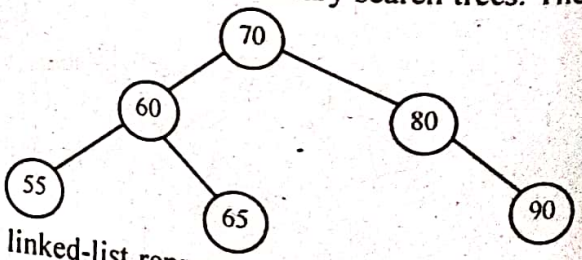
TREES

☞ Chapter at a Glance

- **Tree:** A tree is a finite set of one or more nodes connected by edges such that
  i) There is a specially designated node called the root
  ii) The remaining nodes are partitioned into n (>=0) mutually exclusive disjoint sets
- **Basic Terminology:**
  i) Node: Each data item in a tree is called a node. It specifies the data information & links (branches) to other data items. In the example of Fig (1) the tree has 13 nodes.
  ii) Root: It is the parent of all other nodes of a tree. A root node does not have any parent
  iii) Degree of a node: The number of sub trees of a node is called its degree. In our example the root node has three sub trees (B, C, D). Hence, the degree of the root node A is 3. Degree of B is 2. Degree of C is 1. And so on.
  iv) Degree of a tree: It is the maximum degree of nodes in a given tree. In our example the root node A has the highest degree. Hence, the degree of the tree is 3.
  v) Terminal node(s): A node with degree zero is called a terminal node(s) or leaf node(s) or external node(s). In our example E, F, G, I are leaf nodes.
  vi) Non Terminal node(s): Any node whose degree is not zero is called non-terminal nodes or internal nodes or interior node(s). In our example A, B, C, D & H are the non-terminal node(s).
  vii) Path: It is a sequence of consecutive edges from the source node to the destination node. In our example the path between A & I is given by the edges (A, D) (D, H) & (H, I).
- **Binary tree:** A binary tree is either empty or consists of one single vertex called the root together with two disjoint binary trees called left sub-tree & right sub-tree. Number of branches of any node is either zero or one or at most two.
- **Binary search tree:** A binary search tree (BST) is a binary tree that is either empty or every data node that forms the tree satisfies the following conditions.
  i) The data in the left child of a node is less than the data in its parent node.
  ii) The data in the right child of a node is greater than the data in its parent node.
  The left and right sub trees of the root are also binary search trees. The fig. shown below is Binary search tree.



- **Threaded Binary tree:** In linked-list representation of binary trees, additional storage space is required to store the two links of each node. The null pointers just results in wasting memory. To overcome this drawback, the null pointers are replaced by appropriate pointer values called *threads*.

1. If a binary t
replaced by th
a) success
Answer: (a)

2. In a height b
by more than
a) 2
Answer: (c)

3. Maximum po
a) 3
Answer: (a)

4. The in-order
respectively. W
tree?
a) 1
Answer: (d)

5. A binary tree is
a) That is or
b) Such that
c) For which
d) In which n
Answer: (b)

6. A B-tree is
a) Always bala
c) A direct tree
Answer: (d)

7. Which tree stru
memory?
a) AVL Tree
Answer: (b)

8. Which of the follo
notation?
a) A parse tree
c) An operator st
Answer: (c)

Multiple Choice Type Questions

1. If a binary tree is threaded for in-order traversal a right NULL link of any node is replaced by the address of its [WBUT 2007, 2016]
   a) successor b) predecessor c) root d) own
Answer: (a)

2. In a height balanced tree the heights of two sub-trees of every node never differ by more than [WBUT 2007, 2009, 2018]
   a) 2 b) 0 c) 1 d) –1
Answer: (c)

3. Maximum possible height of an AVL Tree with 7 nodes is [WBUT 2008, 2013, 2016, 2018]
   a) 3 b) 4 c) 5 d) 6
Answer: (a)

4. The in-order and post-order traversal of a binary tree are DBEAFC and DEBFCA respectively. What will be the total number of nodes in the left subtree of the given tree? [WBUT 2008]
   a) 1 b) 4 c) 5 d) None of these
Answer: (d)

5. A binary tree is a special type of tree [WBUT 2008]
   a) That is ordered
   b) Such that no node has degree more than 2
   c) For which both (a) and (b) above are correct
   d) In which non-leaf nodes will have degree 2
Answer: (b)

6. A B-tree is [WBUT 2008, 2018]
   a) Always balanced b) An ordered tree
   c) A direct tree d) All of these
Answer: (d)

