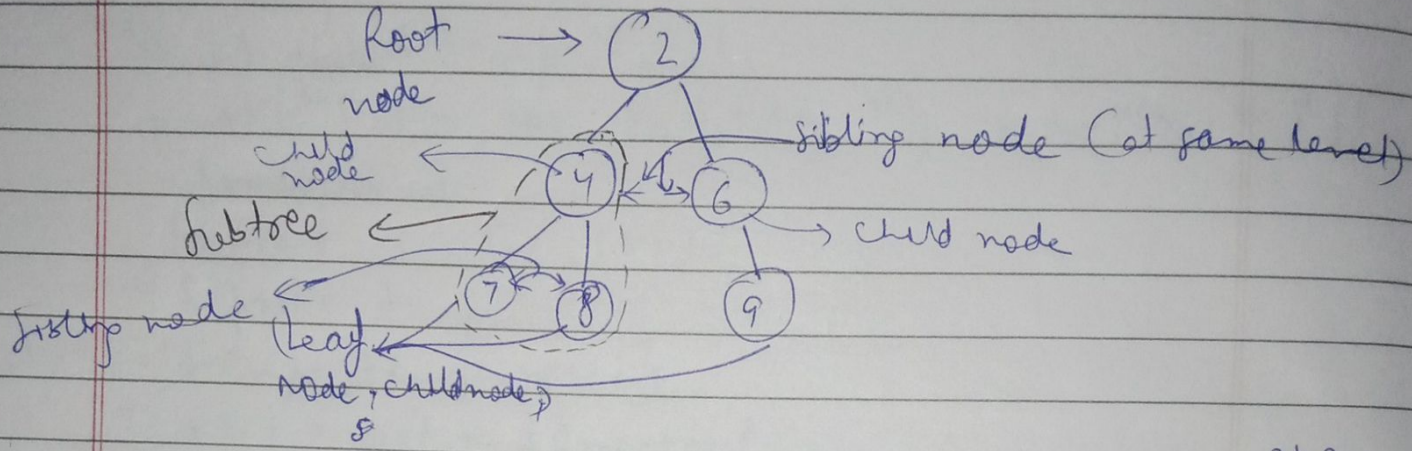
DATE: 25/9/23
PAGE:

Unit II

Tree (Non-Linear Data Structure)

- It is hierarchical in nature.
- We follow top to bottom approach in it.



Depth of node: Path from root node to given node. (Count no of edges that has been traversed).

Note: If in a tree n nodes
so, (n-1) edges will be there.

Height of tree
Path from root to the farthest leaf node (node farther from root node).

Internal nodes → Nodes having atleast one child. (2, 4, 6 are internal node).

DATE: __/__/____
PAGE: _____

Binary Tree

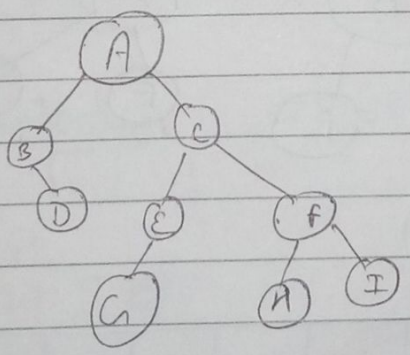
Tree having nodes having atmost two child.

Binary Tree

- **Full / Strict Binary Tree** (Have 2 or 0 child)
- **Complete Binary Tree** (One level is completed then another level is started)
- **Perfect binary Tree** (All the internal nodes will have exactly 2 child).

Traverse in Tree

- Inorder (L Root R)
- Preorder (Root L R)
- Postorder (L R Root)



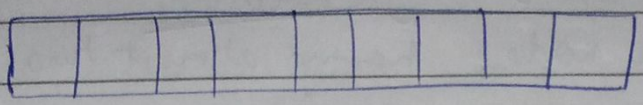
Inorder

1)
2) B
3) B D
4) B D A
5) B D A G
6) B D A G E
7) B D A G E C
8) B D A G E C H
9) B D A G E C H F
10) B D A G E C H F I

Good Write

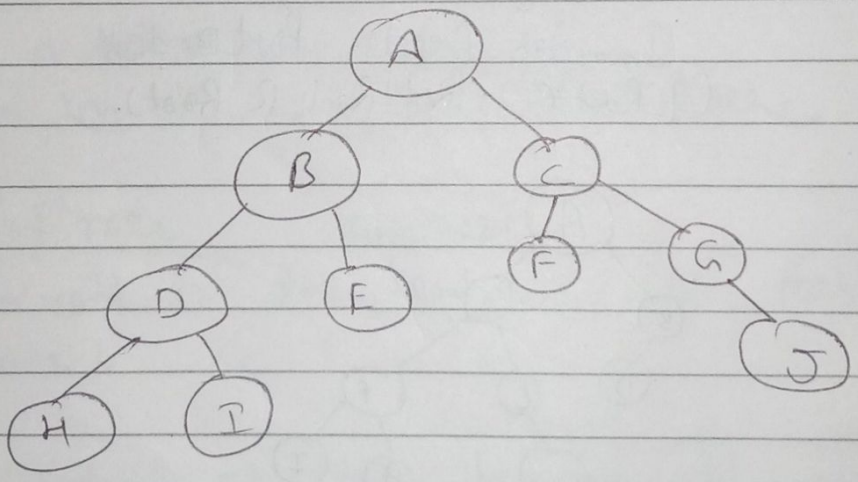
DATE: __/__/____
PAGE: _____

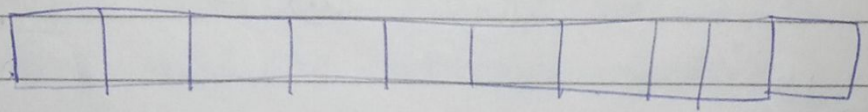
Preorder

1) 

2) A
3) A B
4) A B D
5) A B D C
6) A B D C E
7) A B D C E G
8) A B D C E G F
9) A B D C E G F H
10) A B D C E G F H I

Q. Do solve (In Inorder form) (L Root R)



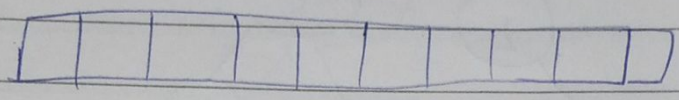
Sol→ 1. 

2. H
3. H D
4. H D I
5. H D I B
6. H D I B E
7. H D I B E A
8. H D I B E A F

Good Write

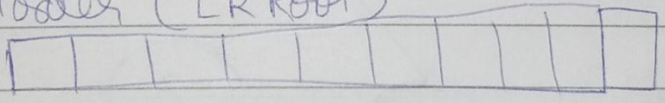
9. HDIBEAFC
10. HDIBEAFCG
11. HDIBEAFCGJ Ans

Do in Preorder (RootLR)

1. 

2. A
3. AB
4. ABD
5. ABDH
6. ABDHI
7. ABDHIE
8. ABDHIEC
9. ABDHIECF
10. ABDHIECFG
11. ABDHIECFGJ

Do in Postorder (LRRoot)

1. 

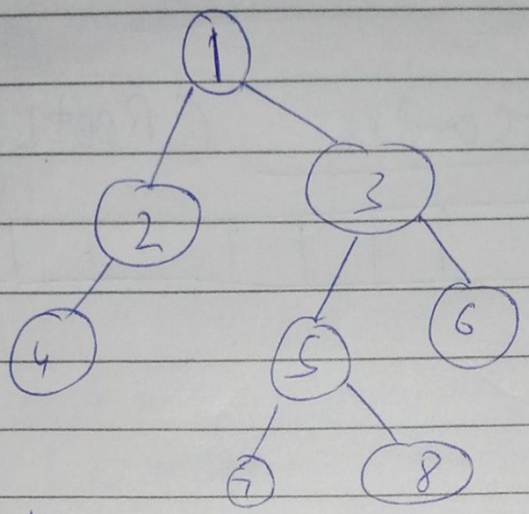
2. H
3. HI
4. HID
5. HIDF
6. HIDEB
7. HIDEBF
8. HIDEBFJ
9. HIDEBFJG
10. HIDEBFJGC
11. HIDEBFJGCA

Good Write

DATE: __/__/____
PAGE: _____

Q Construct a tree if
inorder is [4, 2, 1, 7, 5, 8, 3, 6]
& preorder is [1, 2, 4, 3, 5, 7, 8, 6]

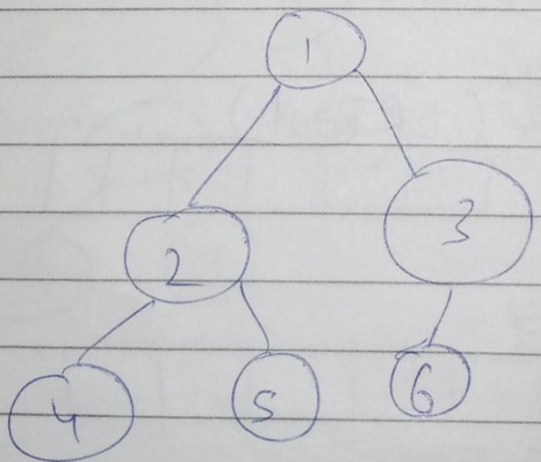
Sol →



[Hint: Preorder se ek ek value pick up in order then checked in inorder to place elements in which position.]

Q Construct tree
inorder [4, 2, 5, 1, 6, 3]
preorder [1, 2, 4, 5, 3, 6]

Sol →



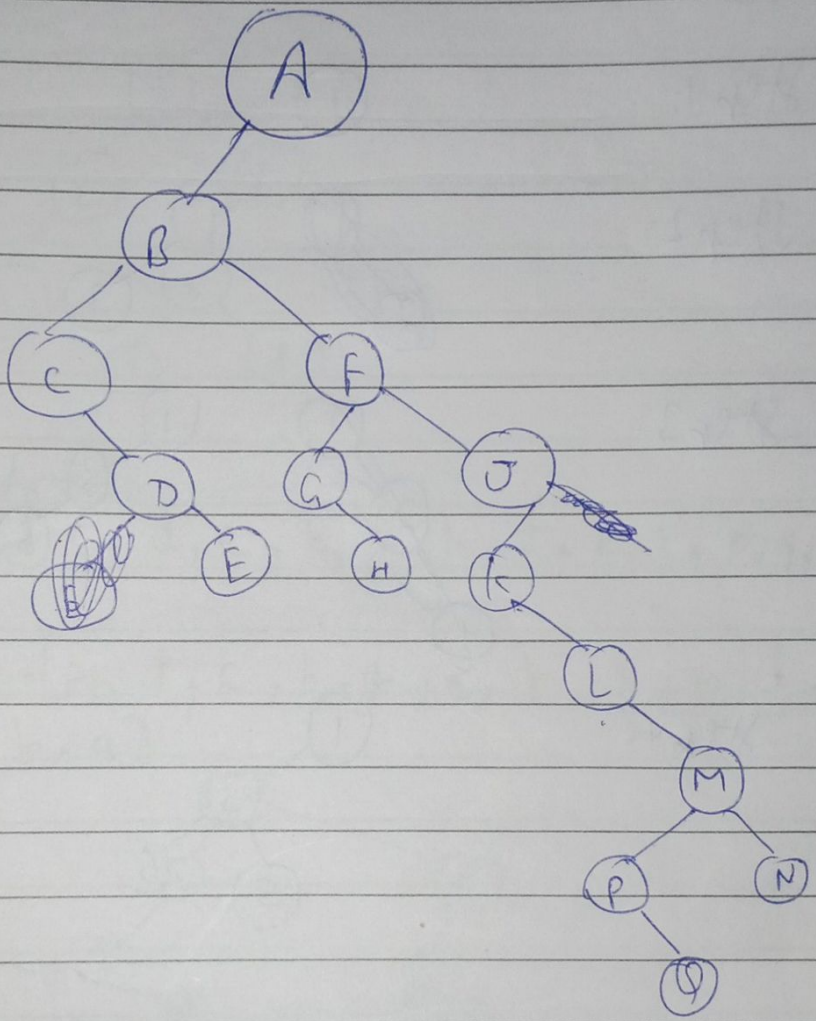
DATE: __/__/____
PAGE: _____

Q. Construct tree.

inorder [C, D, E, B, G, H, F, K, L, P, Q, M, N, J, A]

preorder [A, B, C, D, E, F, G, H, J, K, L, M, P, Q, N]

Sol →



DATE: 29/9/23
PAGE:

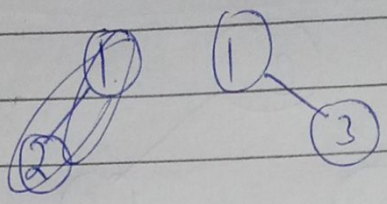
Q. Construct tree

Inorder → [4, 2, 1, 7, 5, 8, 3, 6]
Postorder → [4, 2, 7, 8, 5, 6, 3, 1]

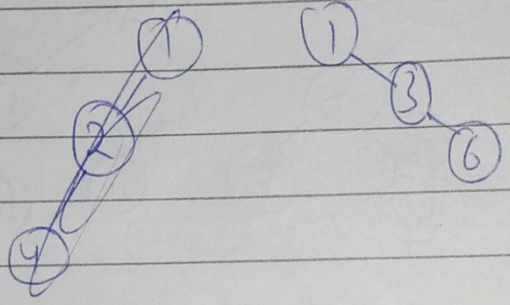
Sol →

Step 1:

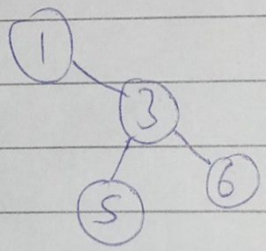
Step 2:



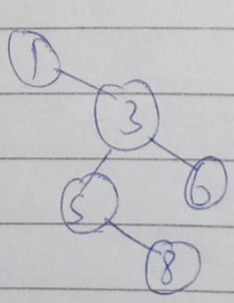
Step 3:



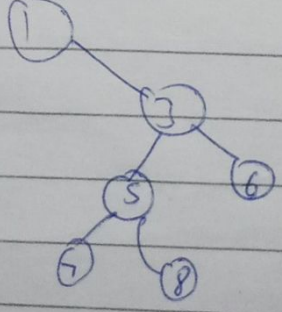
Step 4:



Step 5:

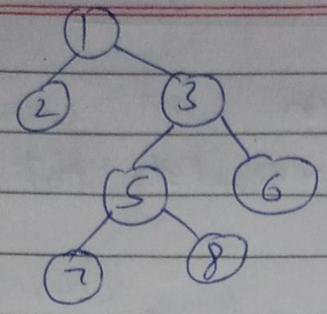


Step 6:

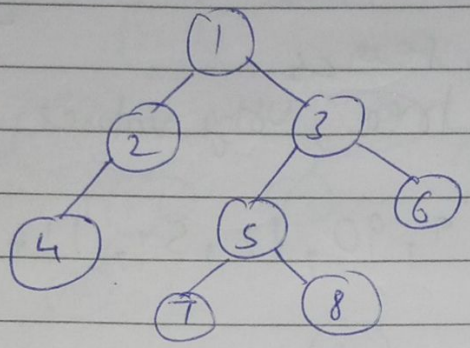


DATE : __/__/____
PAGE : _____

Step 7:



Step 8:

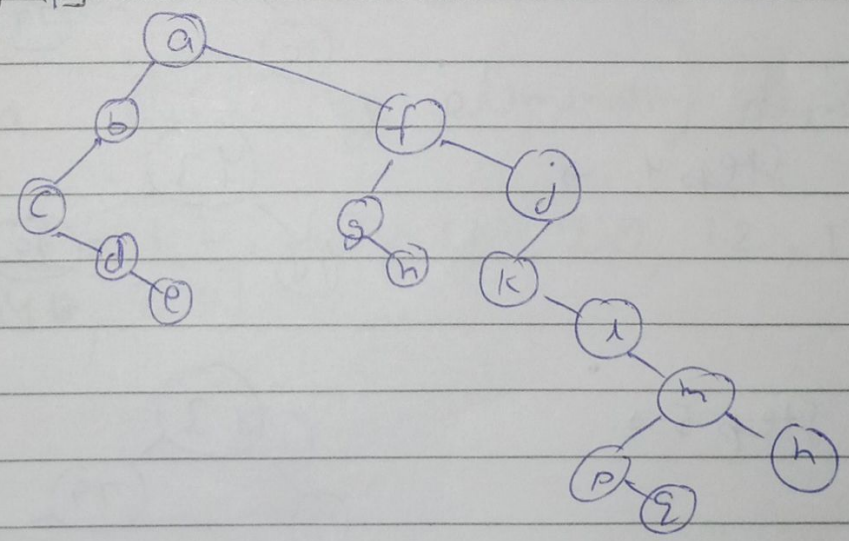


Q Construct tree:

inorder → [c, d, e, b, g, h, f, k, l, p, q, m, n, j, a]

postorder → [e, d, c, h, g, q, p, n, m, l, k, j, f, b, a]

Sol →



DATE: 3/10/23
PAGE:

Binary Search Tree

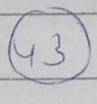
New node is less than root come on left.
And if new node is more or equal to root come on right.

