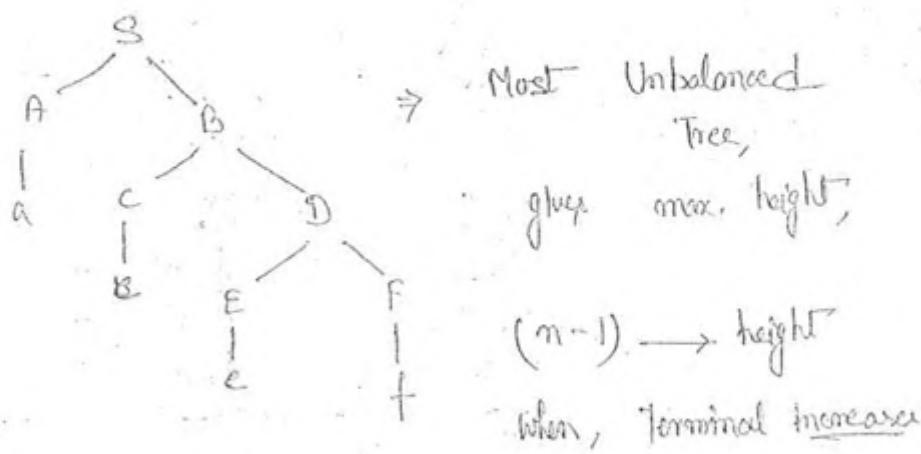
2. Minimum height of derivation tree for a string of length 'n';
 i.e. $|w|=n$, G is CNF is $\lceil \log_2 n \rceil + 1$

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



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) .

- 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:- $S \rightarrow ABA \mid Aa$
 $\boxed{A \rightarrow AB \mid BBC \mid b}$ substituting A

thus $S \rightarrow ABBa \mid BBCBa \mid bBa \mid aba \mid BBCa \mid ba$

$\left[\begin{array}{l} D_a \rightarrow a \\ D_b \rightarrow b \end{array} \right]$ $\rightarrow aBB Da \mid BBC B Da \mid bB D_a \mid aB D_a \mid BBC Da \mid b D_a$.

eg: $S \rightarrow Aa|Bb$ | $S \rightarrow aAbA|aBDA|bDAb|abDa$.
 $A \rightarrow aAb|aB$
 $B \rightarrow b$. | $A \rightarrow aAd|aB$
 $D_a \rightarrow b$; $D_b \rightarrow b$; $D_a \rightarrow a$.

Thus, $S \rightarrow aAD_bD_a | aBD_a | D_bD_b | aBA\underline{Da}$.

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,

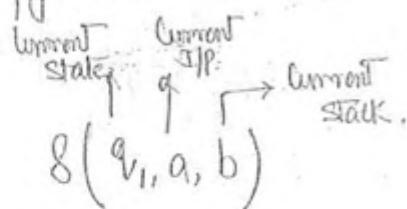
$Z \in \Gamma$ and Z can not push into stack.

$\Sigma = \{a, b\}$

[Instead 'a' there may be '0'
, " or 'b' or " "]

$\Gamma = \{Z, a\}$

\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 $\rightarrow \delta(q_0, a, b) = q_1(a, (q_1, ab))$

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.

$a^n b^n; n > 0$

$$m_a(w) = m_b(w)$$

work

Y

abab|b|b|b|z

push



Here a will pop the b,
& b will pop the a,

$$\delta(q_0, a, z) = (q_1, az)$$

$$\delta(q_1, a, a) = (q_1, aa)$$

$$\delta(q_1, b, a) = (q_2, \lambda)$$

$$\delta(q_2, b, a) = (q_2, \lambda)$$

$$\delta(q_2, \lambda, z) = (q_0, z)$$

$$\delta(q_0, a, z) = (q_1, az)$$

$$\delta(q_0, b, z) = (q_1, bz)$$

$$\delta(q_1, a, a) = (q_1, aa)$$

$$\delta(q_1, a, b) = (q_1, \lambda)$$

$$\delta(q_1, b, a) = (q_1, \lambda)$$

$$\delta(q_1, b, b) = (q_1, bb)$$

$$\delta(q_0, a, z) = (q_0, az)$$

$$\delta(q_0, b, z) = (q_0, bz)$$

$$\delta(q_0, a, a) = (q_0, aa)$$

$$\delta(q_0, b, a) = (q_0, ba)$$

$$\delta(q_0, a, b) = (q_0, ab)$$

$$\delta(q_0, b, b) = (q_0, bb)$$

$$\delta(q_0, \lambda, a) = (q_1, a)$$

$$\delta(q_0, \lambda, b) = (q_1, b)$$

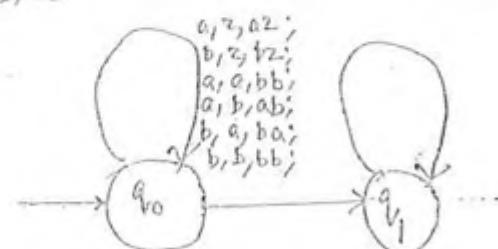
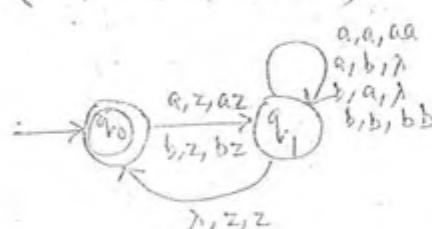
$$\delta(q_1, b, b) = (q_1, \lambda)$$

