5.. In B-tree the structure of Internal node and leaf node are ~~exact~~ same. B-tree with order 4 means, maximum number of node pointer in a node is 4 and maximum number of search key in a node is 3.

The construction of B-tree for given search keys ~~are~~ is as follows.

insert → 12   [| 12 | | | | |]

Insert 14   [| 12 | | 14 | | |]

Insert 7   [| 12 | | 14 | | 7 |]

↓ in sorted order

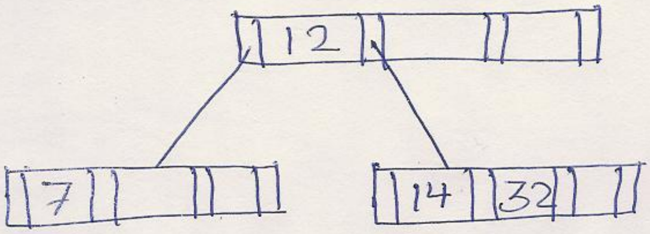
[| 7 | | 12 | | 14 |]

Insert 32   The node can not contain more than 3 search key. So, the node will have to be splited. The node can be splitted in two ways namely left baising and right baising. Depending on this there two possibility of tree to exist. Both are correct. Once a method used for splitting we will have to use the same when ever we will split a node. Here we are using right baising.

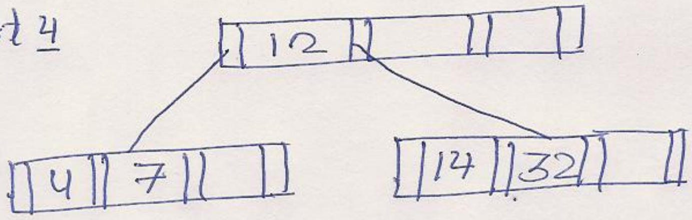
The search key in sorted order are

7, 12, 14, 32.

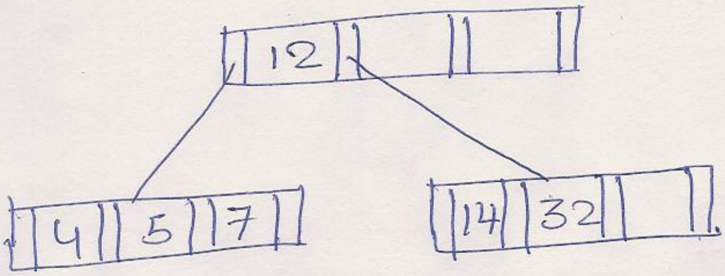
According to right baised splitting 12 will be the root with 7 in left node and 14 and 32 in right child node.