→ Merit: → Searching become easier.
→ Improved Time complexity efficiency.

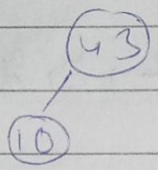
Q2 (University Ques) Construct a Binary Search tree using values

43, 10, 79, 90, 12, 54, 11, 9, 50

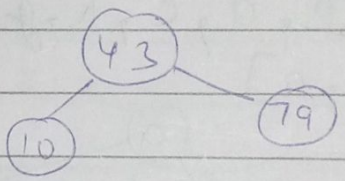
Sol→ Step 1 →



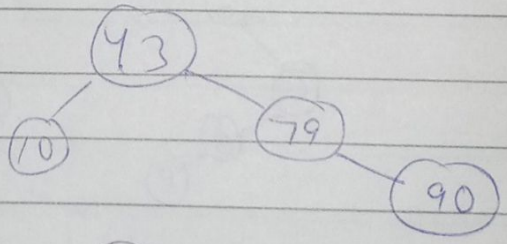
Step 2 →



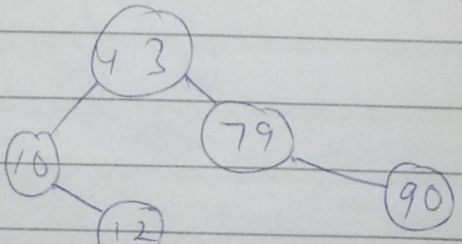
Step 3→



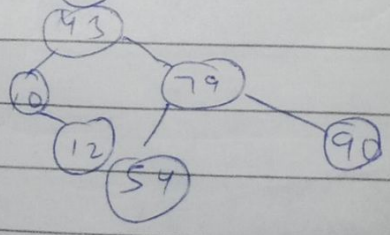
Step 4 →



Step 5 →

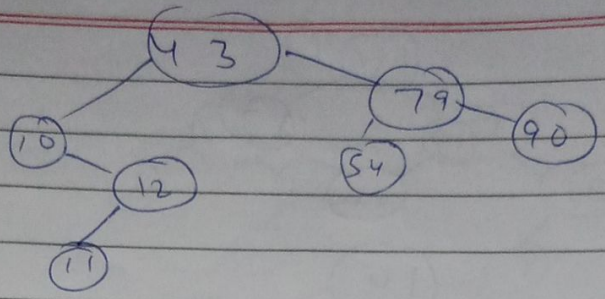


Step 6 →

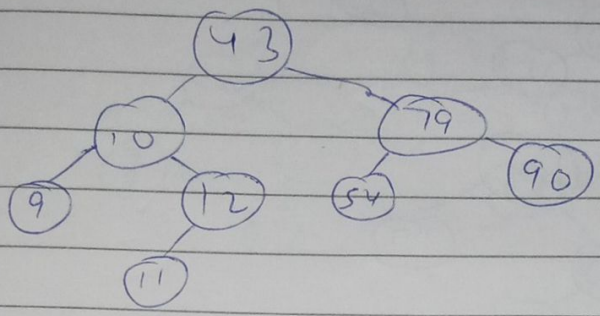


DATE : __/__/____
PAGE : _____

Step 7 →

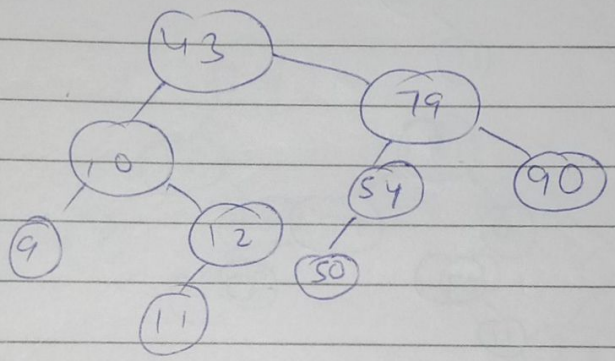


Step 8 →



Step 9 →

Ans →

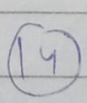


(Ques University)

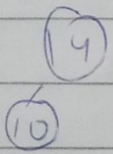
Q - Create a BST by eliminating duplicate values.

[14, 10, 17, 12, 10, 11, 20, 12, 18, 25, 20, 8, 22, 14, 23]

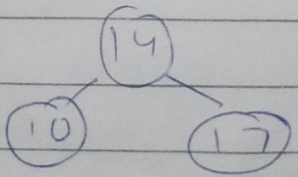
Sol → Step 1 →



Step 2 →



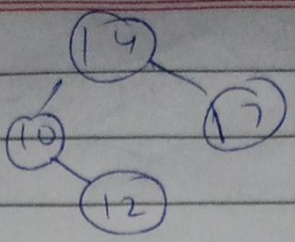
Step 3 →



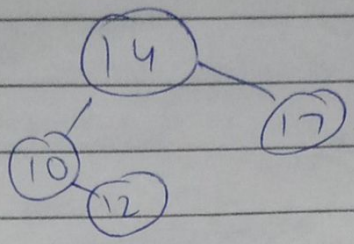
Good Write

DATE : __/__/____
PAGE : _____

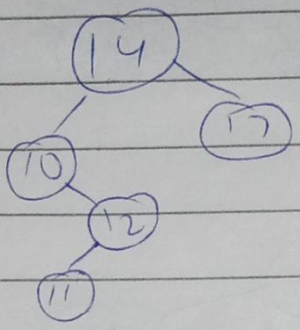
Step 4 →



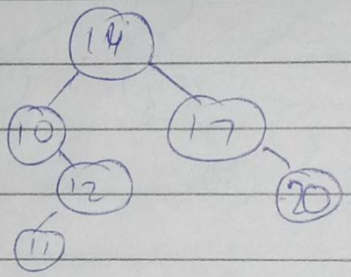
Step 5 →



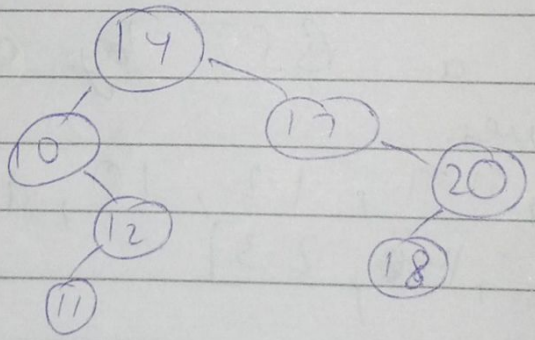
Step 6 →



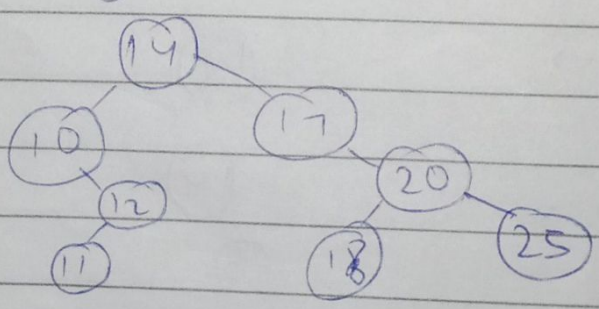
Step 7 →



Step 8 →

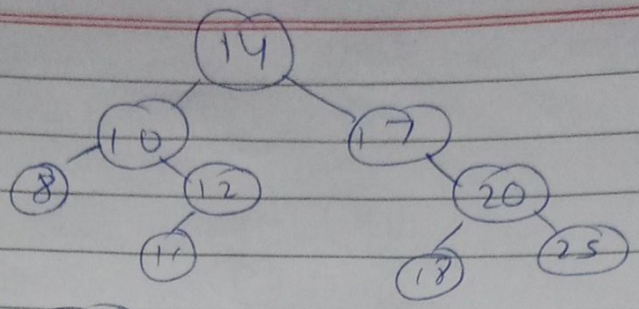


Step 9 →

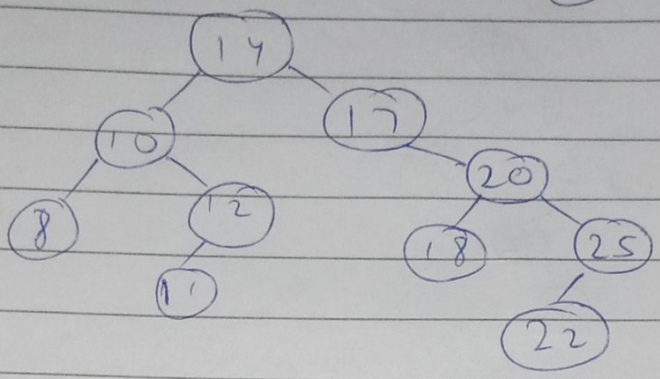


DATE: __/__/____
PAGE: _____

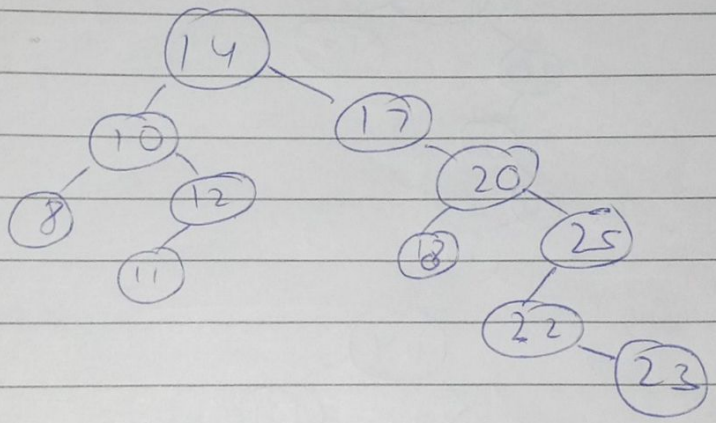
Step 10 →



Step 11 →



Step 12 →

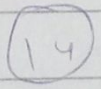


Q2 Construct a BST without eliminating duplicate values.

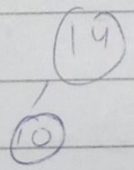
[14, 10, 17, 12, 10, 11, 20, 12, 18, 25, 20, 8, 22, 14, 23]

Sol →

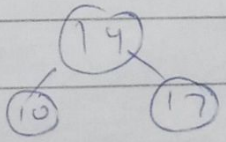
1.



2.



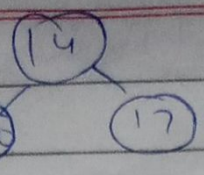
3.



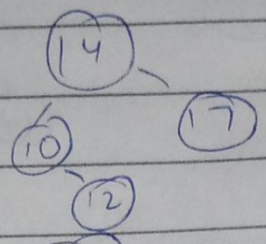
Good Write

DATE : __/__/____
PAGE : _____

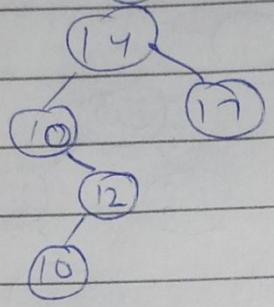
4.



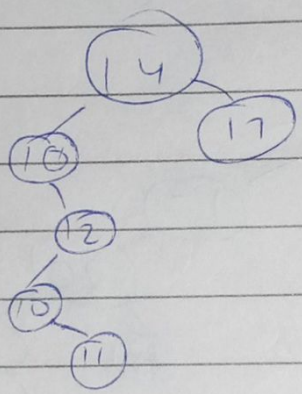
5.



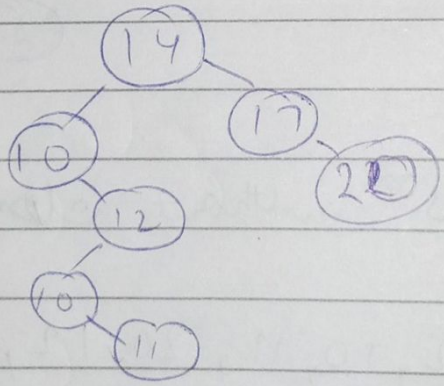
6.



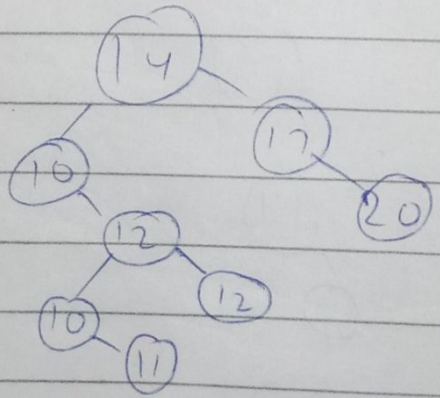
7.



8.

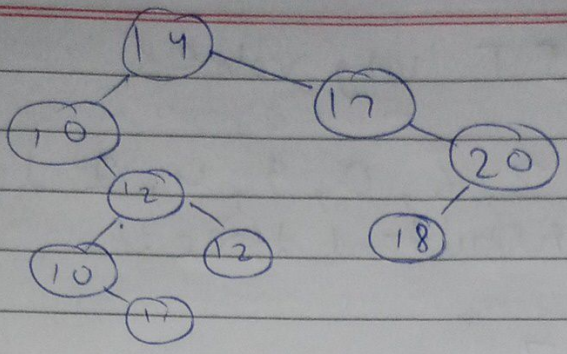


9.

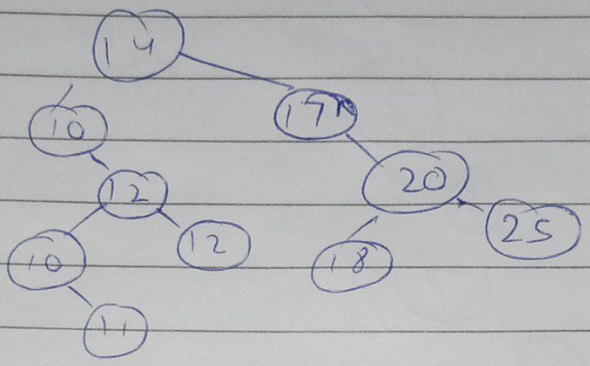


DATE : __/__/____
PAGE : _____

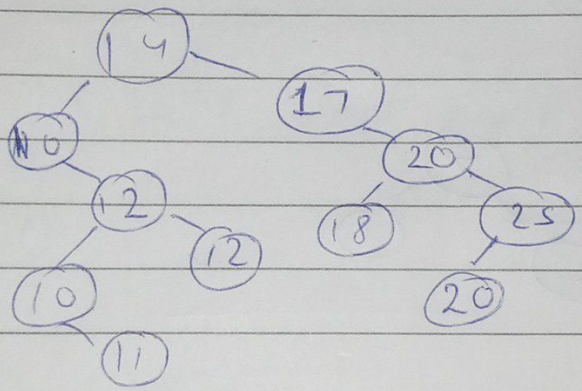
10.