$$\delta(q_1, a, a) = (q_1, \lambda).$$

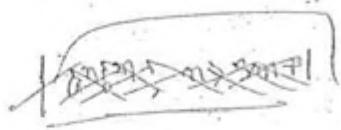
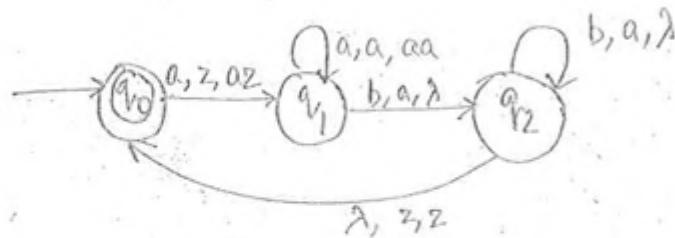
$a^n b^{2n}; n > 0$

$$\delta(q_0, a, z) = (q_1, aa.z)$$

$$\delta(q_1, a, a) = (q_1, aaa.a)$$



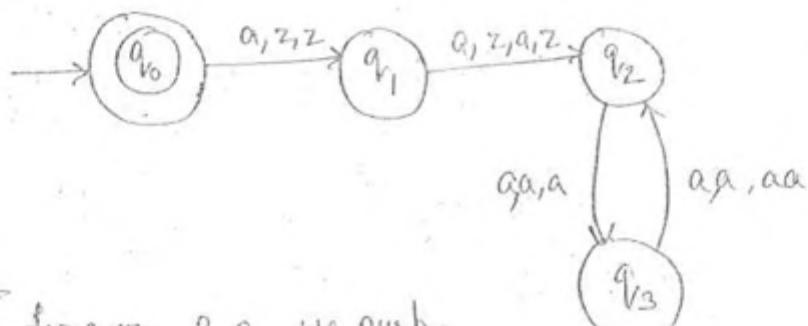
e.g. i) state diagram of $a^n b^n; n \geq 0$



$L(L) \rightarrow$

$L \text{ of } \bar{L}$ is regular
 $L \text{ of } L^c$ is RE
 $L = ?$

e.g. ii) $L = \{a^{2n}, b^n; n \geq 0\}$



[for every 2a we push
 $\underline{\underline{1'b}}$]

e.g. iii) $L = \{a^m b^{n+1}\} \quad m \geq 0$

{ See by own }

giv $L = \{a^m b^n c\} \quad m \geq 0$

giv $L = a^m b^n; m \geq n \quad \left\{ \begin{array}{l} \text{See by own} \\ m \leq n \\ m \neq n \\ m > n \\ m < n \end{array} \right.$

$\Rightarrow \{ \text{CFL} \cap \text{CFL} \} \Rightarrow \text{Not } \underline{\text{CFL}}$

for e.g:- $a^n b^m c^m \quad (\text{CFL}) \cap a^n b^n c^m \quad (\text{CFL}) = a^n b^n c^n \quad (\text{CSL})$

$$\Rightarrow A \cap B = (A^c \cup B^c)^c \Rightarrow \text{DeMorgan's Law}$$

	CFL
→ Membership Algo.	✓
→ finite / Infinite	✓
→ Emptiness	✓

If L is CFL \rightarrow L satisfies Pumping Lemma.

while converse is not true

$\cancel{\exists s}$ is useful, the language is not empty.

$\cancel{\forall s}$ 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
$\frac{P}{P \rightarrow Q}$	$Q \rightarrow P \quad \{ \text{Converse} \}$
$\frac{\sim Q \rightarrow \sim P}{\sim P \rightarrow \sim Q} \quad \{ \text{Contrapositive} \}$	$\sim P \rightarrow \sim Q \quad \{ \text{Inverse} \}$

(ww → Linear → 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.

$$G' \Rightarrow S \rightarrow SS \mid aS \mid bS \mid a \mid b$$

$$G \Rightarrow S \rightarrow aS \mid bS \mid a \mid b$$

$$\boxed{L(G') = L(G)}$$



$$\Rightarrow L(G) \text{ is defined as, } w \in \Sigma^* \mid S \xrightarrow[G]{*} w$$

$$\Rightarrow S \xrightarrow[G]{*} \alpha \quad \alpha \text{ is called as Sentential form}$$

Ques ($a^m b^n$; $m \leq n \leq 3n$)

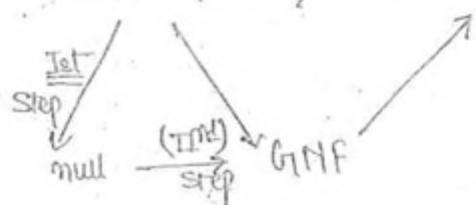
Grammar $\Rightarrow S \rightarrow aSb \mid aSbb \mid aSbbb \mid \lambda$

$$\delta(q_0, 0, z) = (q_1, az) \quad (q_1, aa z), (q_1, aaa z)$$

$$\delta(q_1, a, a) = (q_1, aa) \quad (q_1, aaa), (q_1, aaag)$$

(Conversion of CFG to Machine).