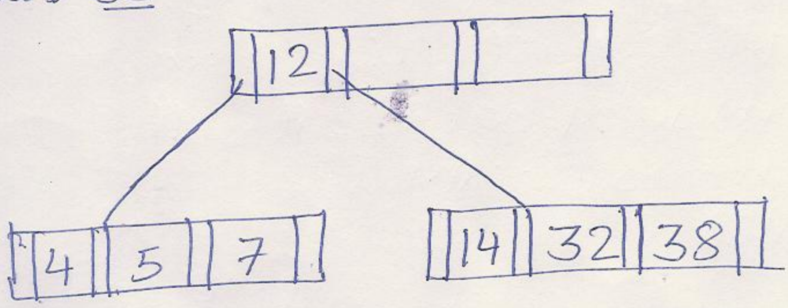
Insert 4



Insert 5



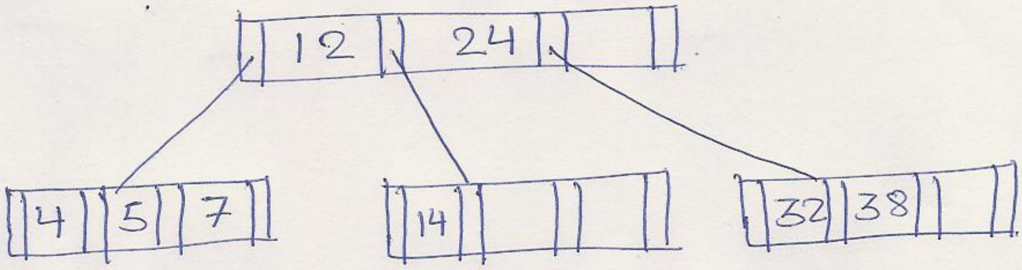
Insert 38



Insert 24

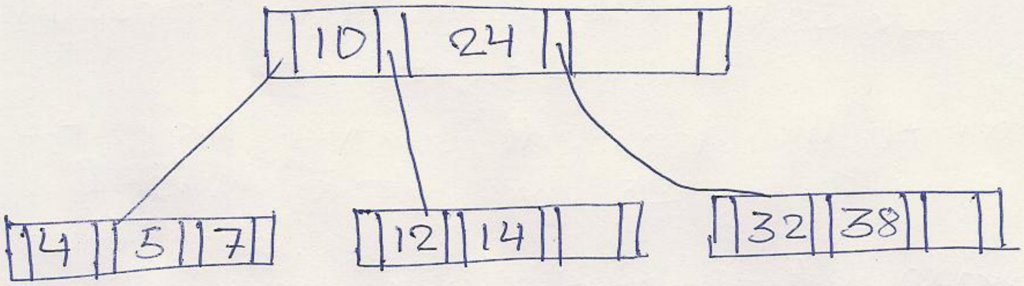
24 will be inserted into right child node where capacity of node is full and no siblings are having space so the node will have to be splitted. The splitting will be done using right basing method because we have used the same method for splitting the node once.

The search keys in sorted ordere are
14, 24, 32, 38.
24 will become the root with 14 in left child node and 32 and 38 in right child node.
Here 24 is being sent to the above node. At root node search keys in sorted orded are 7, 24.



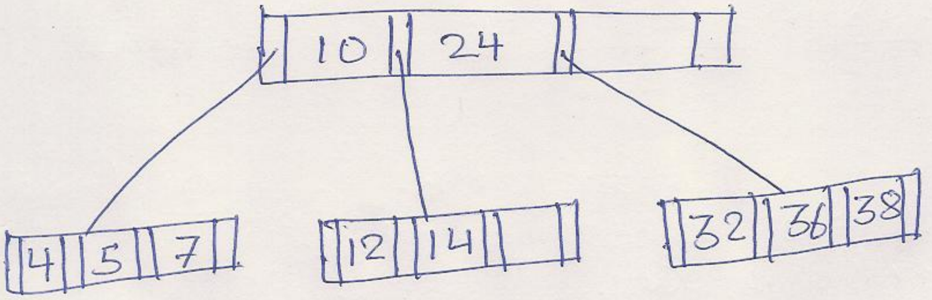
Insert 10

10 will be inserted in first left child node because 10<12. The node is full. We should not go for node splitting till we have space available in sibling. Since in our case its sibling can accomodate two more search keys, we should go for key-redistribution.
The search keys in sorted order are 4, 5, 7, 10. So, 10 will be redistributed to its sibling (right). In this process '10' will be sifted to root at the place of 12 and 12 will be sifted to its right child and the search keys should be in sorted order.

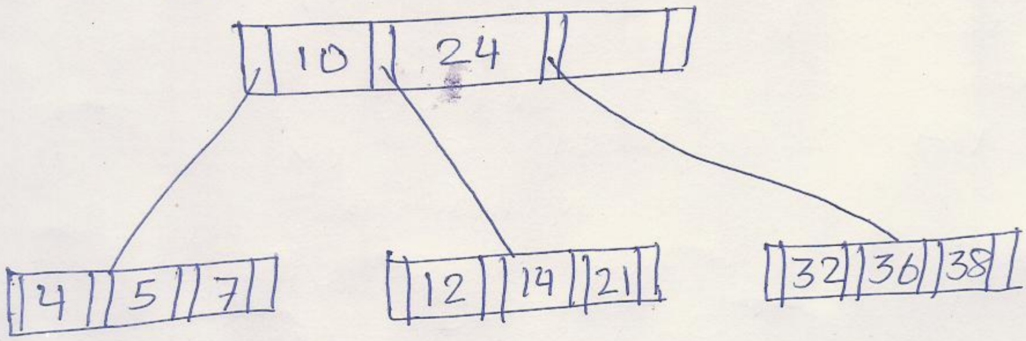


Insert 36

It will be inserted into right most child node (leaf) and the node will be sorted.