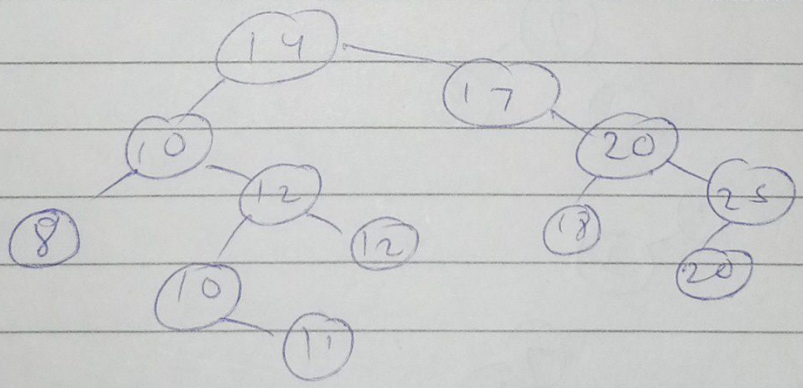
11.



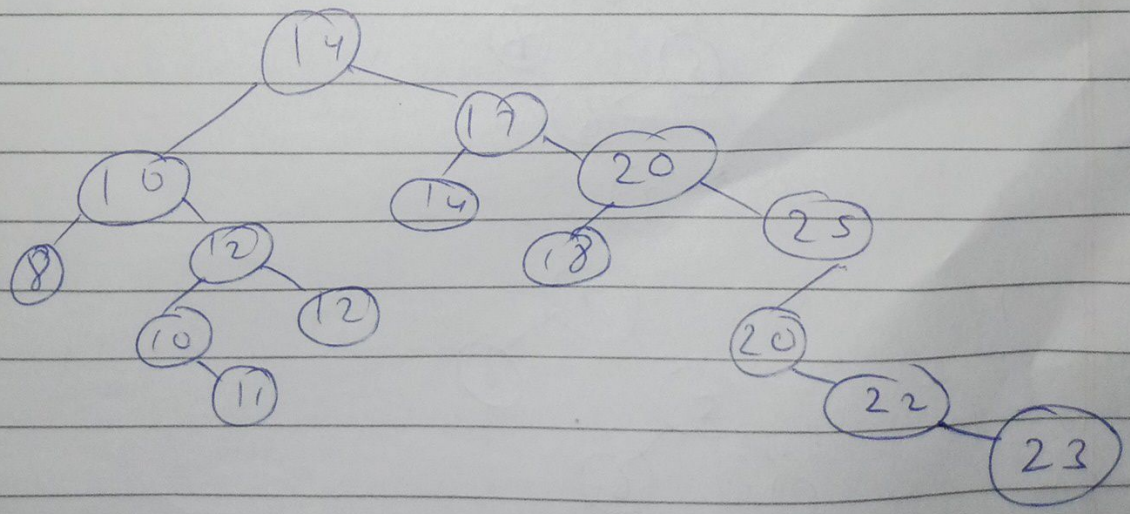
12.



13.



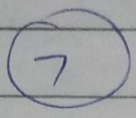
14.



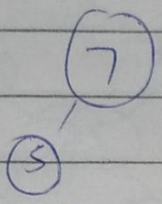
Q) Create a BST using values [7, 5, 1, 8, 3, 6, 0, 9, 4, 2] Write inorder traversal of resultant tree.

Sol:

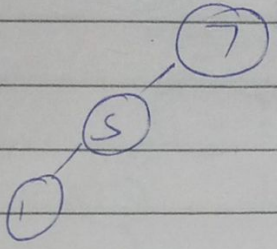
1.



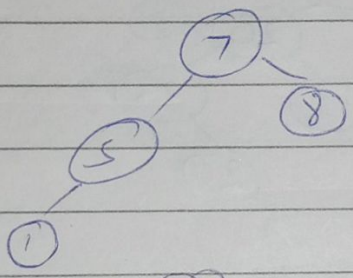
2.



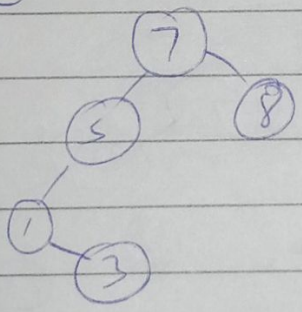
3.



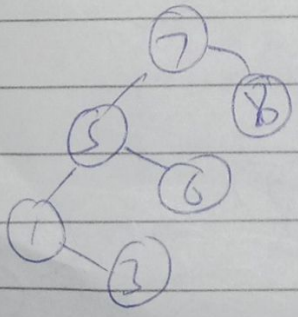
4.



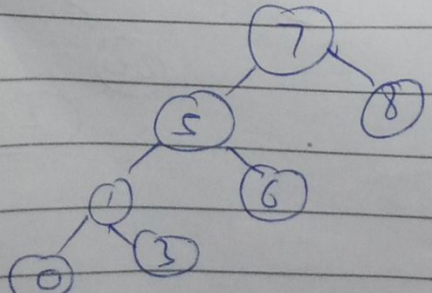
5.



6.



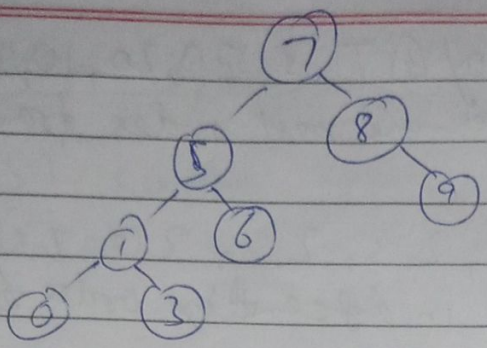
7.



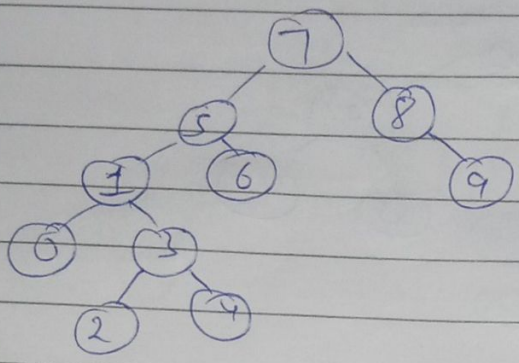
Good Write

DATE : ___/___/_____
PAGE : _____

8.



Ans→ 9.



Inorder (Left Root Right)

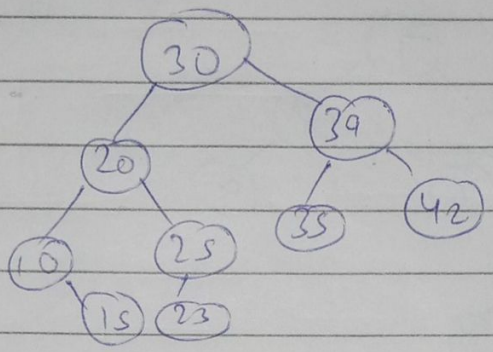
1. 0
2. 0, 1
3. 0, 1, 2
4. 0, 1, 2, 3
5. 0, 1, 2, 3, 4
6. 0, 1, 2, 3, 4, 5
7. 0, 1, 2, 3, 4, 5, 6
8. 0, 1, 2, 3, 4, 5, 6, 7
9. 0, 1, 2, 3, 4, 5, 6, 7, 8
10. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 Ans

11.

DATE: 5/10/23
PAGE:

Q. The preorder traversal of BST is 30, 20, 10, 15, 25, 23, 39, 35, 42. Determine post order traversal of above tree.

Sol → Inorder = 10, 15, 20, 23, 25, 30, 35, 39, 42
(For BST, arrange in ascending order for inorder)



Post order ⇒ 15, 10, 23, 25, 20, 35, 42, 39, 30

• Deletion in BST

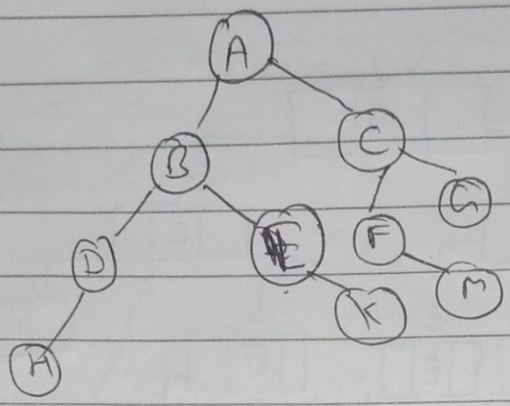
3 categories

- leaf node (remove)
- node have 1 child (swap with parent & remove)
- node have 2 child (swap with immediate successor in inorder traversal & remove)

DATE: __/__/____
PAGE: _____

Limitation of BST

→ BST can be represented by using array or Linked List in a memory.



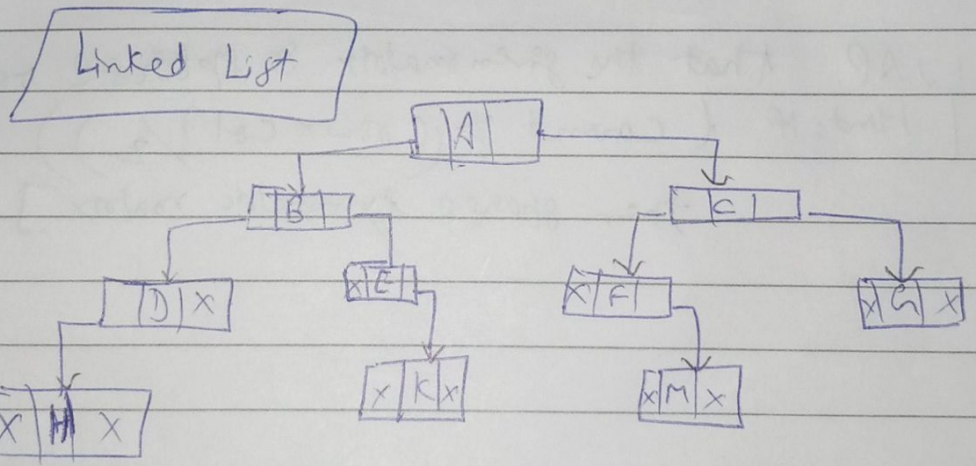
Array

	A	B	C	D	E	F	G	H	_	_	K	----
	1	2	3	4	5	6	7	8	9	10	11	

i

2i 2i+1

where _ is Null



The memory representation of BST using array or Linked list contains large no. of null values. This is wastage of memory.

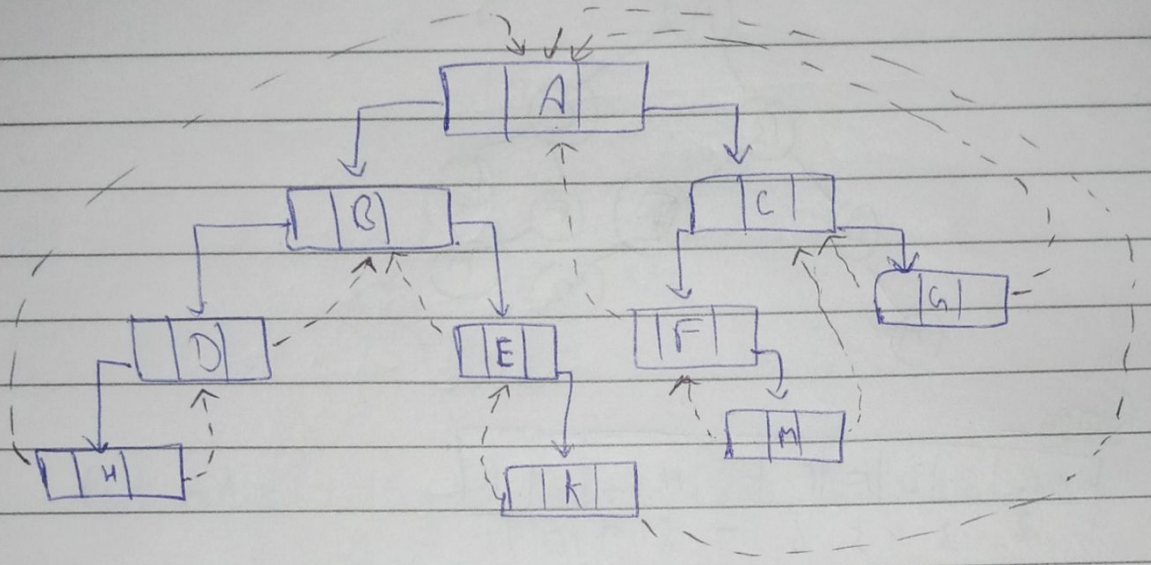
This wastage of memory can be reduced by using threaded Binary Tree.

DATE: __/__/____
PAGE: _____

Threaded BST

Merit :→
Traversal become easier.

Inorder → H D B E K A F M C G



Q2 WAP that the given matrix is sparse or not.
[Hint: if (count > (row × col) / 2))
then sparse symmetric matrix]

AVL

ABL

DATE: 9/1923
PAGE:

→ Another name → Height Balance BST

Balance factor = height of left subtree − height of right subtree

if ans → −1, 0, 1

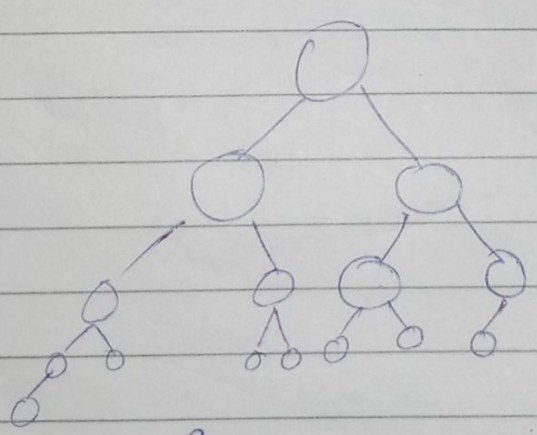
↳ Then Balanced

Others than these no. ; then tree is unbalanced.