for eg:- CFG \Rightarrow NPDA



Ques

$$S \rightarrow aA$$

$$A \rightarrow aABC \mid bB \mid a$$

$$B \rightarrow b$$

$$C \rightarrow c$$

\Rightarrow The above grammar is already in the form of
Greibach Normal form,

$$\begin{bmatrix} V \rightarrow TV^* \\ \rightarrow T^* \end{bmatrix}$$

Here, Variable portion kept in the stack.

and Terminal kept in the Tape.

a | b | b | λ

Tape

$S \Rightarrow aA \Rightarrow abB \Rightarrow \underline{ab}$

Stack

$$\delta(q_0, \lambda, z) = \{(q_1, S_2)\}$$

$$\delta(q_1, \lambda, z) = \{(q_1, z)\}$$

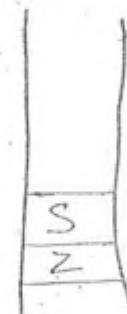
$$\delta(q_1, a, S) = \{(q_1, A)\}$$

$$\delta(q_1, b, A) = \{(q_1, \text{NSC}), (q_1, \lambda)\} \rightarrow \text{represent NDPA}$$

$$\delta(q_1, b, A) = \{(q_1, B)\}$$

$$\delta(q_1, b, B) = \{(q_1, \lambda)\}$$

$$\delta(q_1, c, C) = \{(q_1, \lambda)\}$$



→ If the grammar has n production maximum Commands
one $(n+2)$.

↳ Any CFG, accepted by machine (NDPA | DPDA) is
of at most 3 states.

done

(Question to be ask):-

→ a. Type '0' grammar / RE grammar can generate Regular grammar

if i) True / false, ii) Pumping Lemma \Rightarrow Ans?

→ Every CFG generate CSL (that one under CL)

→ every regular grammar is a linear grammar, $\{(\text{subset}) \mid (\text{total})\}$

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

is $|\Sigma|^n$ True / False

→ $a^n b^n c^n$

Set of productions :-

$S \rightarrow aSBc \mid aBc$

$cB \rightarrow BC$,

$0B \rightarrow ab$,

$bB \rightarrow bb$,

$bC \rightarrow bc$,

$cC \rightarrow cc$

\Rightarrow equivalent

to CSL

but not CSL

due to $cB \rightarrow BC$,

True / False

Date: 16/09/10

↳ atleast 1 a's and 2 and 2 b's.

$$\begin{array}{c}
 (a+b)^* a (a+b)^* + b(a+b)^* + b(a+b)^* \\
 | \qquad \qquad \qquad | \qquad \qquad \qquad | \\
 (a+b)^* b (a+b)^* + a(a+b)^* + b(a+b)^* \\
 | \qquad \qquad \qquad | \qquad \qquad \qquad | \\
 (a+b)^* b (a+b)^* + b(a+b)^* + a(a+b)^* \\
 \end{array}
 \left\{ \begin{array}{l} \\ \\ \end{array} \right.$$

$B \rightarrow$

$B \xrightarrow{a} L$
 $B \xrightarrow{b} L$

$C \xrightarrow{a} -$
 $B \xrightarrow{a} C$

$C \xrightarrow{b} B$
 $B \xrightarrow{b} -$

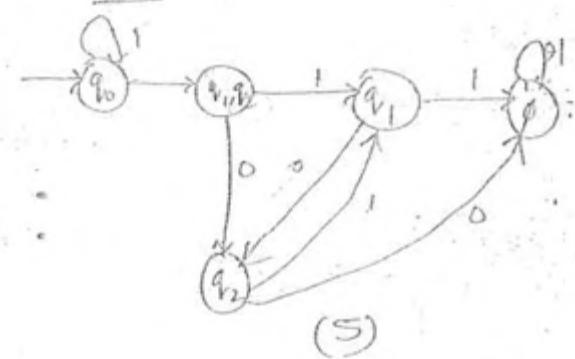
$(00)^* 0 \rightarrow \underline{\text{odd group}}$

$00^* 0 \rightarrow \underline{\text{even group}}$

$$(00)^* 0 \neq 00^* 0$$

$$(00)^* 0 = \underline{\underline{000}}^*$$

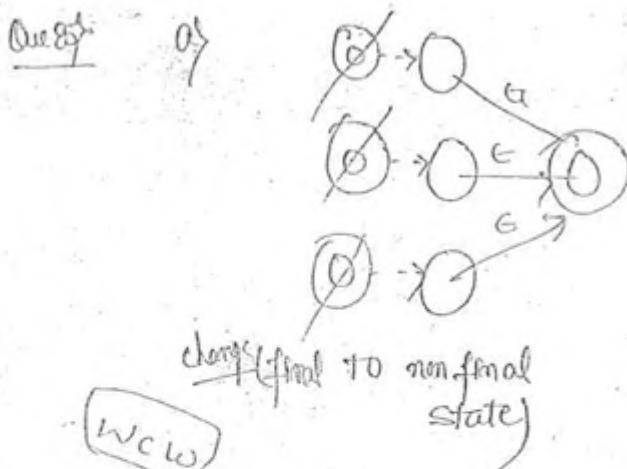
Self loop



$$\Rightarrow 0^* 1^* \Rightarrow 0^* + 1^* + 0^* 1^*$$

$$\Rightarrow 0^* 1^* 2^* \Rightarrow 0^* + 1^* + 2^* + 0^* 1^* + 0^* 2^* + 1^* 2^* + 0^* 1^* 2^*$$