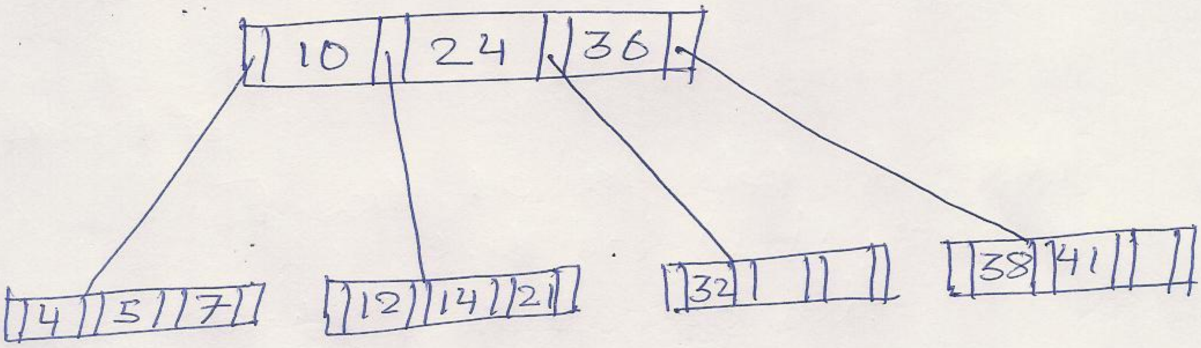
Insert. 21



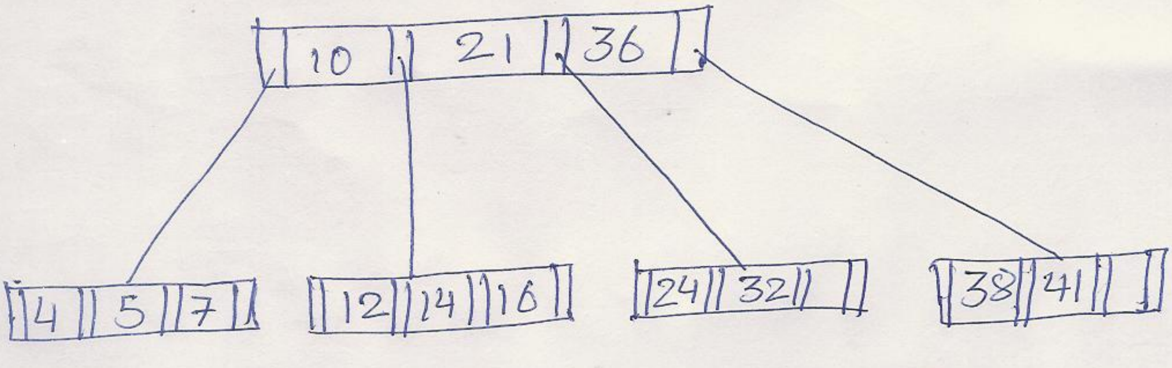
Insert 41

It will be inserted into right most child node. Since the node is full and also the siblings are full so, the node will have to be splitted.

The search keys in sorted order are
32, 36, 38, 41
The key '36' will become the root.