To balance ways are :-

- L L
- R R
- L R = R R + L L
- R L = L L + R R

Eg →

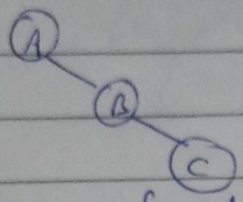


Balance factor = 4 − 3 = 1

So, this tree is balanced

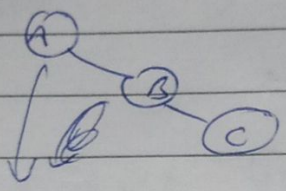
DATE : __/__/____
PAGE : _____

- RR

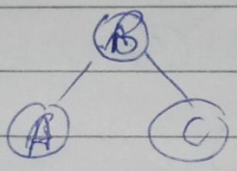


Balance factor = 0 - 2 ⇒ -2

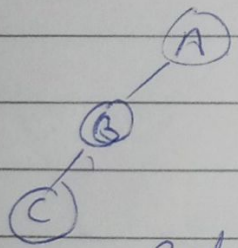
So, to balance



Result is →

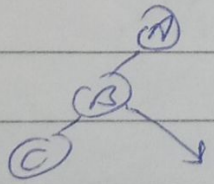


- LL

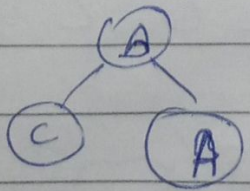


Balance factor = 2 - 0 ⇒ 2

So, balance

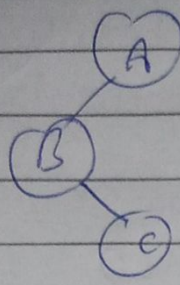


Result is



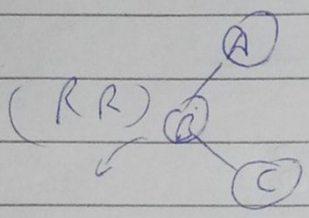
DATE : __/__/____
PAGE : _____

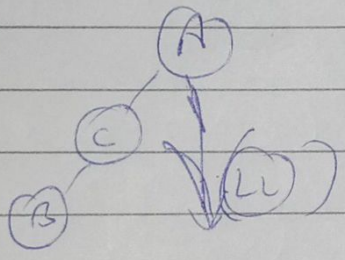
. LR (RR+LL)

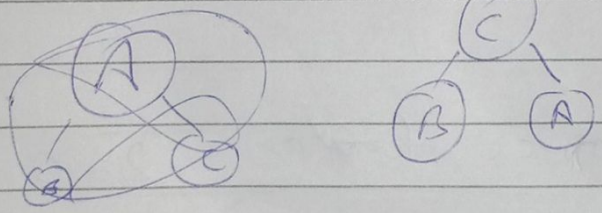


Balance factor = 2-0 ⇒ 2

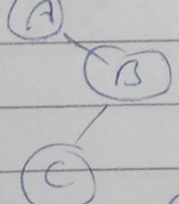
Balance ⇒





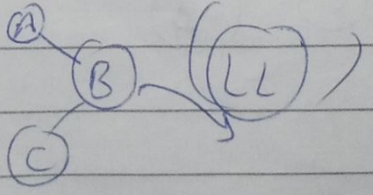


. RL (LL+RR)



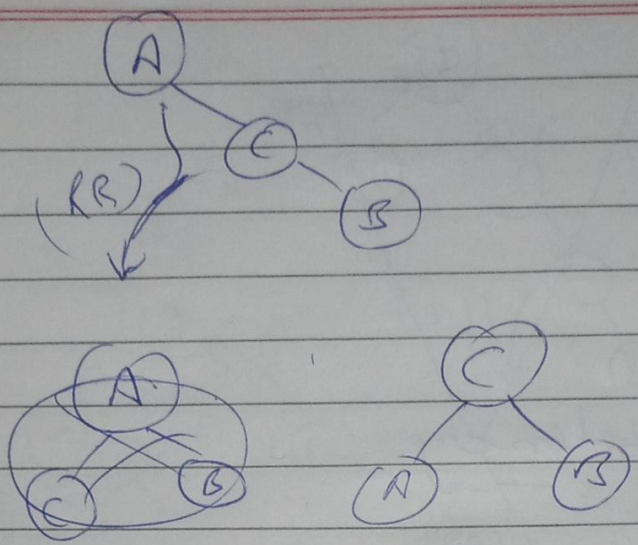
Balance factor = 0-2 ⇒ -2

Balance →



Good Write

DATE : ___/___/_____
PAGE : _____



Q To construct the AVL tree

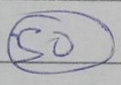
50, 20, 60, 10, 8, 15, 32, 46, 11, 48,

Q. To construct AVL Tree

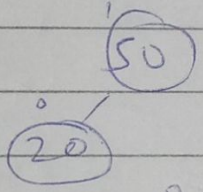
50, 20, 60, 10, 8, 15, 32, 46, 11, 48

Sol:

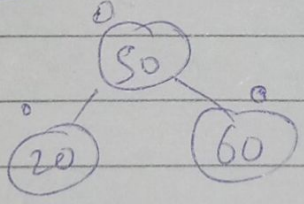
Step 1 →



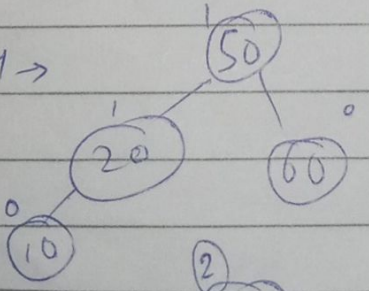
Step 2 →



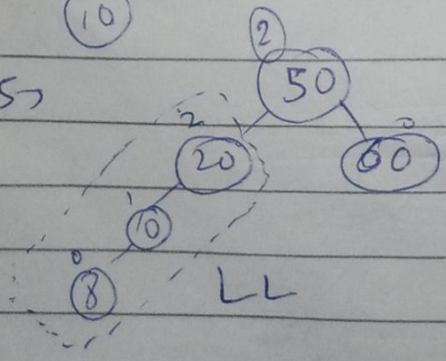
Step 3 →



Step 4 →



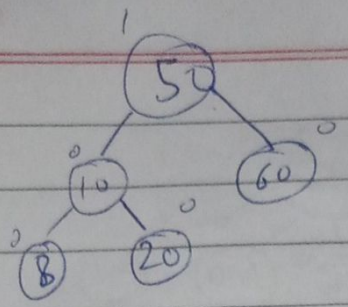
Step 5 →



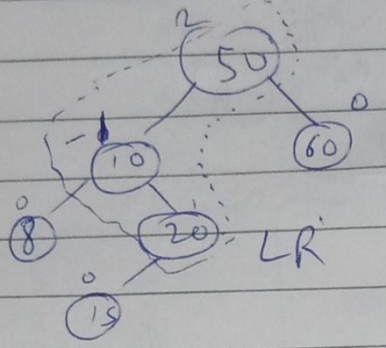
Good Write

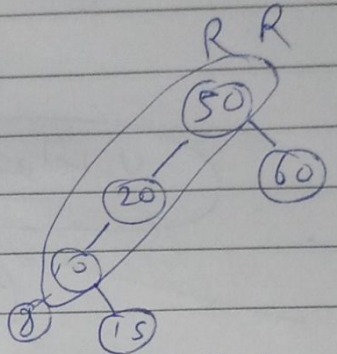
DATE : __/__/____
PAGE : _____

Step 6 →



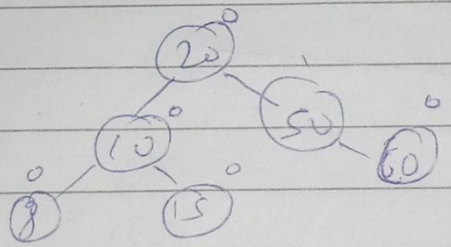
Step 7 →



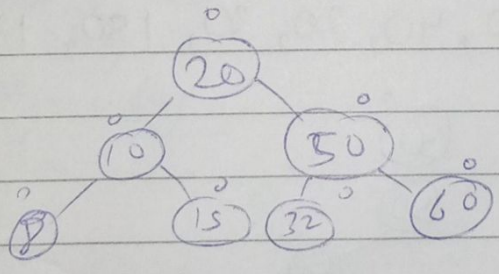


(as 15 is small to 20 & greater to 10)

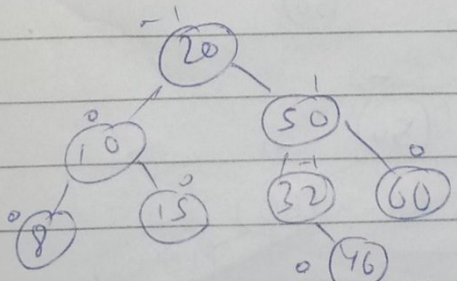
Now LL



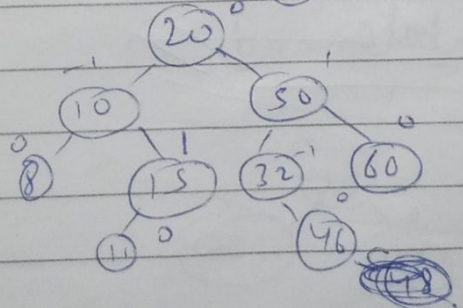
Step 8 →



Step 9 →



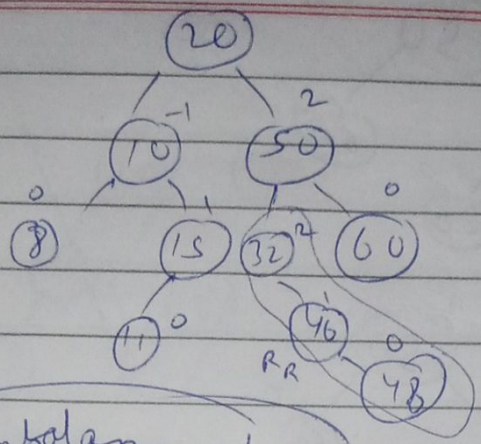
Step 10 →

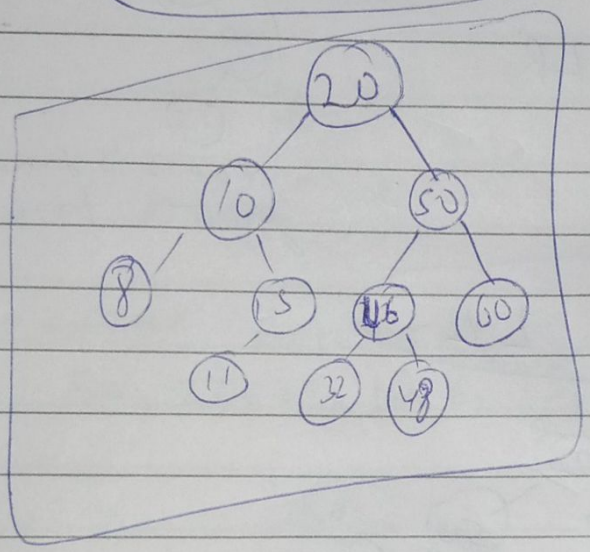


Good Write

DATE: __/__/____
PAGE: _____

Step 1 →



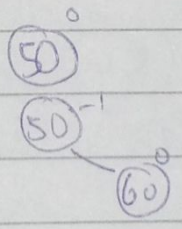


Ans

Q2 Construct a AVL tree

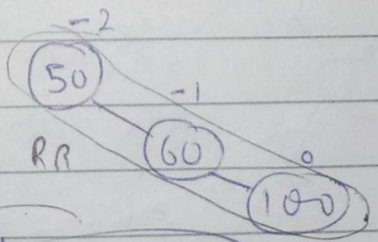
50, 60, 100, 40, 20, 30, 150, 110, 90, 80