Ques 85 :-



But in dfa it is
not possible,
because, In dfa, null
move is not possible

Ques 86 :-

$$L = (\overline{a+b})^* (e+b)$$

$$h(L) = \left(h(a)^* + h(b) \cdot N(b) \right)^* (e + h(b))$$

e.g:-

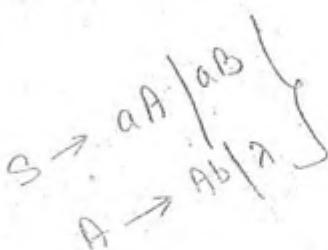
$$L = \{01\}$$

$$h(a) = 00$$

$$\therefore h(b) = 11$$

$$h^{-1}(L) = \emptyset$$

$$h(h^{-1}(L)) = h(\emptyset) = \emptyset.$$



$$\begin{aligned} & \Rightarrow S \rightarrow aA \\ & \rightarrow \underline{aAb} \\ & \rightarrow aaAbb \\ & \rightarrow aaaAbbb \\ & \rightarrow \boxed{a^n b^n} \end{aligned}$$

$$\begin{aligned} & S \rightarrow aCa \\ & C \rightarrow \underline{aCa} | b \end{aligned}$$

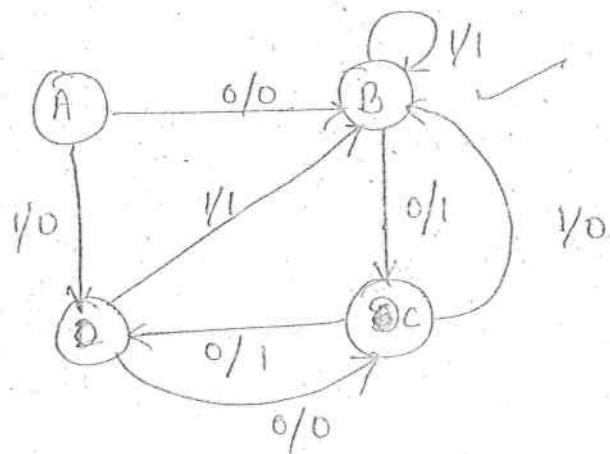
$$\begin{aligned} & S \rightarrow aCa \\ & \rightarrow a\underline{a}Ca \\ & \rightarrow aa\underline{a}Ca \\ & \rightarrow aaabaaa \\ & \rightarrow \boxed{a^n b a^n} \end{aligned}$$

$$U, L^c \Rightarrow \oplus$$

$$A \oplus B \Rightarrow (A - B) \cup (B - A)$$

$$\Rightarrow (A \cap B^c) \cup (B \cap A^c)$$

Sol 132



Sol

a) 01

b) 10

c) 101

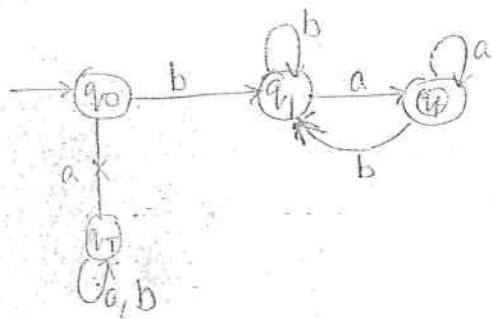
d) 110

\Rightarrow last bit condition also asked?

①

~~Only~~ Not neither starting with 'a' nor ending with 'b'
Opposite

~~Sol~~ either starting with 'b' or ending with 'a'.

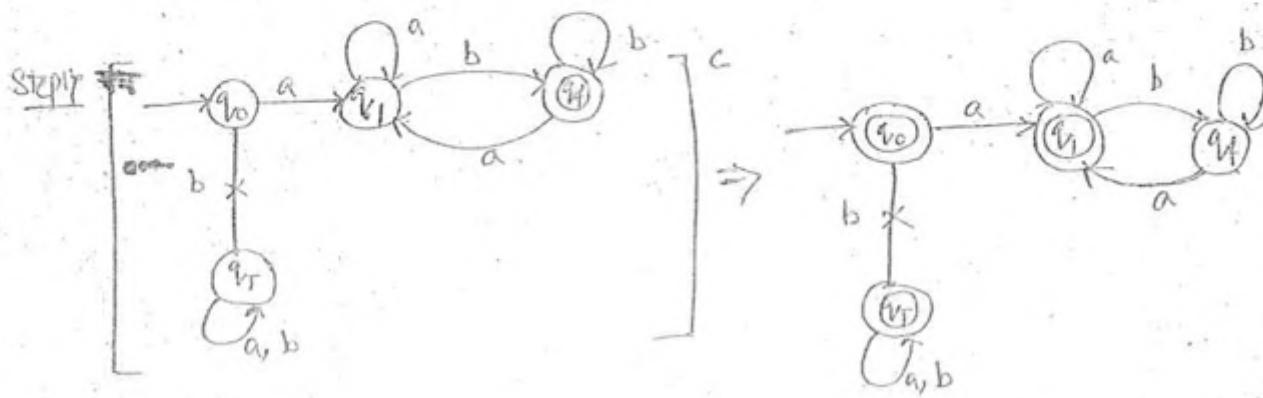


20/09/10Context free grammarQues

Not starting with 'a' or not ending with 'b'.