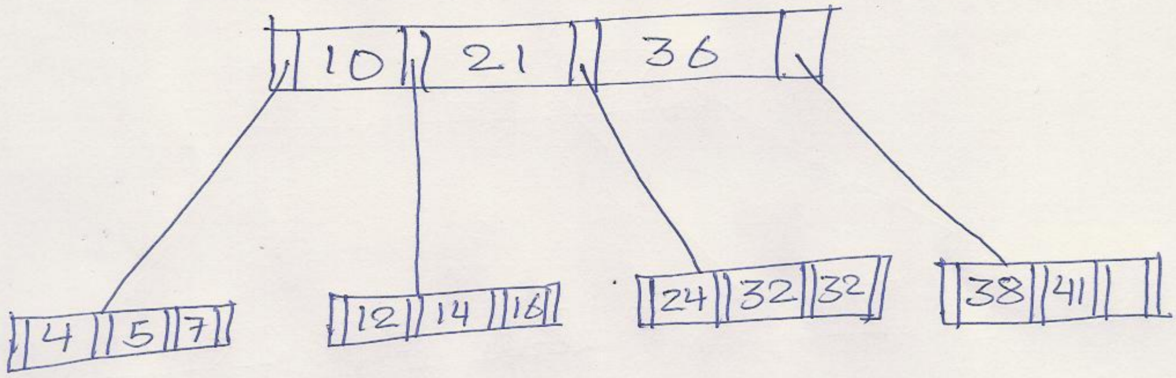


Insert <u>16</u>
It will be inserted in second leaf node because 16 > 10 and 16 < 24. The node is full but space is available in next sibling. So, we will go for key redistribution.
The search keys in sorted order are 12, 14, 16, <u>21</u>.
'21' will be shifted.



Insert 32

It will be inserted in 3rd leaf node because 32 < 36 and 32 > 21.



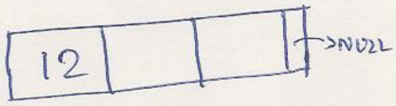
This is the final tree. In the expected answer the students should have drawn the tree after each step of insertion at least the documentation of node splitings and key redistributions.

Ans-no-6-> In B+-tree the leaf node has record pointers but the internal nodes do not have record pointer. B+-tree with order 4 can have maximum 4 node pointer and maximum 3 search keys.

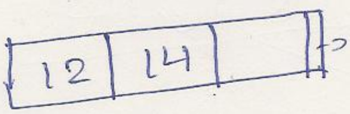
Multiple answers are possible depending on the method used in tree creation. methods can be left baising and right baising. The search key which becomes the root when node is splitted will have to be present in either of the child node. Only one method should be followed throughout the tree creation.

A sample tree creation is as follows.

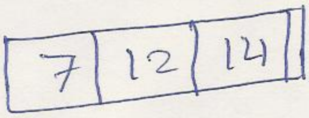
Insert 12



Insert 14

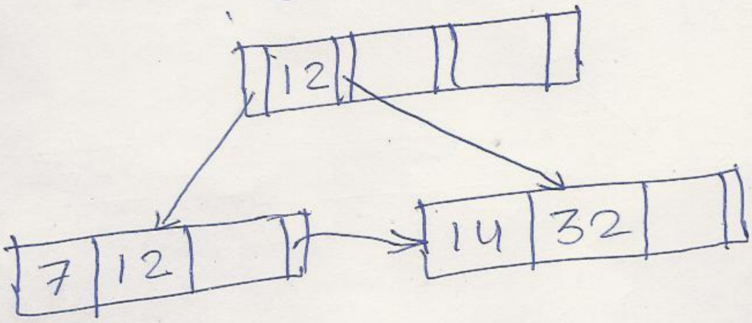


Insert 7



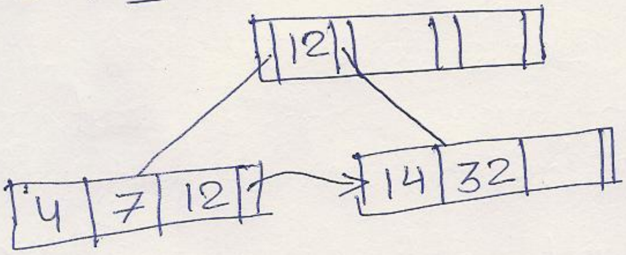
Insert 32 Right based node splitting.