7. Which tree structure is used for efficient access of records residing in disc memory? [WBUT 2009]
   a) AVL Tree b) B Tree c) 2-3 Tree d) Binary Tree
Answer: (b)

8. Which of the following is essential for converting an infix expression to postfix notation? [WBUT 2009]
   a) A parse tree b) An operand stack
   c) An operator stack d) None of these
Answer: (c)

9. The values in a BST can be sorted in ascending order by using which of the following traversals? [WBUT 2009]
a) Pre – order b) In - order c) Post – order d) Level - order
Answer: (b)

10. The prefix expression for the infix expression a. (b+c) / e – f is [WBUT 2009]
a) / * a + bc – ef b) - / * + abcef
c) - / * a + bcef d) None of these
Answer: (c)

11. In array representation of Binary tree, if the index number of a child node is then the index number of its present node is [WBUT 2010, 2014]
a) 2 b) 3 c) 4 d) 5
Answer: (d)

12. The depth of a complete binary tree with n nodes [WBUT 2012]
a) log(n+1)–1 b) log (n) c) log(n–1)+1 d) log(n)+1
Answer: (a)

13. In a binary search tree , if the number of nodes of a tree is 9, then the minimum height of the tree is [WBUT 2012]
a) 9 b) 5 c) 4 d) None of these
Answer: (d)

14. Which method of traversal does not stack to hold nodes that are waiting to be processed: [WBUT 2013]
a) Breadth - first b) Depth – first
c) D- search d) None of these
Answer: (a)

15. Which of the following is essential for converting an infix expression to a postfix expression efficiently? [WBUT 2015]
a) An operator stack b) An operand stack
c) An operand stack and operator stack d) A parse tree
Answer: (a)

16. Number of all possible binary trees with 4 nodes is – [WBUT 2014]
a) 13 b) 12 c) 14 d) 15
Answer: (c)

17. If the inorder and preorder traversal of a binary tree are D, B, F, E, G, H, A and A, B, D, E, F, G, H, C respectively then, the postorder traversal of that tree is [WBUT 2014]
a) D, F, G, A, B, C, H, E b) F, H, D, G, E, B, C, A
c) C, G, H, F, E, D, B, A d) D, F, H, G, E, B, C, A
Answer: (d)

18. The number of p
a) 4082
Answer: (b)

19. A binary tree has
a) $\log n$
Answer: (d)

20. The minimum hei
a) n
Answer: (d)

21. The number of ed
a) $2^{h+1}-1$
Answer: (c)

22. Minimum numbe
h is
a) 2^h-1
Answer: (b)

23. Which one is requ
a) Only inorder se
b) Both preorder a
c) Both inorder an
d) Only postorder
Answer: (c)

24. Number of nodes i
a) 2^k
Answer: (c)

1. Prove that for any n
be the number of node
Answer:
Let n and B denote the to
branches in T.
Let $n0$, $n1$, $n2$ represen
respectively.
$n = n0+n1+n2$,

18. The number of possible distinct binary trees with 12 nodes is [WBUT 2015]
a) 4082 b) 4084 c) 3082 d) 3084
Answer: (b)

19. A binary tree has n leaf nodes. The number of nodes of degree 2 in this tree is [WBUT 2015]
a) $\log n$ b) $n-1$ c) n d) cannot be said
Answer: (d)

20. The minimum height of a binary tree of n nodes is [WBUT 2016]
a) n b) n/2 c) n/2 – 2 d) $\log_2(n+1)$
Answer: (d)

21. The number of edges in a full binary tree of height h is [WBUT 2017]
a) $2^{h+1}-1$ b) 2^h-1 c) $2^{h+1}-2$ d) 2^h-2
Answer: (c)

22. Minimum number of nodes required to make a complete binary tree of height h is [WBUT 2017]
a) 2^h-1 b) 2^h c) 2^h+1 d) 2^{h-1}
Answer: (b)

23. Which one is required to reconstruct a binary tree? [WBUT 2017]
a) Only inorder sequence
b) Both preorder and postorder sequences
c) Both inorder and postorder sequences
d) Only postorder sequence
Answer: (c)

24. Number of nodes in a complete binary tree of depth k is [WBUT 2018]
a) 2^k b) $2k$ c) 2^k-1 d) None of these
Answer: (c)

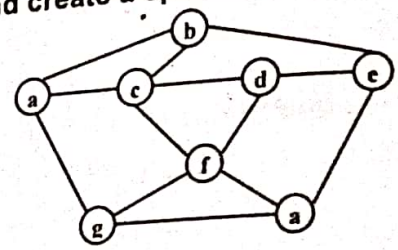
Short Answer Type Questions

1. Prove that for any non-empty binary true T, if n_0 is the number of leaves and n_2 be the number of nodes having degree 2, then $n_0 = n_2 + 1$. [WBUT 2007]