↳ [Not starting with 'a' or not ending with 'b']^c

→ [Starting with 'a' and ending with 'b']



n.e. $\Rightarrow \lambda + a(a+b)^* + (a+b)^*b$

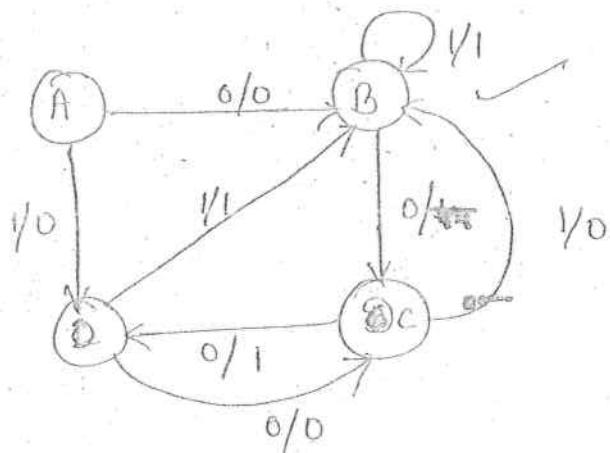
or

n.e. $\Rightarrow \lambda + a(a+bb^*a)^* + b(a+b)^*$

$$U, L^c \Rightarrow \oplus$$

$$\begin{aligned} A \oplus B &\Rightarrow (A - B) \cup (B - A) \\ &\Rightarrow (A \cap B^c) \cup (B \cap A^c) \end{aligned}$$

Sol) 132



Sol)

a) 01

b) 10

$\Rightarrow 101$

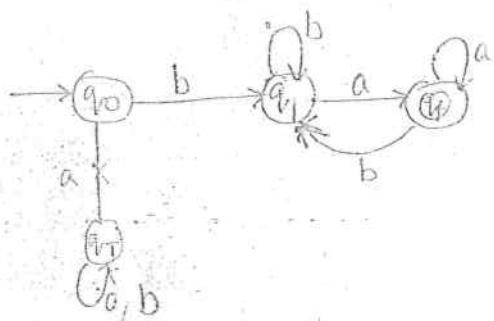
$\Rightarrow 110$

\Rightarrow last bit condition also asked?

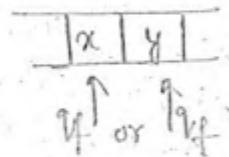
①

~~Q0~~ ~~Not~~
Neither starting with 'a' nor ending with 'b'
~~Not~~ opposite

~~Q1~~ \Rightarrow either starting with 'b' or ending with 'a'.



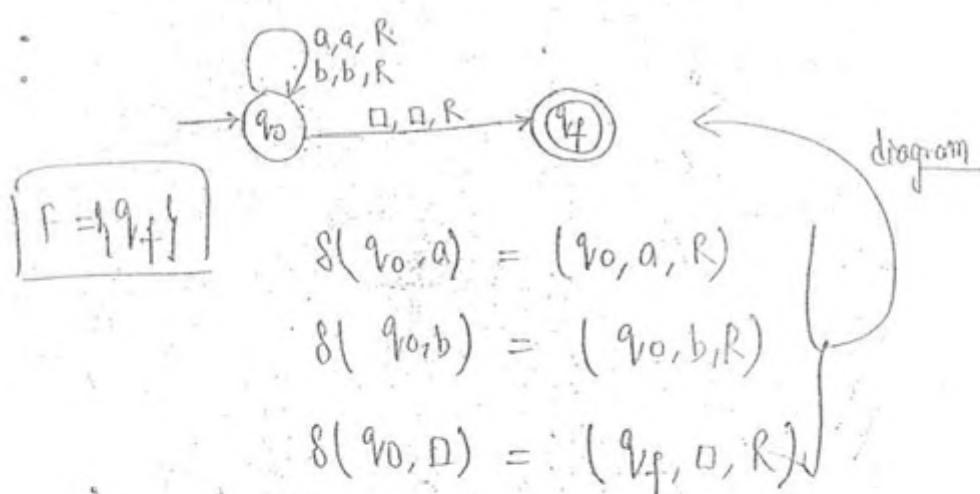
↳ $q_f \neq q_f \rightarrow$ Tells about the final state whether on left or right.



↳ $x, y \in \Gamma^*$ → It shows Input Tape.

Ques Converting a machine into language.

$$M \rightarrow L(M)$$



Ques What language accepting by TM.