Sol → Step 1 →

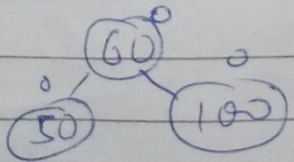


Step 2 →

Step 3 →



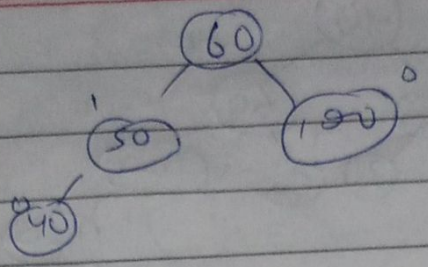
Unbalanced



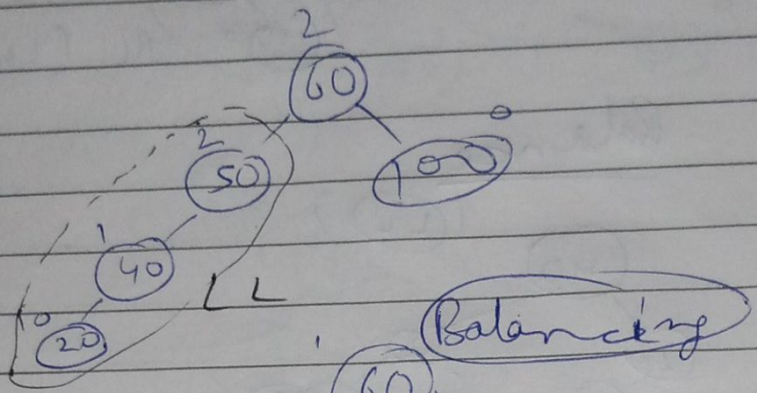
Good Write

DATE : __/__/____
PAGE : _____

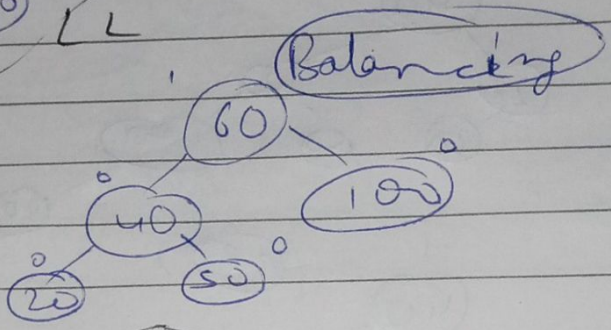
Step 4 →



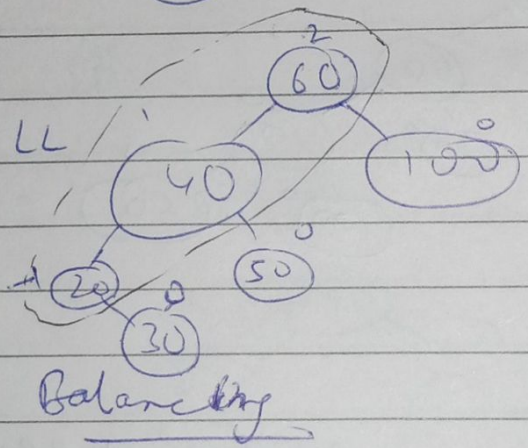
Step 5 →



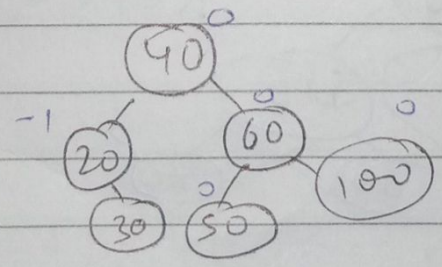
Step 6 →



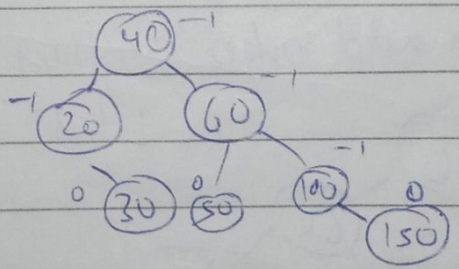
Step 6 →



Balancing

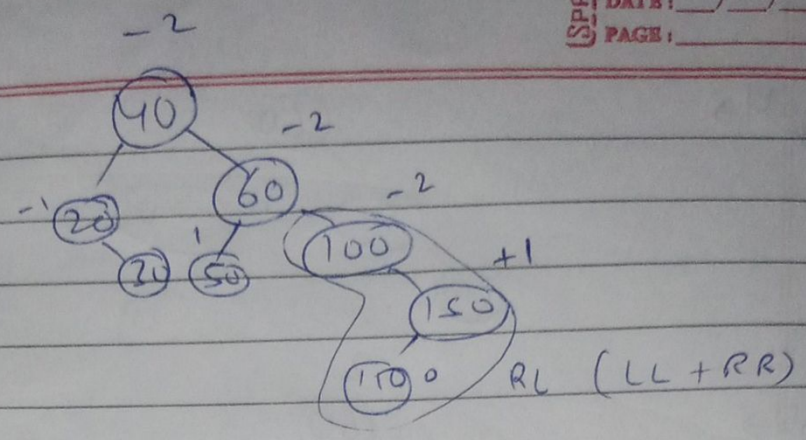


Step 7 →

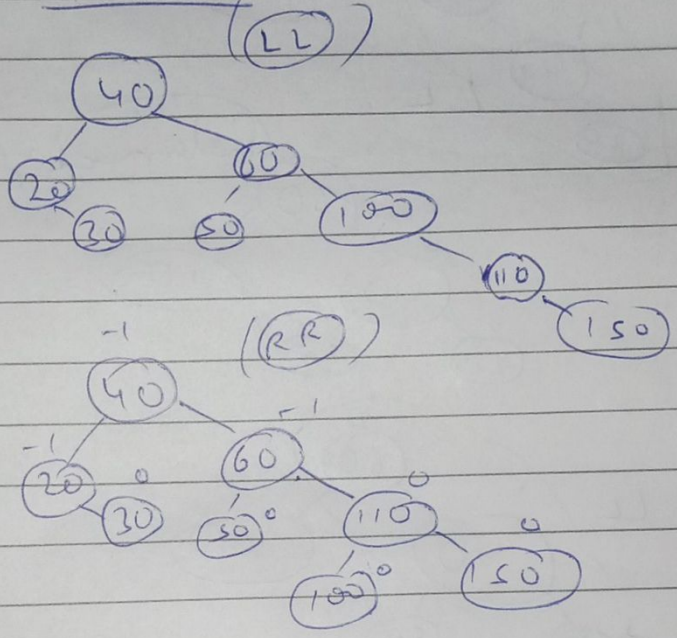


DATE: __/__/____
PAGE: _____

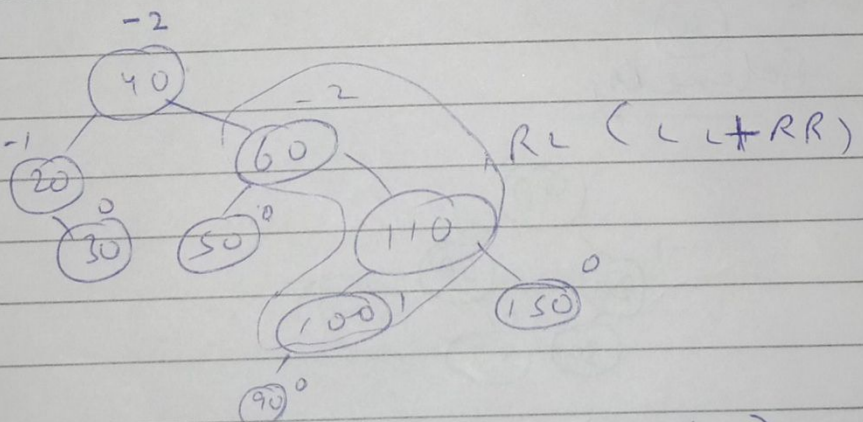
Step 8 →



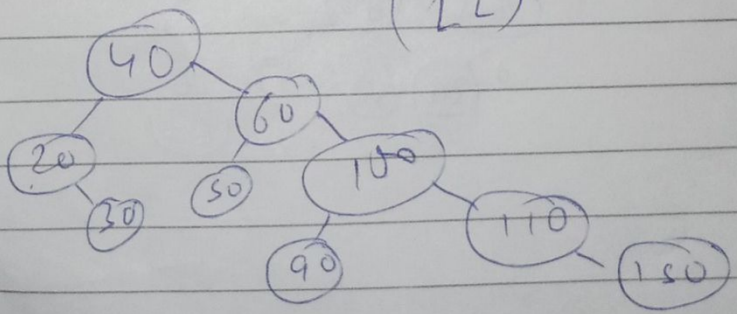
Balance



Step 9 →



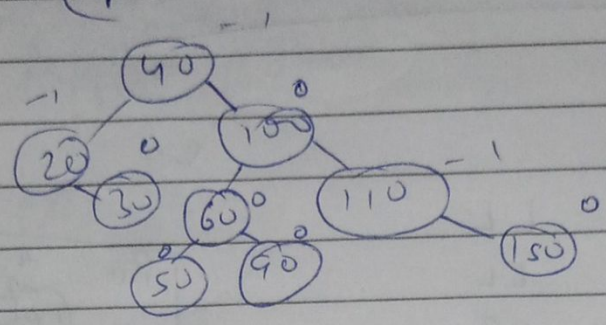
(Note → Jahan add usko consider)



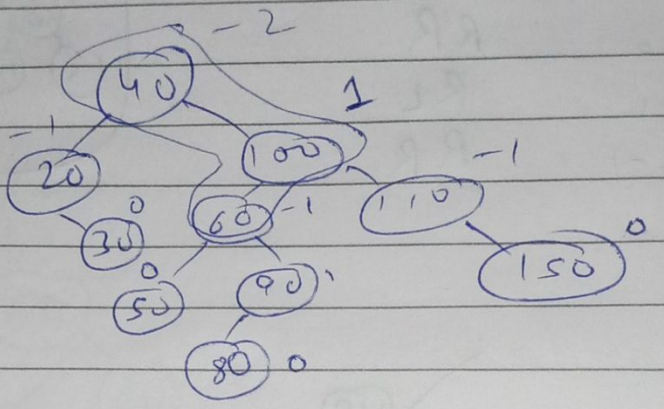
Good Write

DATE : __/__/____
PAGE : _____

(R R)

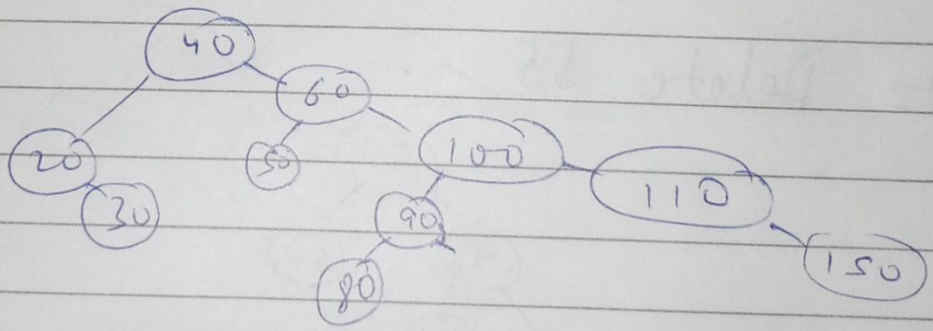


Step 1 →



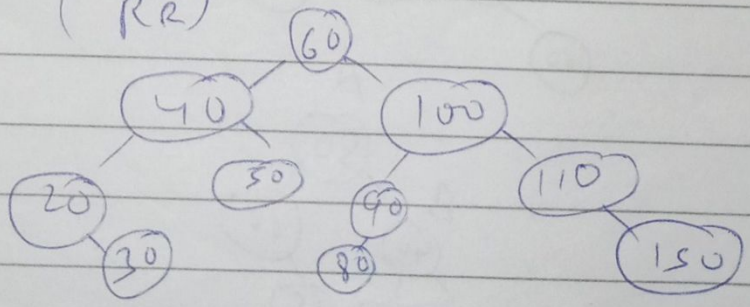
(R L) = LL + RR

((LL))



(RR)

Ans →

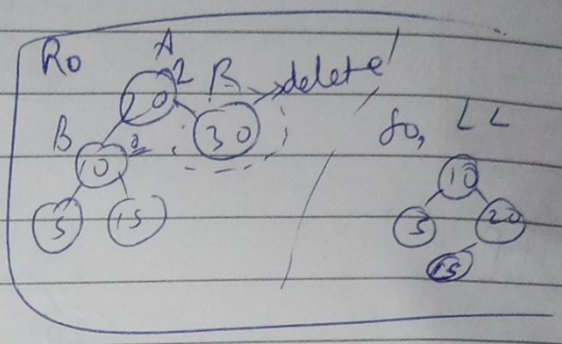


DATE: 16/10/23
PAGE:

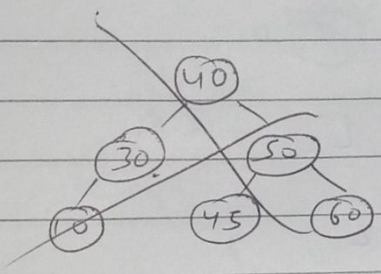
Deletion in AVL Tree

Table

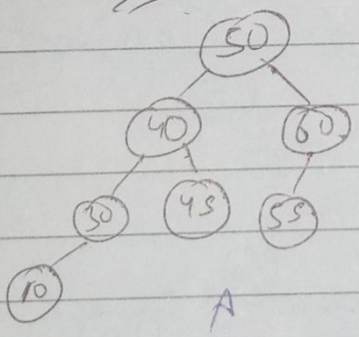
R_0	LL
R_1	LL
R_{-1}	LR
L_0	RR
L_1	RL
L_{-1}	RR



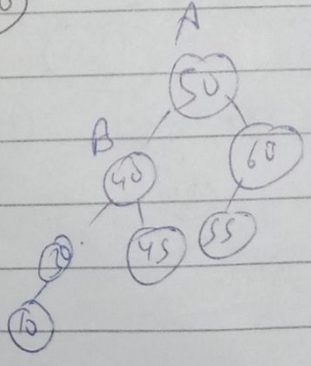
Ques→



Ques→ Delete 55 //.

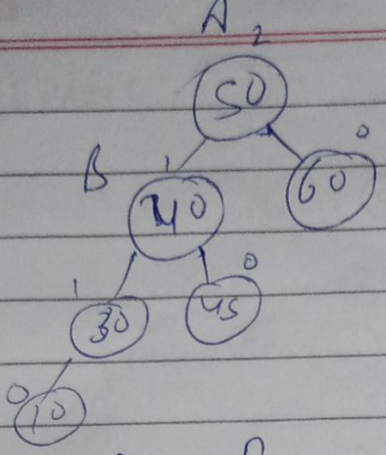


Sol→

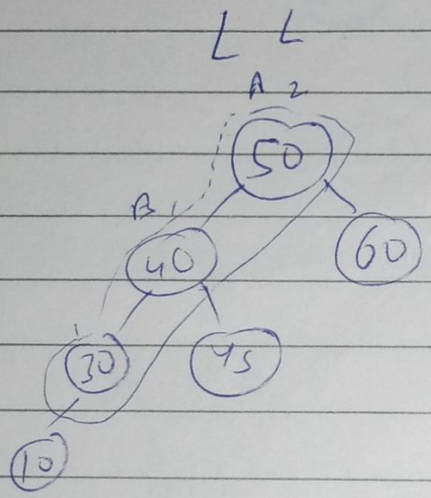


Good Write

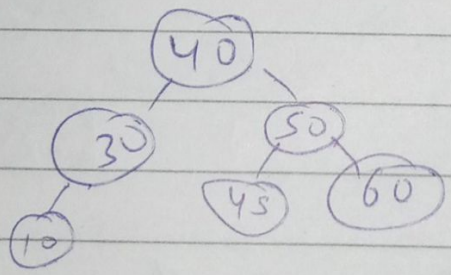
DATE : __/__/____
PAGE : _____



So, R1



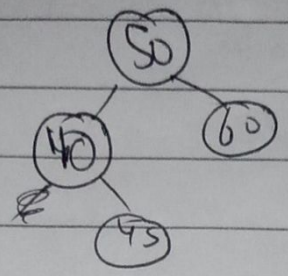
⇒



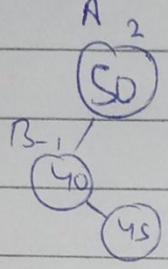
Ans

DATE : ___/___/____
PAGE : _____

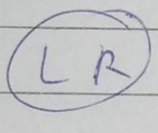
Q. Delete 60.



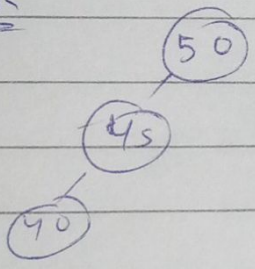
Sol →



R-1

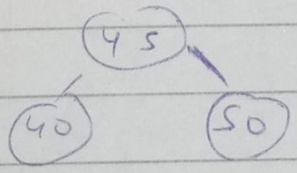


RR



then

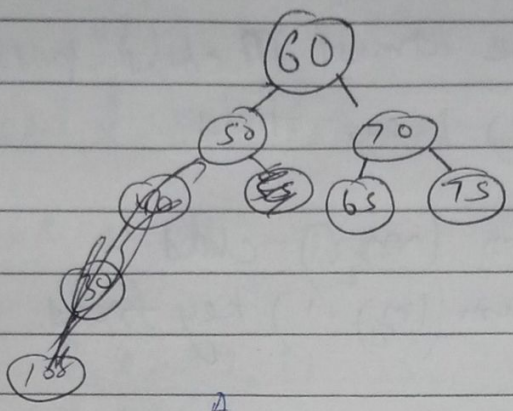
LL



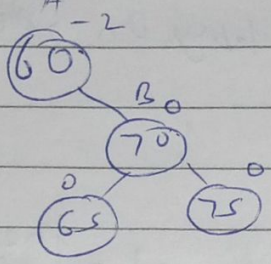
Ans

DATE : ___/___/_____
PAGE : _____

Q2 Delete (50)

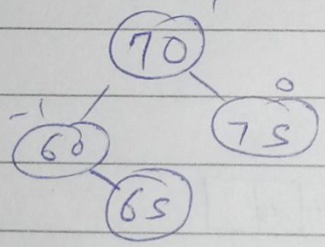


Q1 →



LO

So, RR



Ans

• m-way tree

↳ multi

→ These tree have atmost m child possible.

→ have max (m-1) key field.

(to order upper & value)

→ have minimum $(\frac{m}{2})$ child.

→ have minimum $((\frac{m}{2})-1)$ key field.

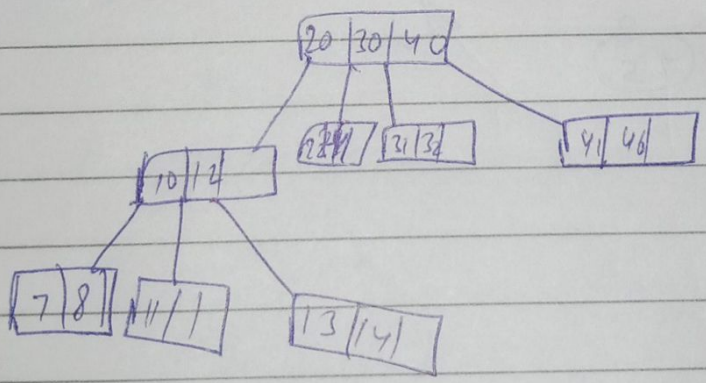
Condition : Doesnot apply on Root

Keyfield

It indicates the values of the given node that are used to improve the searching process.

Eg:

m = 4



· B - Tree
(Balance Tree)

→ Concept taken from m-way tree.

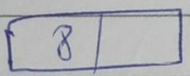
Q Construct a B-Tree having values

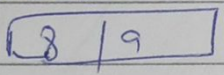
8, 9, 10, 11, 15, 20, 17

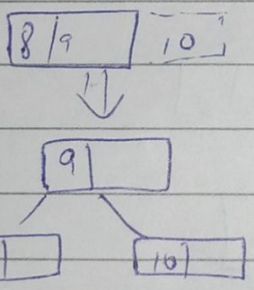
The value of m is 3.