7, 12, 14, 32



Insert 4

$4 < 12$

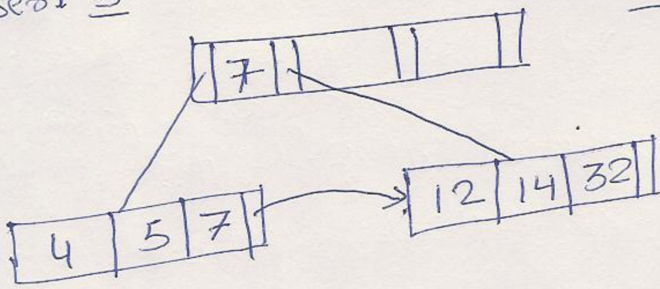


Insert 5

$5 < 12$

Key redistribution

4, 5, 7, 12,

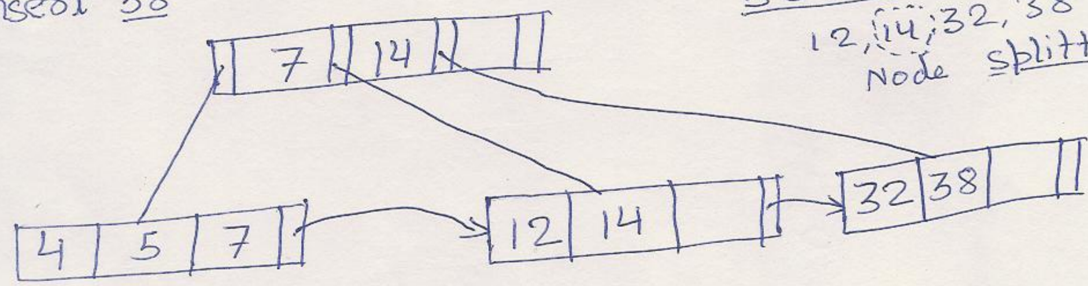


Insert 38

$38 > 7$

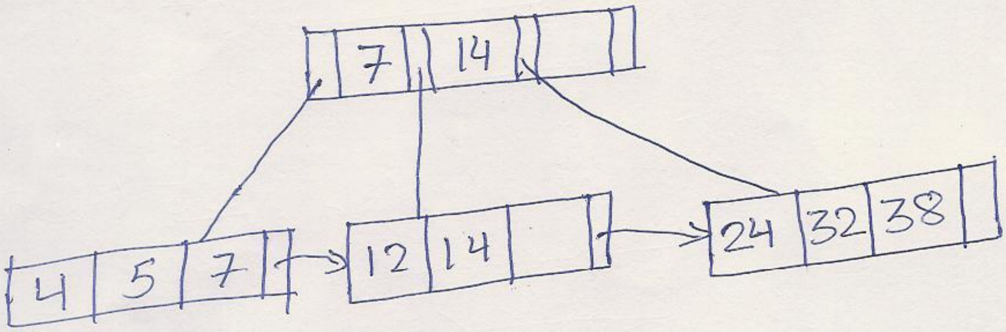
12, 14, 32, 38

Node splitting



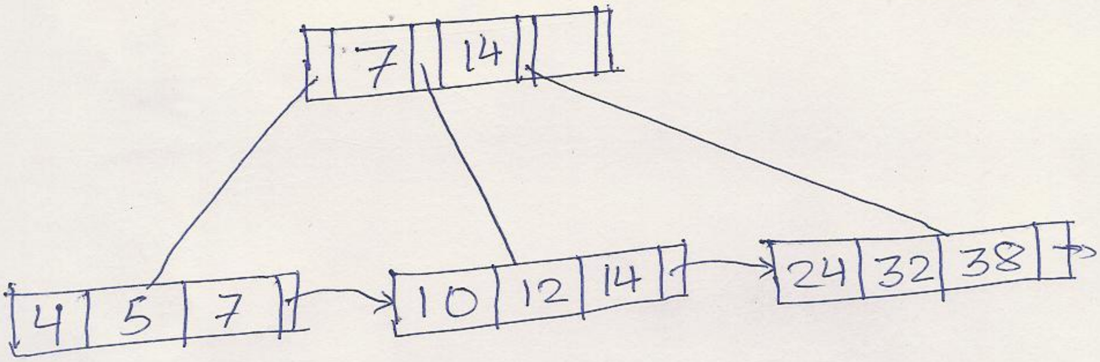
Insert 24

$24 > 14$



Insert 10

10 > 7 < 14

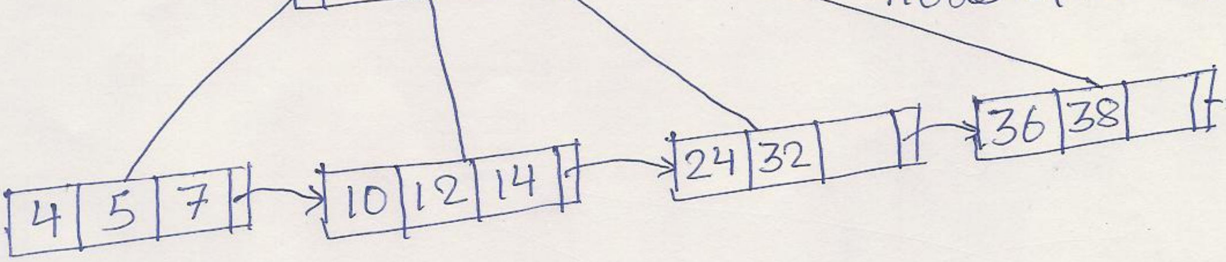


Insert 36

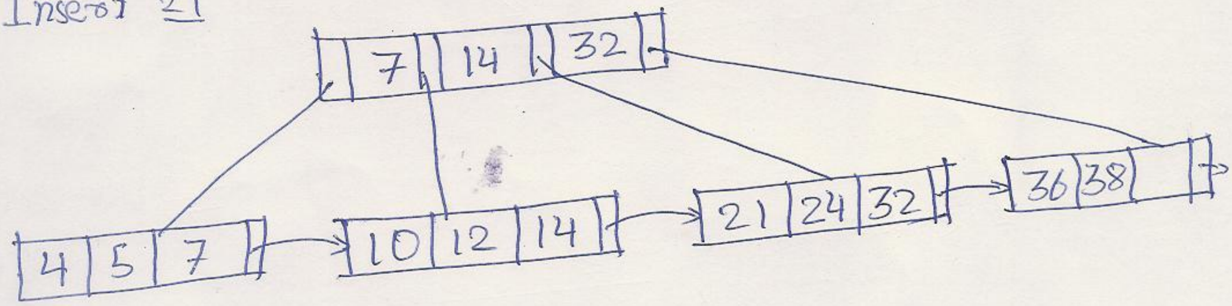
36 > 14

24, 32, 36, 38

node splitting

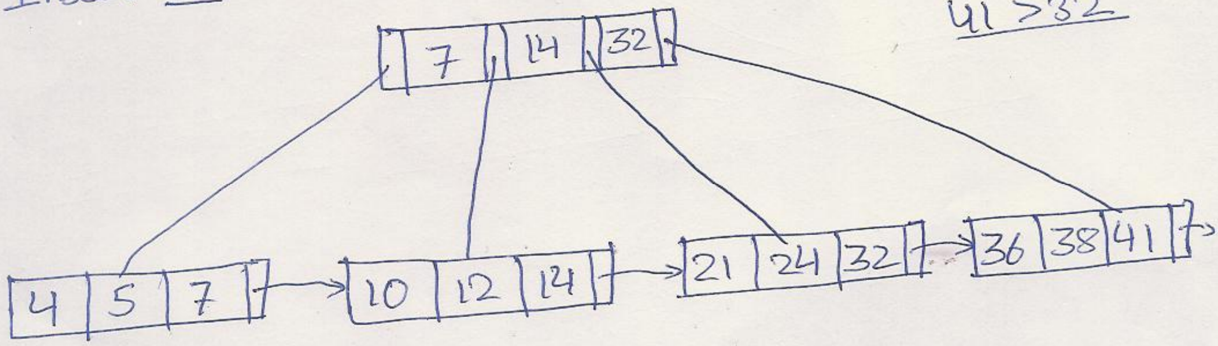


Insert 21



Insert 41

41 > 32



Insert 16

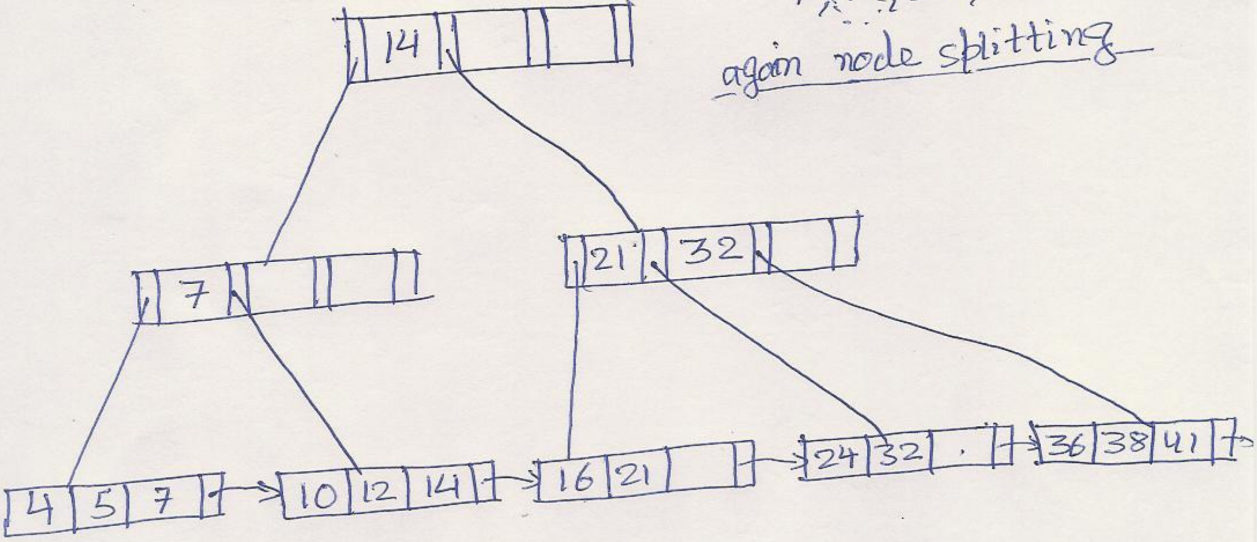
14<16<32

16, 21, 24, 32

at internal node

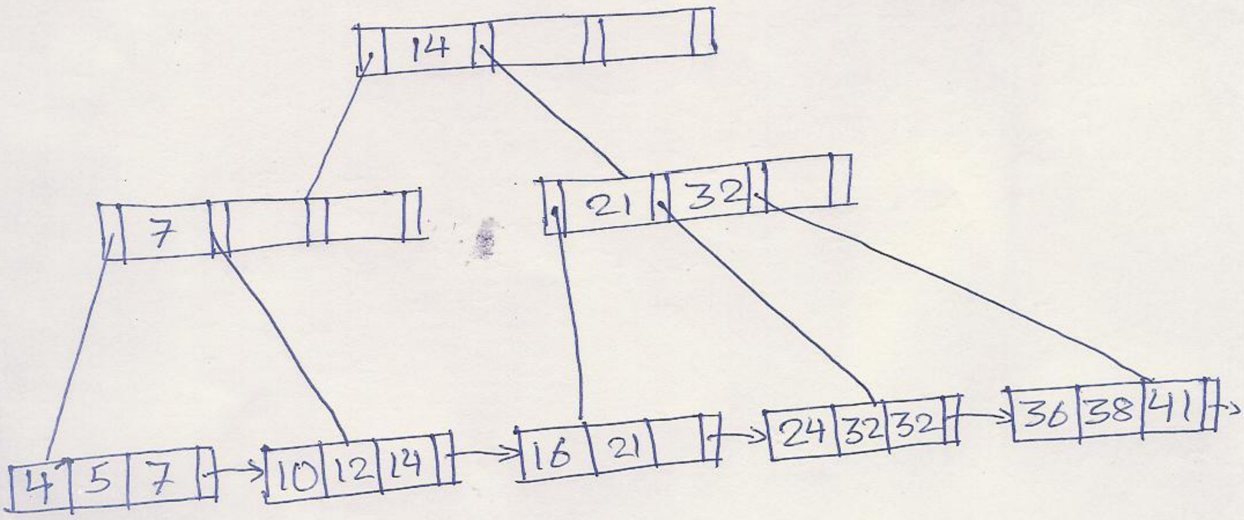
7, 14, 21, 32

again node splitting



Insert 32

32≤32 ✓



Using any method, if this is created, it will go to three levels. Above is the final tree created.

Ans-no-7- i> $\pi_{sid, sname} (\sigma_{rating < 4} (suppliers))$