Soln

$$(a+b)^*$$

$|a|b|a|b|0|0|$

Ques What is set of string (w) it will built?

Soln

$$(a+b)^*$$

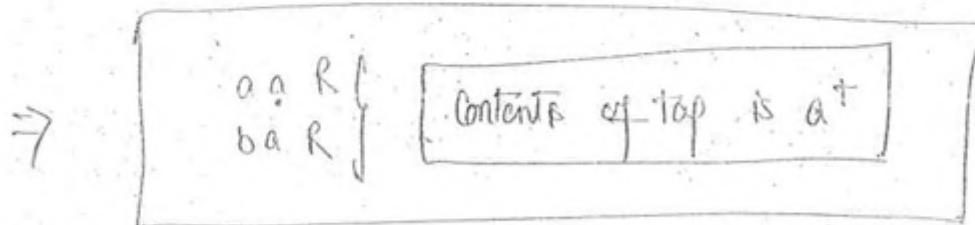
Ques $w \rightarrow \text{hang}$.

Solt \emptyset (empty)

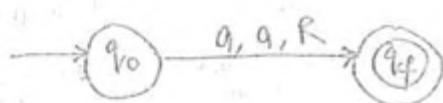
Ques Contents of Top of set,

Solt Unchanged (By default always)

If it goes 'L', it will hang



eg 2

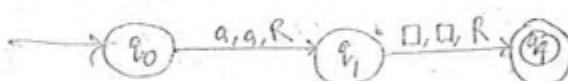


$\boxed{\square a a b \square} \xrightarrow{a,a,q_0} \boxed{\square \square \square} \xrightarrow{0|p} a(a+b)^*$