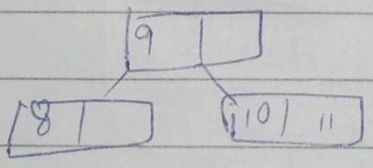
Sol→ Child → max = 3
min = 2

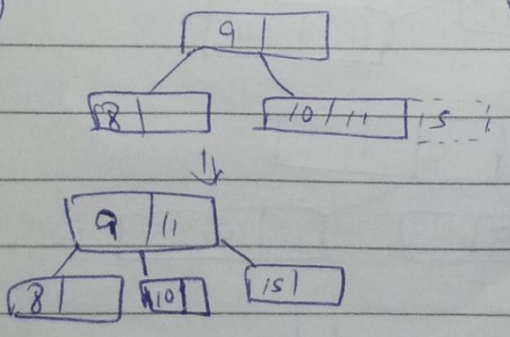
key field → max = 2
min = 1

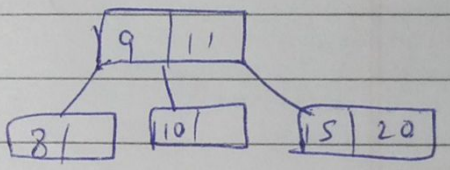
1) 

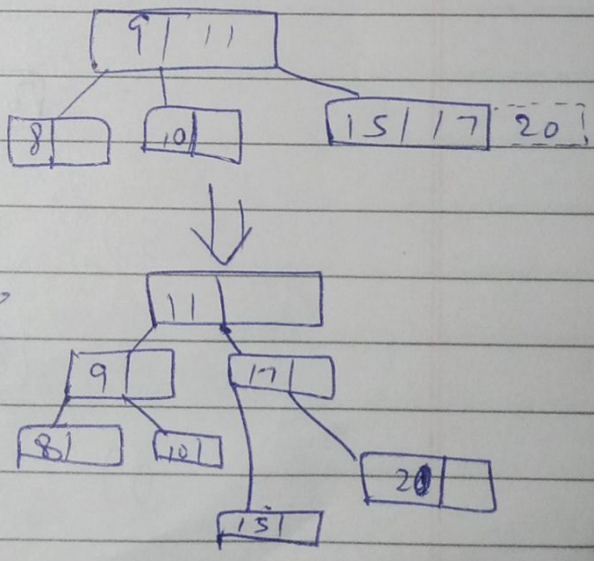
2) 

3) 

4) 

5) 

6) 

7) 

8)

DATE: __/__/____
PAGE: _____

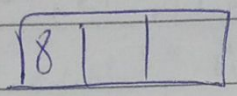
Q. Construct a Balance tree having values

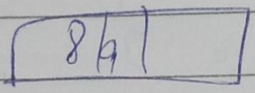
8, 9, 10, 11, 15, 20, 17

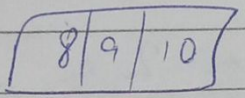
Value of m is 4. (Note: it has two left bias B-tree & Right bias B-tree)

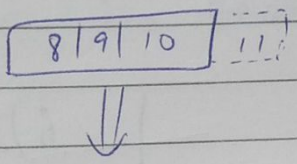
Sol → Child → Max = 4 Key field → Max = 3

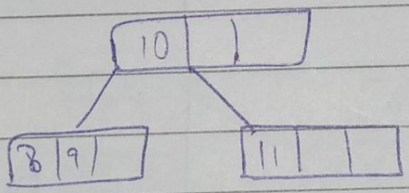
Min = 2 Min = 1

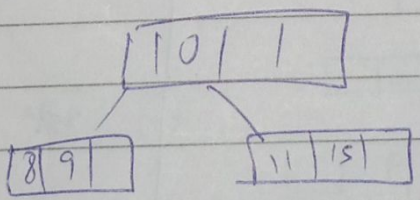
1) 

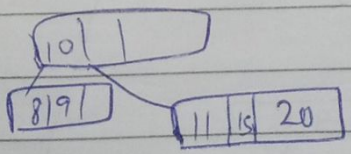
2) 

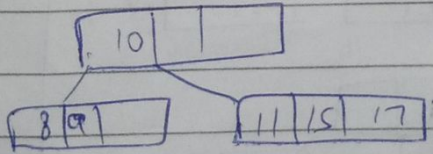
3) 

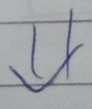
4) 



5) 

6) 

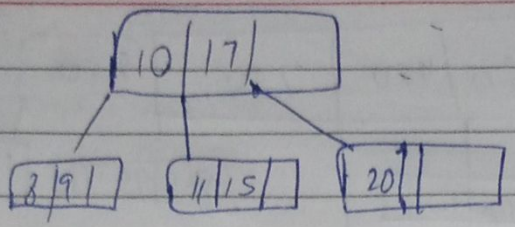
7) 



Good Write

Right Bias B-tree

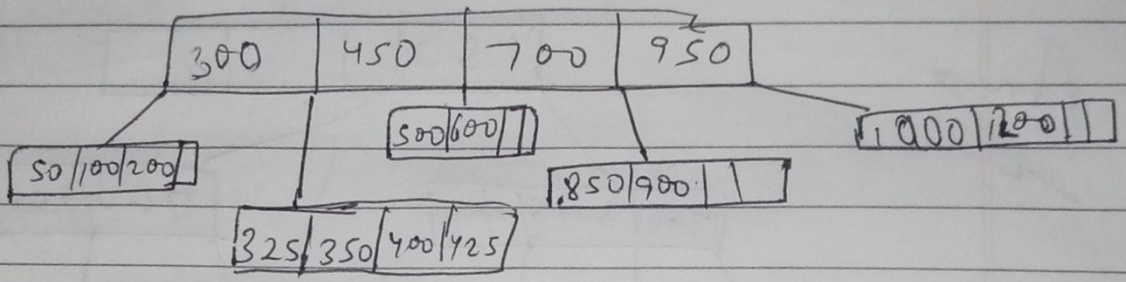
DATE: __/__/____
PAGE: _____



Ans

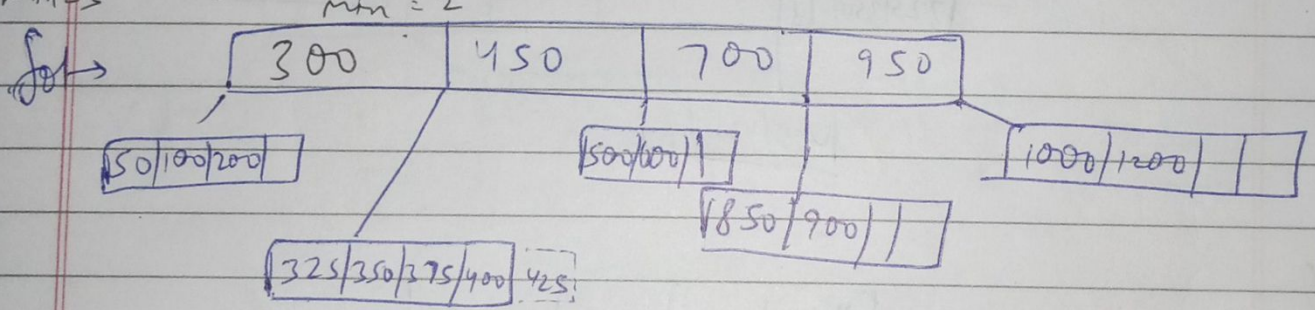
19/10/23

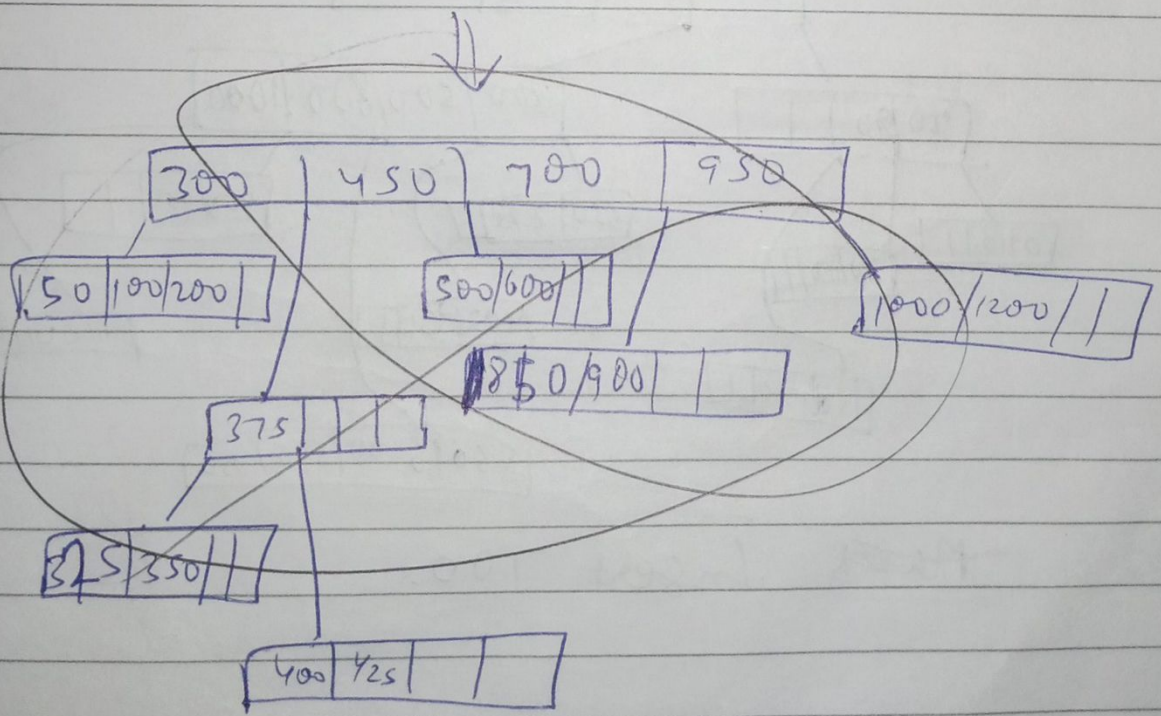
Q $m = 5$

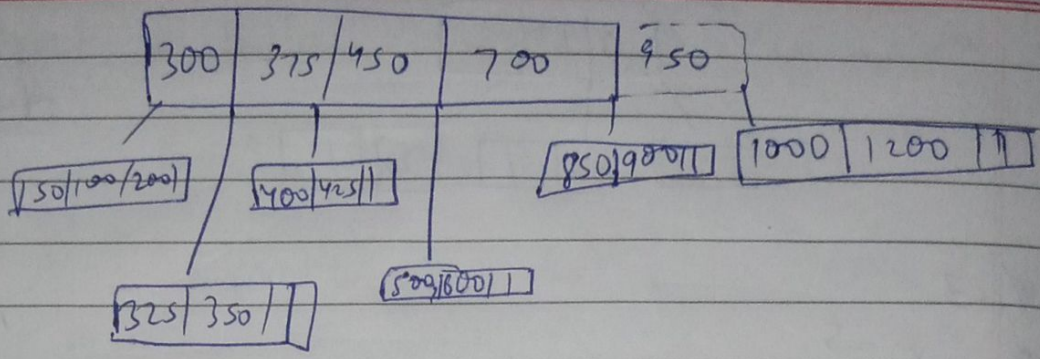


Insert 375.

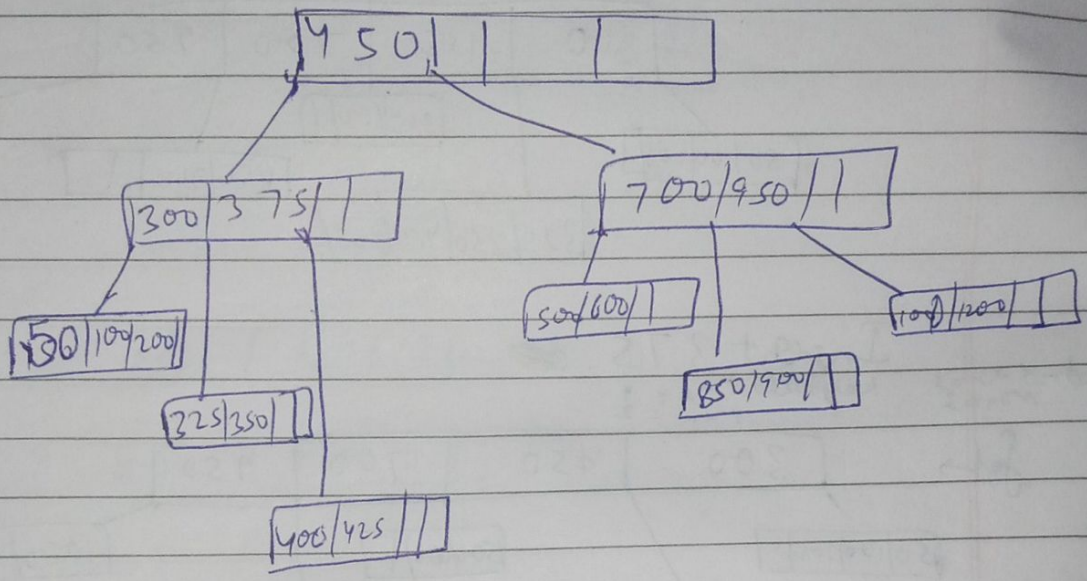
child = max = 5, min = 3
key field: max = 4, min = 2



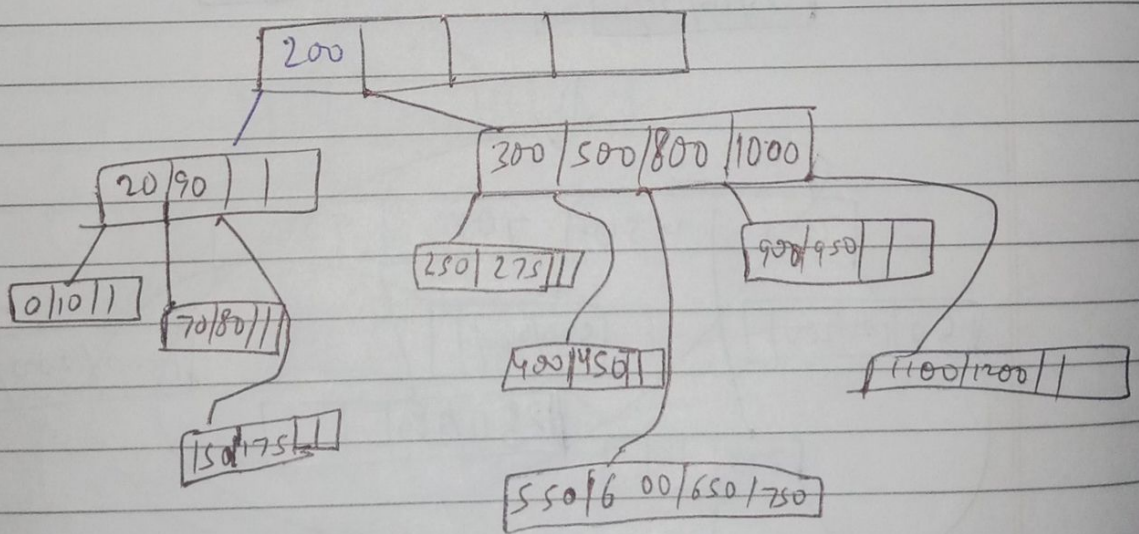




Ans →



Q



~~Mark~~ Insert 700.