ii> $\pi_{sid} (\sigma_{color = yellow'} (Catalog \bowtie Parts))$

iv> $\pi_{sid} (\sigma_{\substack{color = 'red' \text{ OR} \\ color = 'yellow'}} (catalog \bowtie Parts))$

v> $\pi_{sid} (\sigma_{color = 'yellow'} (catalog \bowtie Parts))$

$\cap$

$\pi_{sid} (\sigma_{color = 'red'} (catalog \bowtie Parts))$

vi> $\pi_{t1.sid} \left(\sigma_{\substack{t1.sid = t2.sid \\ \text{AND } t1.pid \neq t2.pid}} (\rho(t1, catalog) \times \rho(t2, catalog)) \right)$

Ans-no-8- i> Select sid, sname
from Suppliers
where rating < 4;

ii> Select sid
from catalog natural join parts
where color = 'yellow';

iii) Select sname, color
from Suppliers natural join catalog natural
join Parts
Order by (color);

iv) Select sid
from catalog natural join Parts
where color = 'red'
Intersect
Select sid
from catalog natural join Parts
where color = 'yellow';

v) select C1.sid
from catalog C1, catalog C2
where C1.sid = C2.sid and C1.Pid!=C2.Pid;

Ans-no-9. The given table is unnormalized. The table has following attributes.