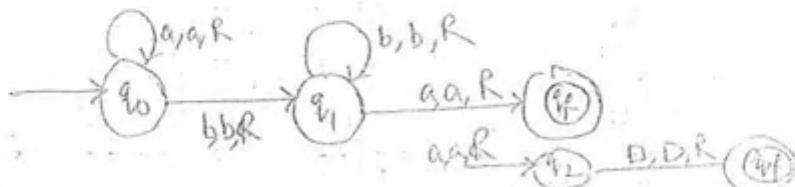
iii 4

$\boxed{\square a \square}$

$0|p \Rightarrow (0) \text{ or (exactly) only } a$



iv

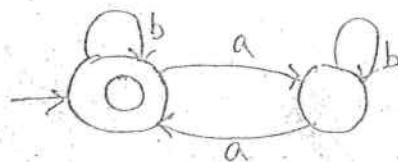


$\Rightarrow a^* b b^* (a+b)^*$

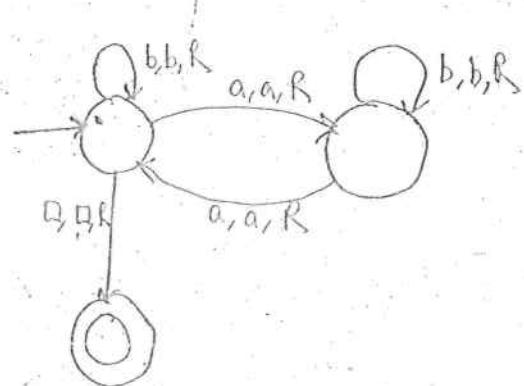
$\Rightarrow a^* b b^* a$

Even as

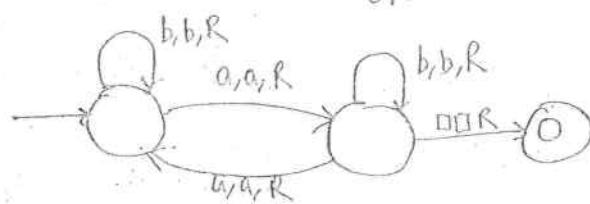
DFA



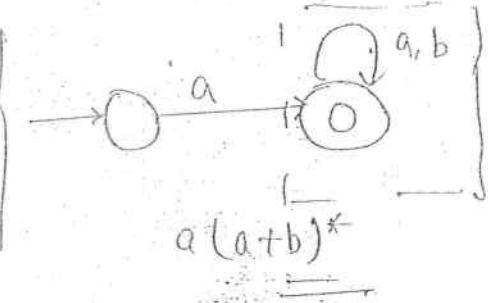
TM



OR

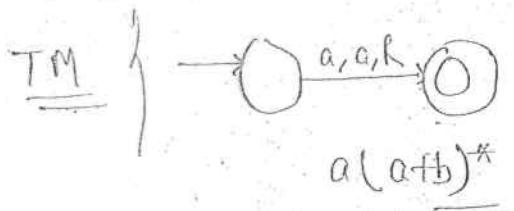


DFA



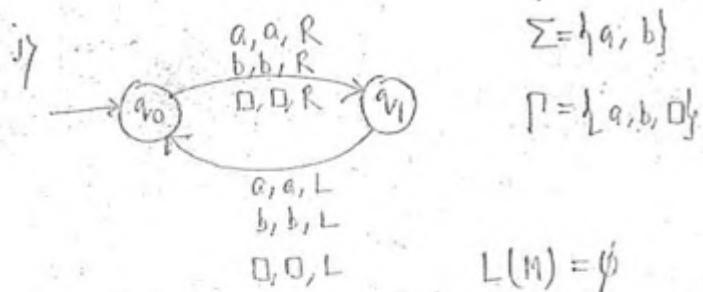
Turning M/c
does not
allow it,

Rewriting
this,



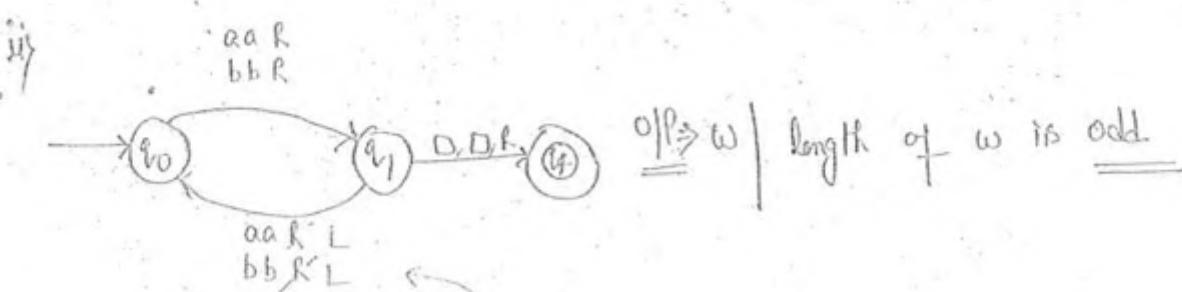
TM





$$\text{halt} = \emptyset$$

$$\text{hang} = (a+b)^+$$



\Rightarrow If Condition changed

Then output

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

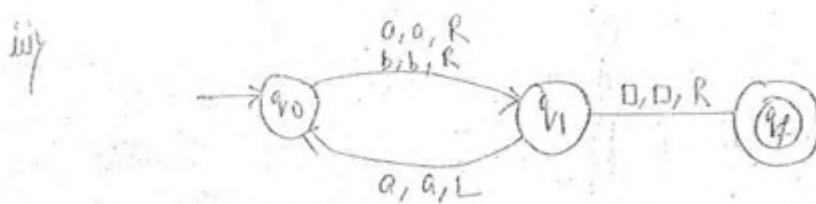
$$\text{halt} = \{a, b\}$$

$$\text{hang} = [(a+b)^+] - \{a, b\} \text{ or } 2$$

$$w \left| \begin{array}{l} \text{length of } w \geq 2 \end{array} \right.$$

$\square a \square$	Hang
$\square b \square$	Hang
$\square a \square$	Hang
$\square a b \square$	Hang
$\square b c \square$	Hang
$\square b b \square$	Hang

Content of Tape \Rightarrow Unchanged.



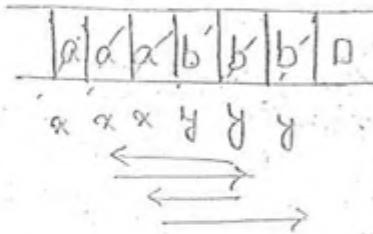
op $L(M) = \{a, b\}$

$$\text{halt} \Rightarrow (a+b)(a+b)^*$$

$$\text{hang} \Rightarrow (a+b)a(a+b)^* \text{ | Anything starting with } a$$

3. Turning Machine as acceptor

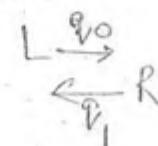
$$\{ \underline{a^n b^n} ; n \geq 1 \}$$



$$\delta(q_0, a) = (q_1, x, R)$$

(When L and Right move
stat must change)

$$\delta(q_1, a) = (q_1, a, R)$$



$$\delta(q_1, y) = (q_1, y, R)$$

$$\delta(q_1, b) = (q_2, y, L)$$

$$\delta(q_2, y) = (q_2, y, L)$$

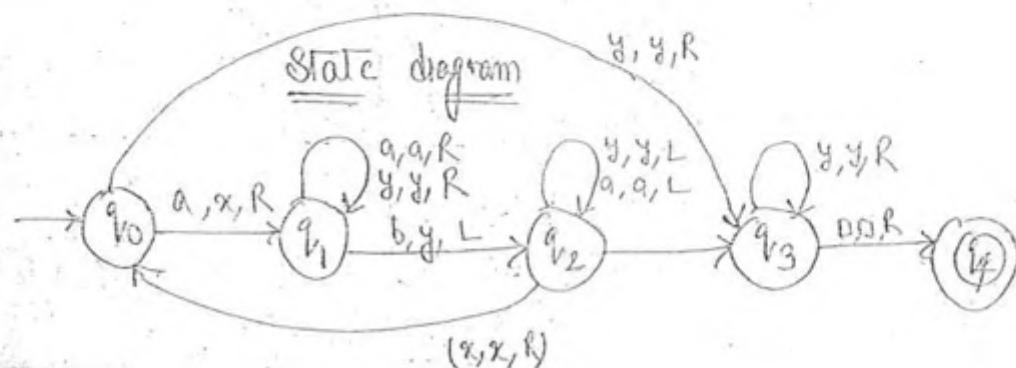
$$\delta(q_2, a) = (q_2, a, L)$$

$$\delta(q_2, x) = (q_0, x, R)$$

(It is only possible when, a is finished) $\leftarrow \delta(q_0, y) = (q_3, y, R)$