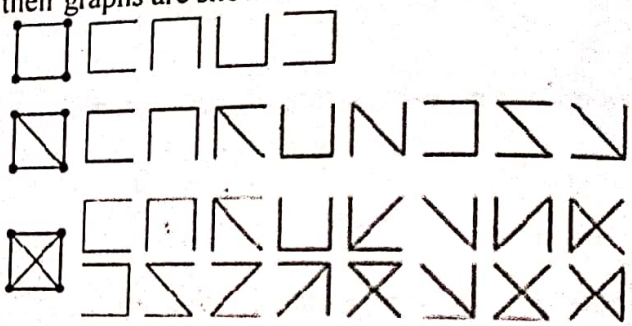
Answer:
Let n and B denote the total number of nodes & branches in T.
Let $n0$, $n1$, $n2$ represent the nodes with no children, single child, and two children respectively.
$n = n0+n1+n2$,

$B+1=n$,
$B=n1+2n2$
$\Longrightarrow n1+2n2+1=n$,
$n1+2n2+1=n0+n1+n2$
$\Longrightarrow n0=n2+1$

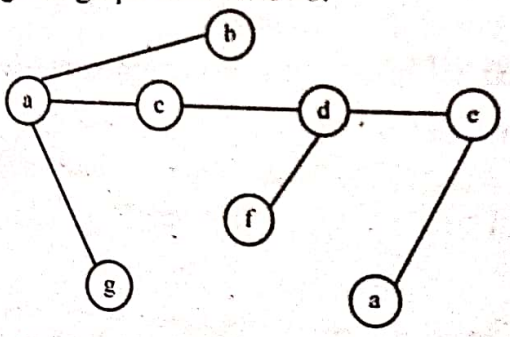
2. Explain spanning tree and create a spanning from the following graph. [WBUT 2007]



Answer:
A spanning tree of a graph is a subset of edges that form a tree. Examples of various spanning trees from their graphs are shown below:



The spanning tree for the given graph is as follows:



3. Do you consider the following data-structure as linear? [WBUT 200

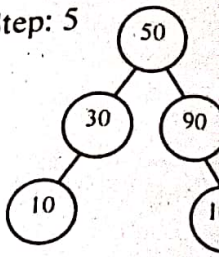
- **Binary tree**

Answer:
Nonlinear, traversing the nodes in a nonlinear pattern, root, then left or right, any one that. So in worst case the traversing will be the height of the tree.

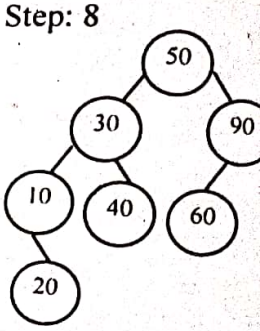
4. Show how the following integers can be inserted in an empty binary search tree in the order they are given:
50, 30, 10, 90, 100, 40, 60, 20, 110, 5.
Draw the tree in each step. [WBUT 20

Answer:
Step 1: 50

Step: 5



Step: 8



5. Construct the ex

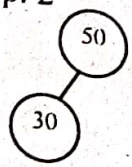
$E = ($

Answer:

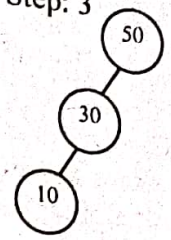
Answer:

Step 1: 50

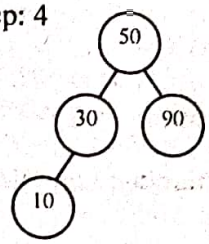
Step: 2



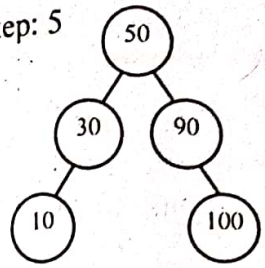
Step: 3



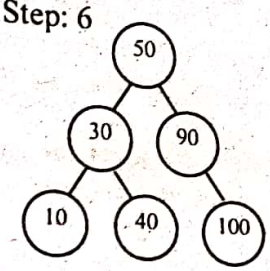
Step: 4



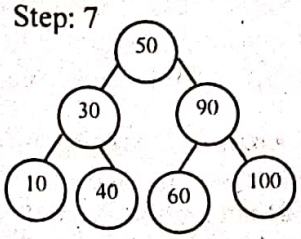
Step: 5



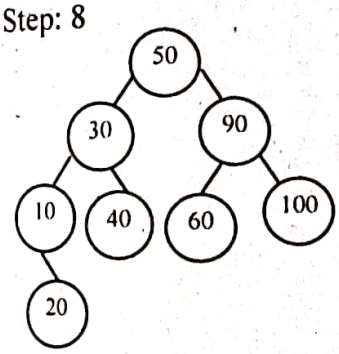
Step: 6



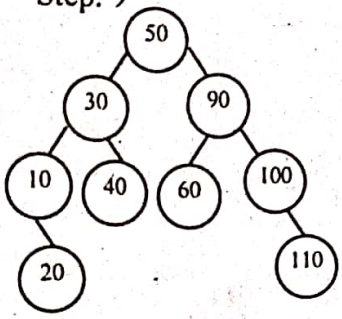
Step: 7



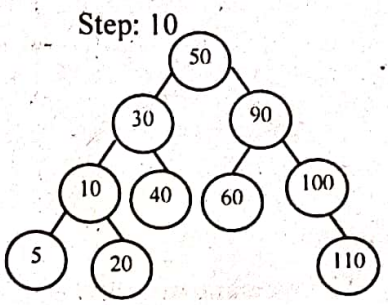
Step: 8



Step: 9



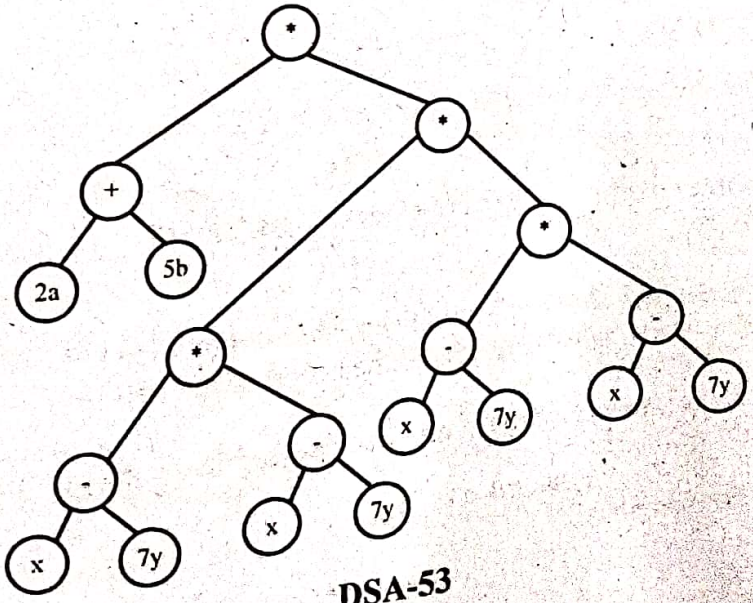
Step: 10



5. Construct the expression tree for the following expression: **[WBUT 2010]**

$$E = (2a + 5b)(x - 7y)^4$$

Answer:



6. Write an algorithm for non-recursive in-order traversal of a threaded binary tree. [WBUT 2011, 2018]

Answer:

Step-1: For the current node check whether it has a left child which is not there in the visited list. If it has then go to step-2 or else step-3.

Step-2: Put that left child in the list of visited nodes and make it your current node in consideration. Go to step-6.

Step-3: For the current node check whether it has a right child. If it has then go to step-4 else go to step-5

Step-4: Make that right child as your current node in consideration. Go to step-6.

Step-5: Check for the threaded node and if it's there make it your current node.

Step-6: Go to step-1 if all the nodes are not over otherwise quit

7. Write a C language function to find the in-order successor of the root of a binary tree. [WBUT 2011]

Answer:

Let the structure of any node be and let d be the root node.

```
struct Node
{
    int data;
    Node* lptr;     // pointer to the left subtree
    Node* rptr;     // pointer to the right subtree
};
Node* successor(Node * d)
{
   if(d == NULL)
       return NULL;

   // If d has a Right Child
   if(d->rptr != NULL)
   {
      // Move to Right Node
      d = d->rptr;

      // Move to the extreme left
      while(d->lptr != NULL)
         d = d->lptr;

      return d;
   }
```

Below are two functions – first one to find out the minimum key in the binary tree and the second one to find the inorder successor of the root which uses first function.

```
Node *treeminimum (Node *root, Node *nil)
{
   /* returns a pointer to the minimum key in a NONEMPTY tree */

   while (root->left != nil)
```



```
            root =
        return(roo
}
Node *treesucc
{
    /* returns
       nil if
       assumes

    Node *tree
    Node *y;
    if (root->
        return
    y = root->
    while (y !
        {
        root =
        y = y-
        }
    return(y);
}
```