SID, SNAME, PNAME, PCOST.
Pname and Pcost are multivalued attributes.
For normalizing the table to 1st NF we will have to decompose the table to two tables with any name say R_1 and R_2.

R_1 (Sid, Sname) R_2 (Sid, Pname, Pcost)

The table R_1 is in 2nd NF but R_2 is not in 2 NF. We further decompose R_2 into R_{21} and R_{22} as

R_{21} (Sid, Pname) , R_{22} (Pname, Pcost)

These tables R_1, R_{21}, R_{22} are in 3NF.

Students should draw tables and show the reasons of not being in perticular NF. They will have to show whether the decomposition is lossless and dependency preserving or not.

10.(a) Following are the various operators in Relational algebra.

- Basic operators
  - Projection (π)
  - Selection (σ)
  - cross-product (X)
  - Union (U)
  - Set-difference (-)
  - Rename
- Derived operators
  - Join
  - Intersection (∩)
  - Division (÷)

Students should have explained any four of these operators with a simple example which would make clear the use of particular operator.

(b) Two relations $R(A_1, A_2, A_3, A_4, \ldots, A_n)$ and $S(B_1, B_2, B_3, \ldots, B_n)$ are said to be union compatible if they have same degree(number of attributes in a relation) and $Dom(A_i) = Dom(B_i)$ for $1 \le i \le n$. Dom represents domain of a particular attribute.

Significance: Here we have to explain why union compatibility is important for performing any set operation. Student should write the view as if we are trying to take the union of records present in two relations then we cannot imagine the resultant relation if the two relations are not having same number of attributes. Because in the resultant relation the record will be of type of either of the relation and if the relations are not agreeing on the above specified two conditions then the resultant relation will have records of two different types which contradict a fact that a relation will have records of similar type.