Ans → m = 5

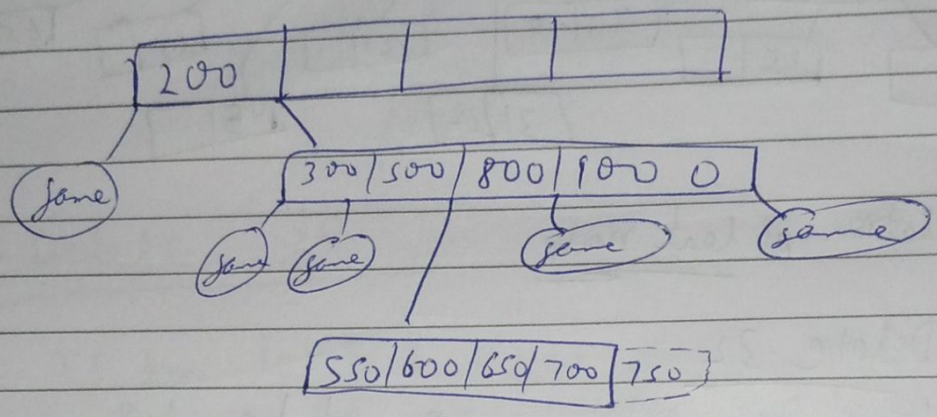
Keyfield = max = 4

min = 2

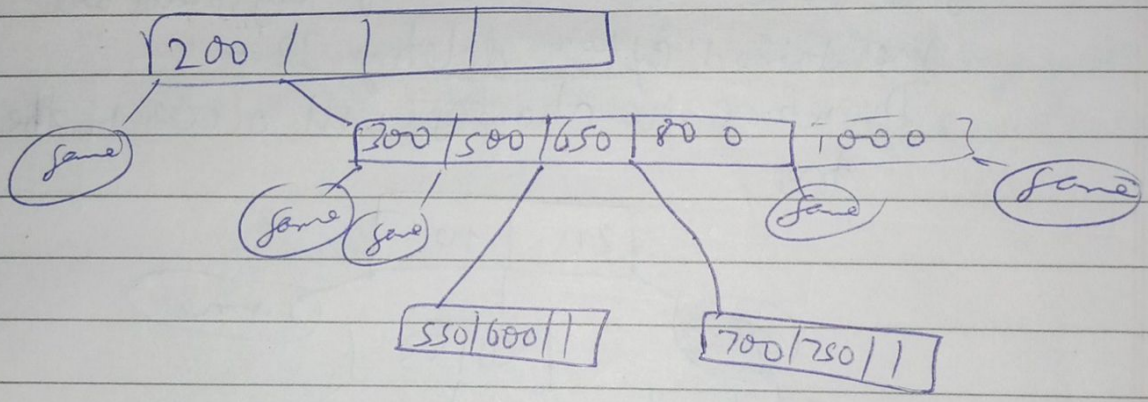
child = max = 5

min = 3

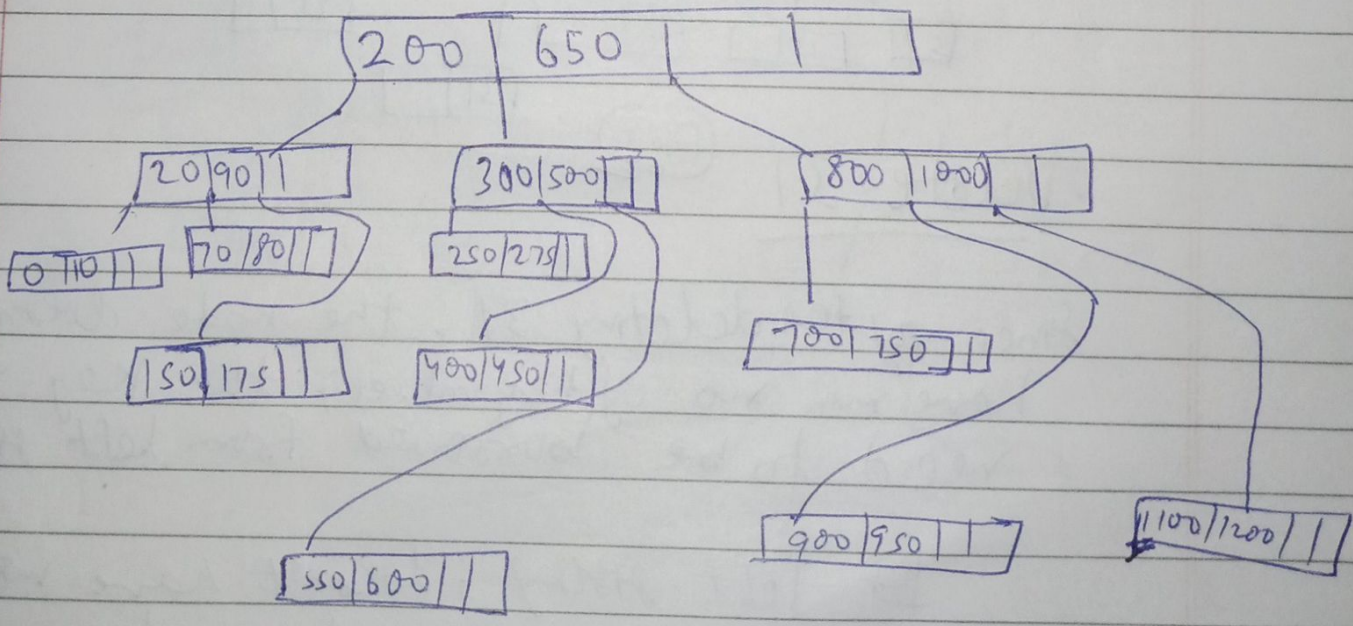
⇒



⇒



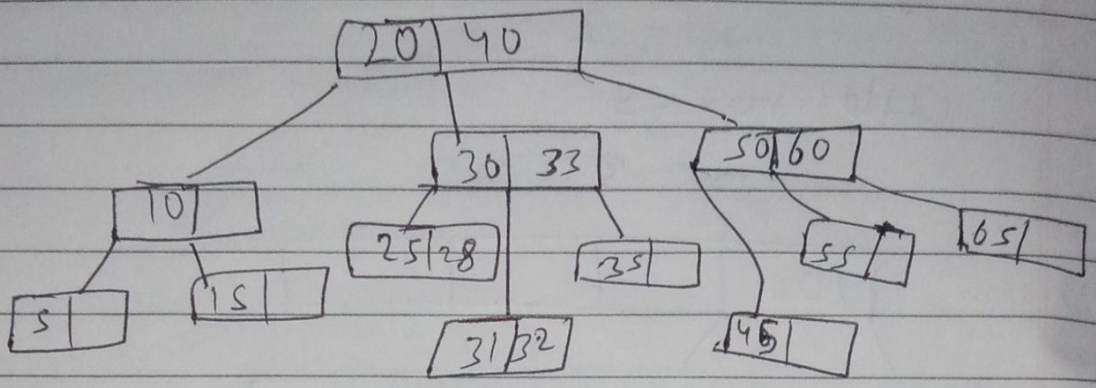
Ans →



Good Write

DATE: __/__/____
PAGE: _____

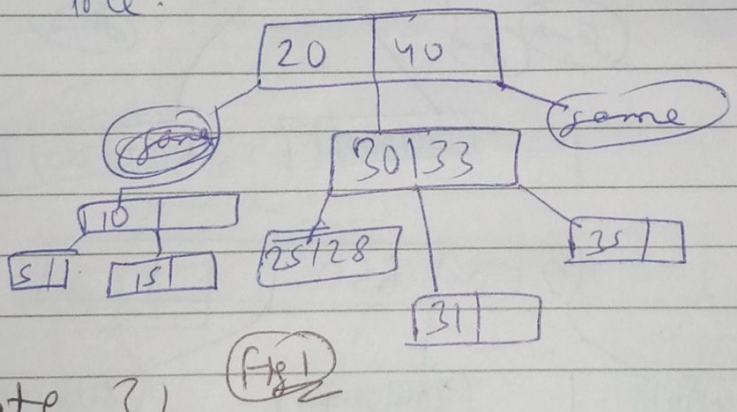
• Deletion in B-Tree



→ Deletion of leaf node

• Delete 32

Since the minimum no. of keyvalues are maintained after deleting 32.
Therefore; no changes will occur in the tree.

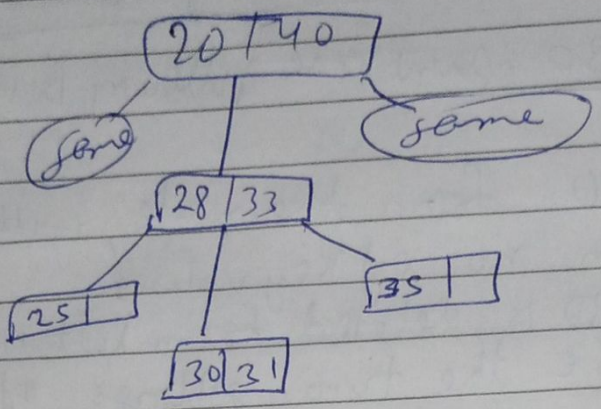


• Delete 31

Since after deleting 31; the node doesn't have min no. of keyvalues; the key value need to be borrowed from left sibling.

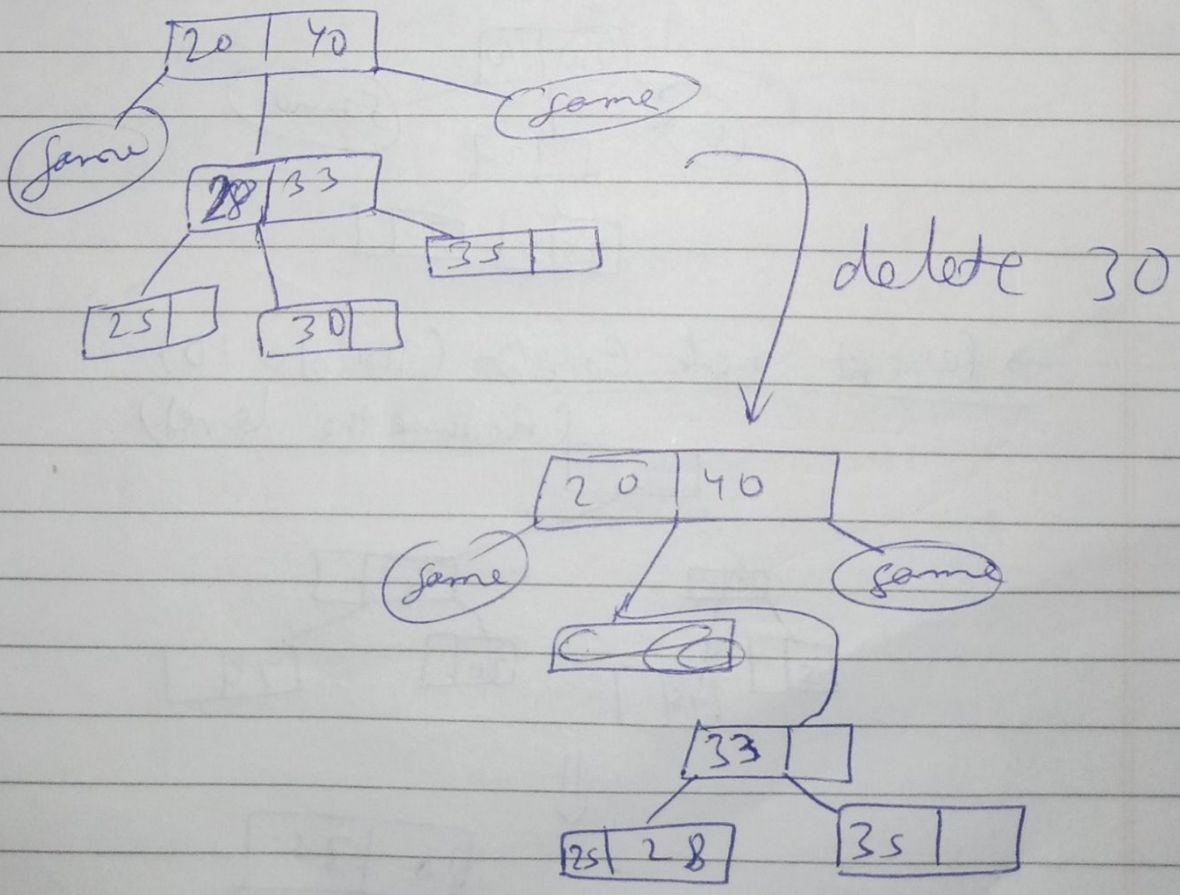
If left sibling; doesn't have more than min no. of keyvalues; then keyvalue need to be borrowed from right sibling.

Good Write



- Delete 30

If the left & right sibling doesn't have sufficient no. of extra key values then merging of the left child along with parent or ~~left~~ child with parent take place right

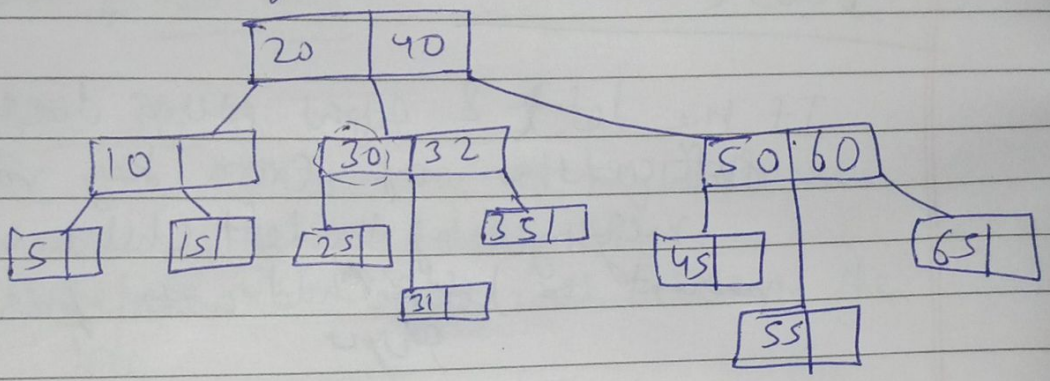


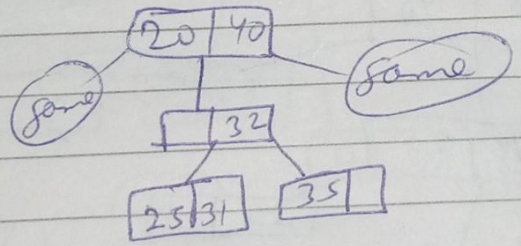
DATE: __/__/____
PAGE: _____

→ Deletion of internal node

Delete 30 (parent node following B-way Rule)

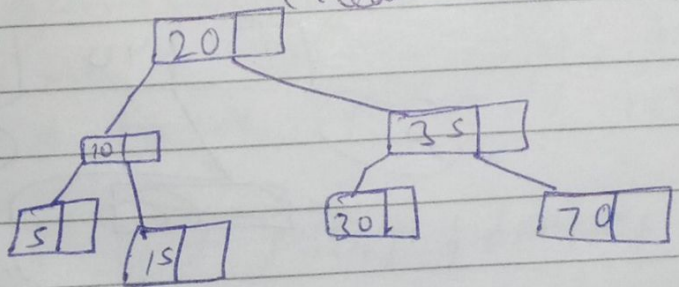
Delete 30 from the node; still maintains minimum no. of key values.
Since 30 is deleted from left side of node; therefore the two siblings of the left of 32 will merge together.

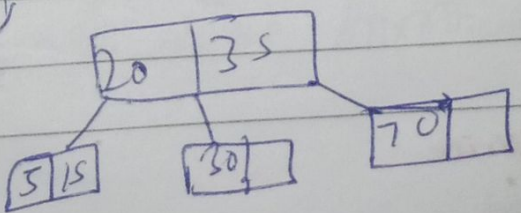




→ Parent node empty (Delete 10)

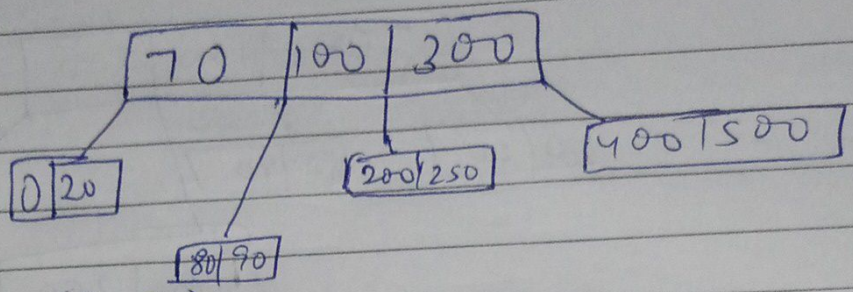
(Reduce the level)





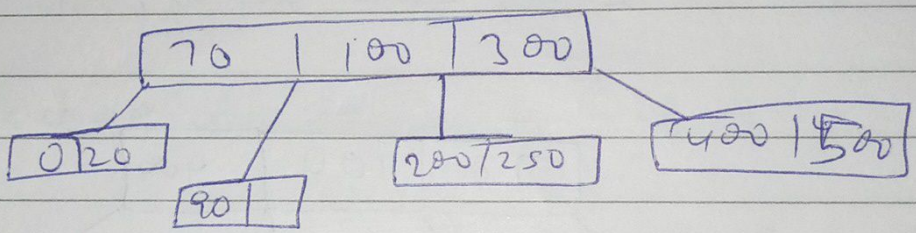
Good Write

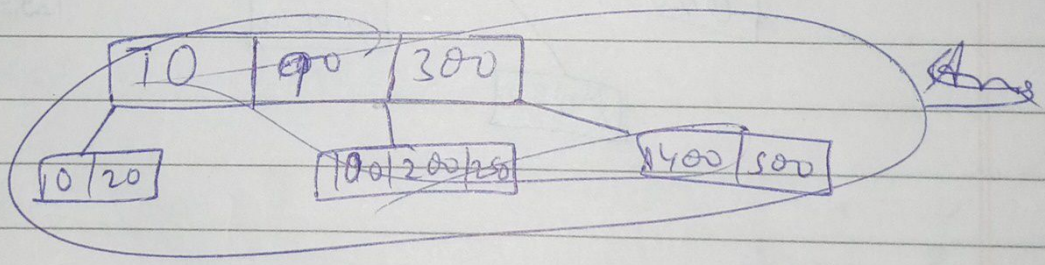
Q

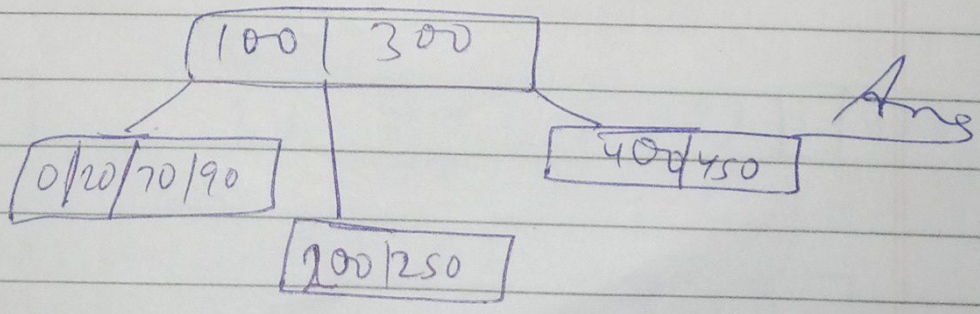


m=5 (Given)
Delete 80 from this tree.