$$\delta(q_3, y) = (q_3, y, R)$$

$$\delta(q_3, \square) = (q_4, \square, R)$$



$$q_0 = q_0 1 + q_0 0 1 + \lambda$$

$$q_0 \Rightarrow q_0(1+01) + \lambda$$

$$\Rightarrow R = R 1 + Q$$

$$R = Q P^*$$

$$\therefore q_0 = \lambda (1+01)^*$$

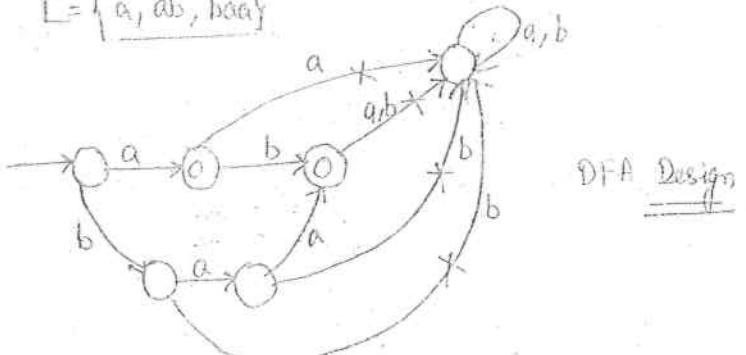
$$\hookrightarrow q_1 = q_1 0 0^* 1 (0+1)^*$$

$$\Rightarrow q_0 0 0 0^* 1 (0+1)^*$$

$$q = \boxed{(1+01)^* 0 0 0^* 1 (0+1)^*}$$

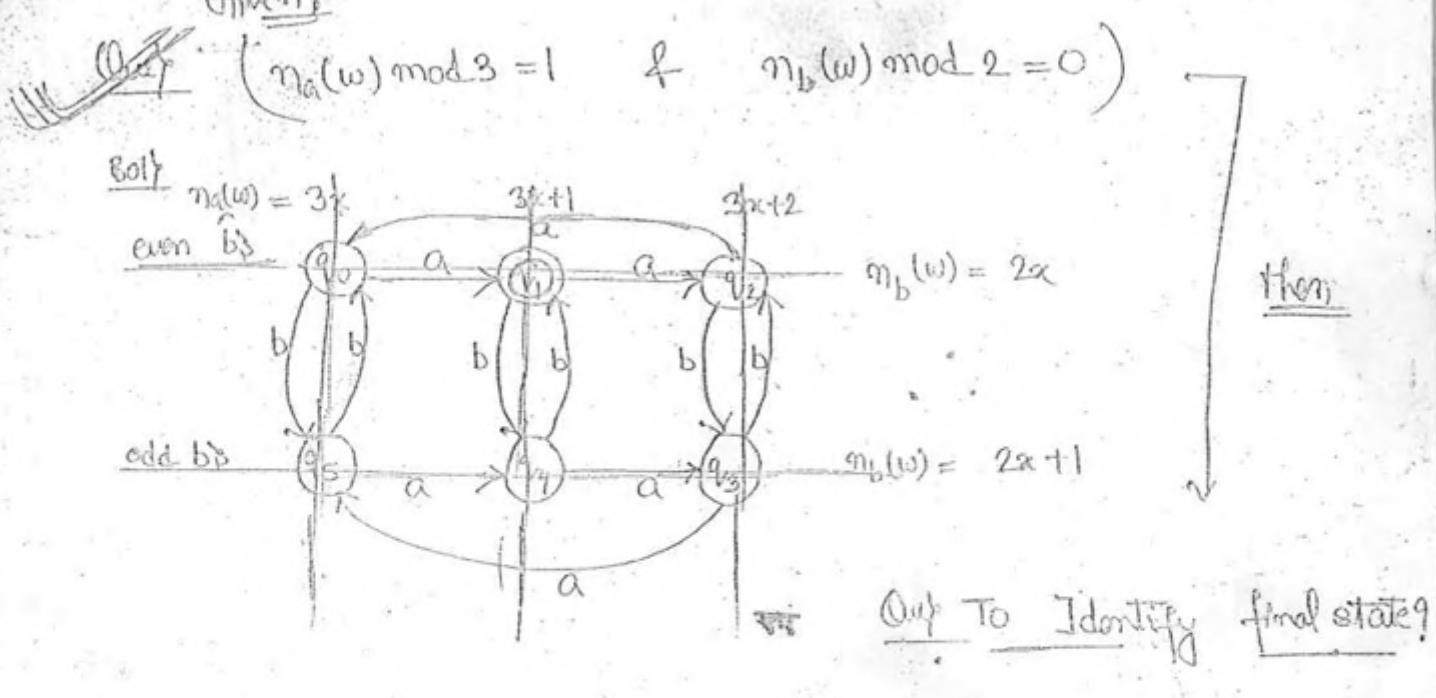
Ques $L = \{a, ab, baa\}$

Sol:



DFA Design

Ques



T.D.C AUTOMATA

(106)