Sol→ m = 5
Max key field = 4
Min " = 2
Child. Max = 5
Min = 3



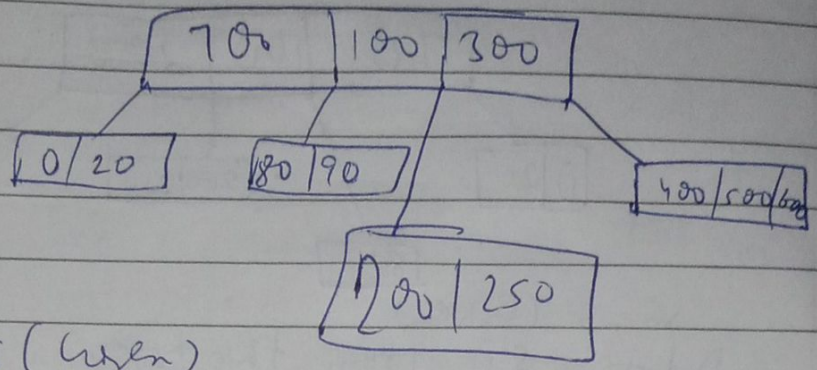




Good Write

Case 2 Leaf node

Q



m = 5 (given)

Delete 250.

Sol → m = 5

Keyfield

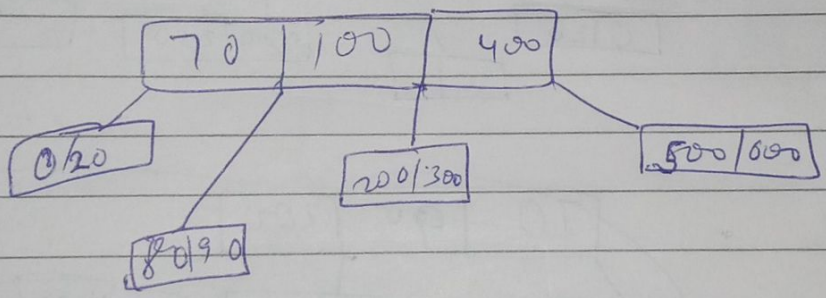
Max = 4

Min = 2

Child

Max = 5

Min = 3



Ans

B+ Tree

DATE: __/__/____
PAGE: _____

→ It is extension of B Tree for improved searching process (more efficient searching)
↳ Decrease Time taken to search

→ Note: B Tree → No duplicate values in tree
B+ Tree → Duplicate // // // are also there.

→ Deletion is simpler in this as compared to B tree.

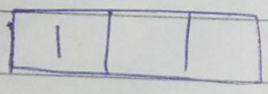
→ All values are in leaf node.

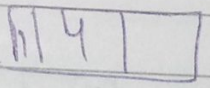
Note → Rather All values are same as B-tree.

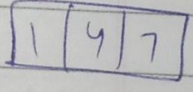
Q. 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42
& m=4 (given)
Construct B+ tree.

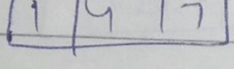
Sol → m = 4
keyfield → Max : 3
Min : 1

Child → Max = 4
Min = 2

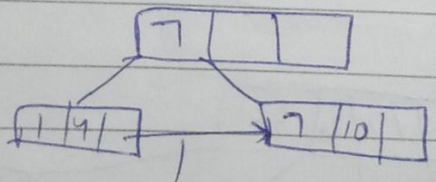
1) 

2) 

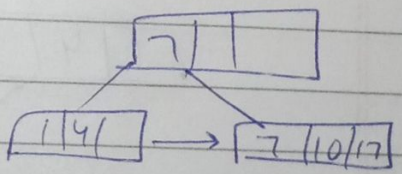
3) 

4) 

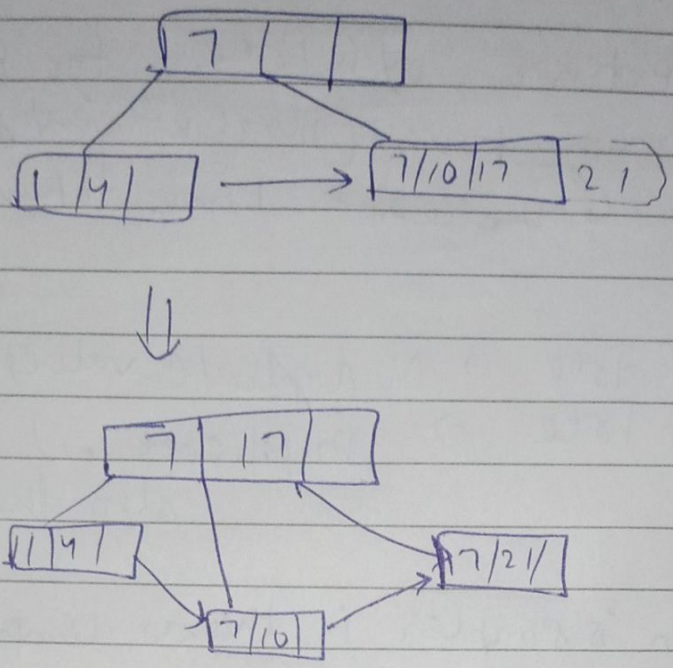
(Right Biasing)

5) 

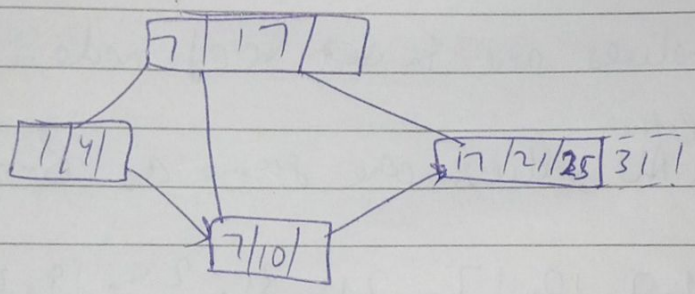
Note: Singly linked list of leaf nodes connect

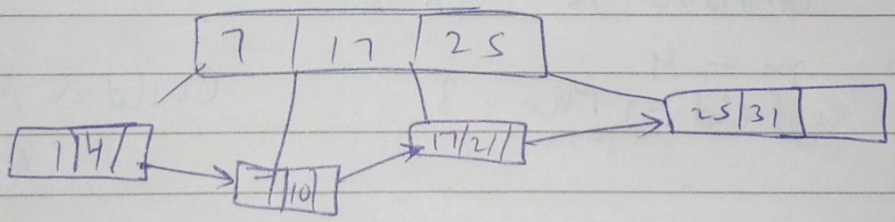
6) 

7)

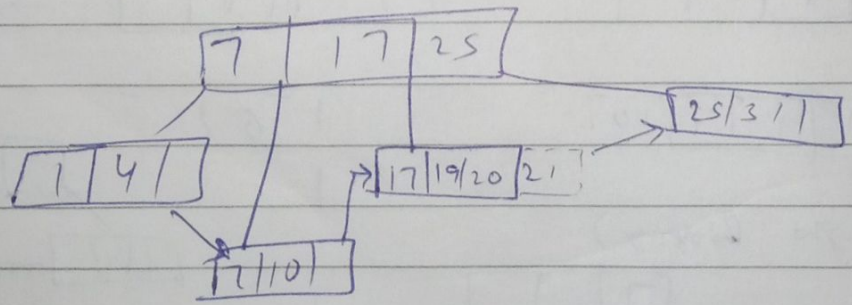

7
1 4
7 10 17 21
7 17
1 4
7 21
7 10

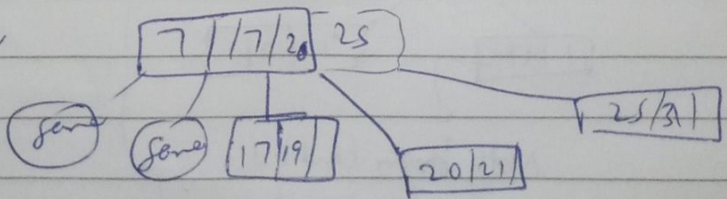
8)


7 17
1 4
17 21 25 31
7 10


7 17 25
1 4
25 31
17 21
7 10

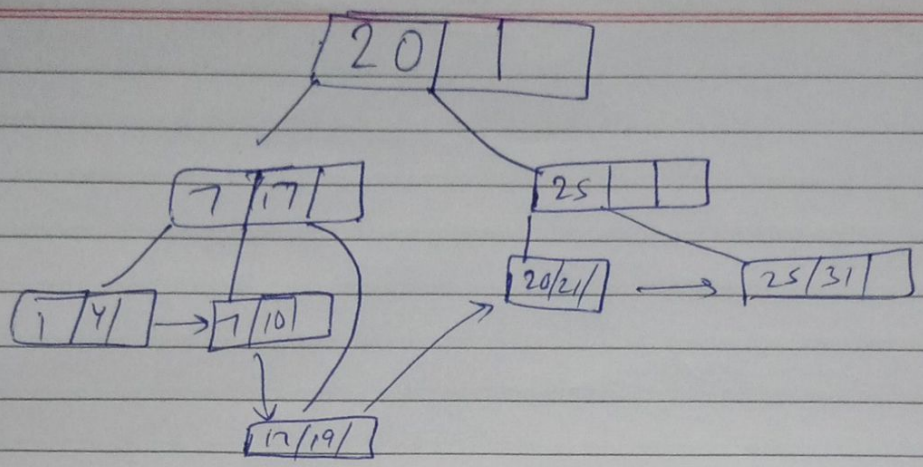
9)


7 17 25
25 31
1 4
17 19 20 21
7 10

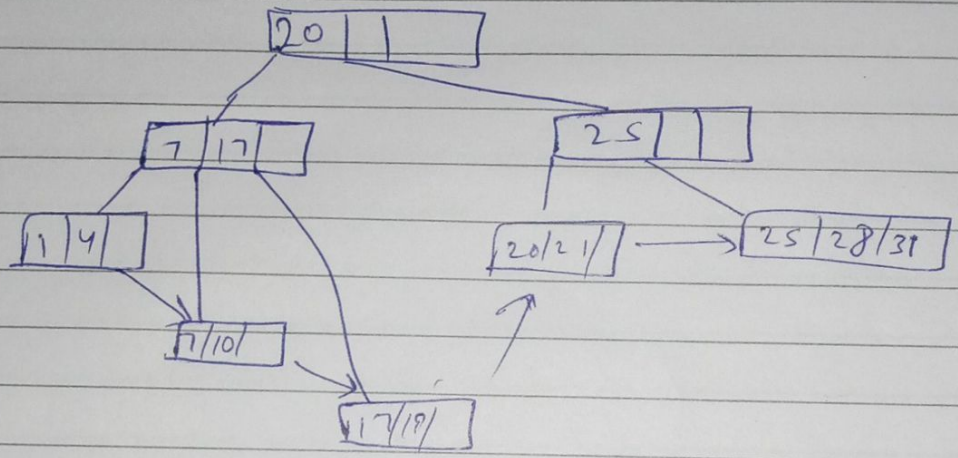

7 17 20 25
25 31
17 19
20 21

(Note: node may go upwill not duplicate in internal node case.)

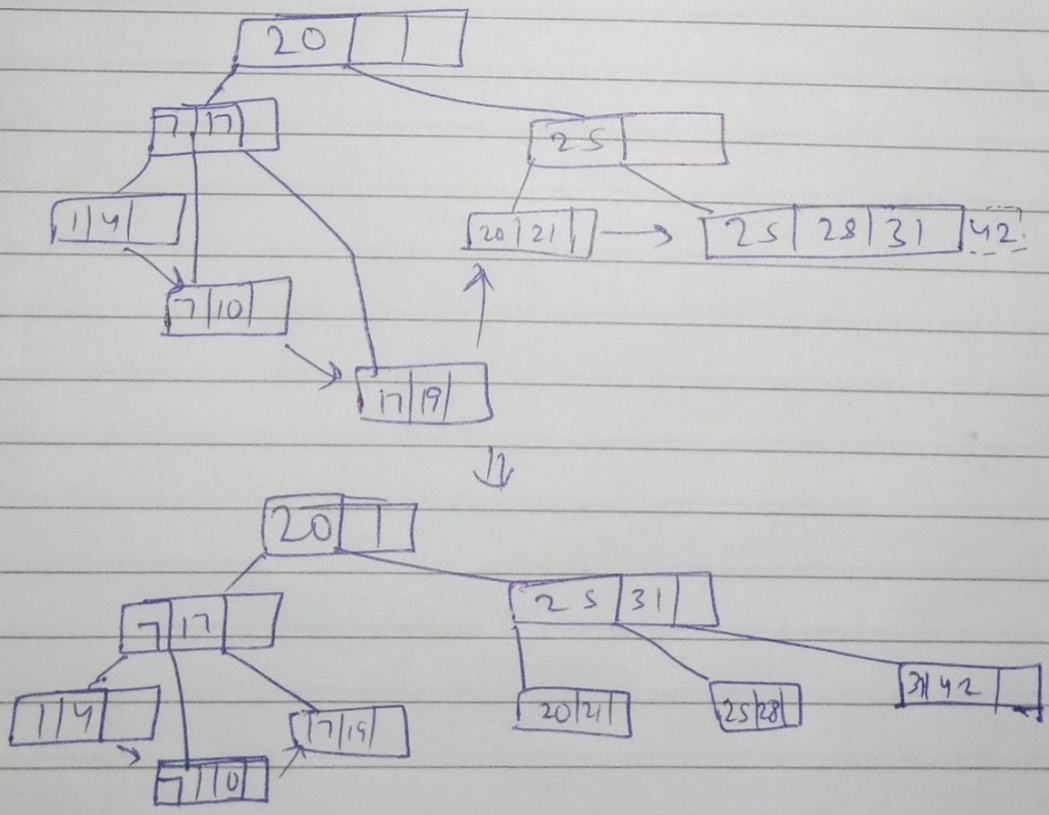
DATE: __/__/____
PAGE: _____



10



11



Good Write