8. Write an algorith

Answer:

The solution is to c
nodes in its left sub
in its right sub tree i
The idea is to "bubb
the conditions above
algorithm is design
(linear) time.
Algorithm
For the current nod
Step1: Get the min a
Step2: If the min an
BST"
Step 3: If there is no
Step 4: Get the min a
Step 5: If the min an
BST"
Step 6: If there is no
current node's key va
Step 7: Return the mi

9. Write an algorithm

```
        root = root->left;
    return(root);
}
Node *treesuccessor (Node *root, Node *nil)
{
    /* returns pointer to inorder successor of root, and to
       nil if root is the maximum of the tree. This code
       assumes that root != nil. */

    Node *treeminimum(Node *, Node *);
    Node *y;
    if (root->right != nil)
        return(treeminimum(root->right,nil));
    y = root->parent;
    while (y != nil && root == y->right)
        {
        root = y;
        y = y->parent;
        }
    return(y);
}
```

8. Write an algorithm to test whether a given binary tree is a binary search tree. **[WBUT 2011]**

Answer:

The solution is to check, for every node in the tree, the min and max key values of the nodes in its left sub tree is less than the value of its key and the min and max key values in its right sub tree is greater than the value of its key. This can be achieved by recursion. The idea is to "bubble up" the min and max values of a node's sub tree, after satisfying the conditions above, to its parent node, which will in turn repeat the same process. The algorithm is designed around pre-order tree traversal and solves the problem in O(n) (linear) time.

Algorithm

For the current node:

Step1: Get the min and max values for left sub tree if left child exists

Step2: If the min and max values are greater or equal to current node's key, return "Not BST"

Step 3: If there is no left child, set the min value to current node's key value

Step 4: Get the min and max values for right sub tree if right child exists

Step 5: If the min and max values are lesser or equal to current node's key, return "Not BST"

Step 6: If there is no right sub tree, set the min value from step 1 and the max value to current node's key value

Step 7: Return the min value from step 1 and max value from step 6

9. Write an algorithm to left rotate a binary tree. **[WBUT 2011]**

Answer:

Input: A binary tree *u*. (*u* may be a subtree of a larger tree).

Output: A (new) binary tree obtained from *u* by performing a left rotation about *u*. If *u* is empty or has an empty right subtree, an empty tree is returned.

Algorithm:

```
BinTree rotateLeft( BinTree u)
    if ( u == nil or rightSubtree(u) == nil )
        return nil;
    v = rightSubtree(u);
    return buildTree(v,rightSubtree(v),buildTree( u, leftSubtree(v), leftSubtree(u)));
```

10. What is binary tree? Construct a binary tree using the Inorder and Postorder traversal of the node given below; **[WBUT 2012]**

Inorder:	D	B	F	E	A	G	C	L	J	H	K
Postorder:	D	F	E	B	G	L	J	K	H	C	A

Answer:

1st Part:

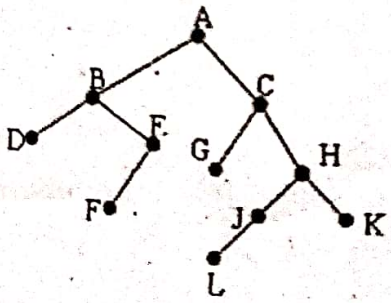
A binary tree is either empty or consists of one single vertex called the root, together with two disjoint binary trees called left subtree & right subtree. Number of branches of any node is either zero or one or at most two.

Binary trees are used to implement binary search trees and binary heaps.

2nd Part:

Based on the postorder traversal, we find A is the root. Now, DBFE is the left child and GCLJHK is the right child of A.

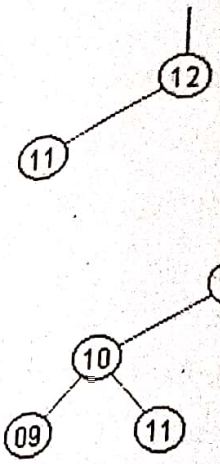
The root of right subtree would be the second last element in preorder i.e., C. We can now further divide the right subtree(with C as root), giving {G} as right subtree and {L J H K} as left. In this way the final trees is given below:

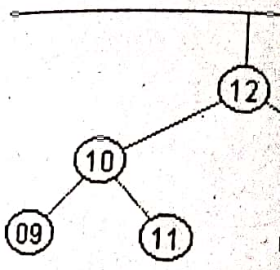


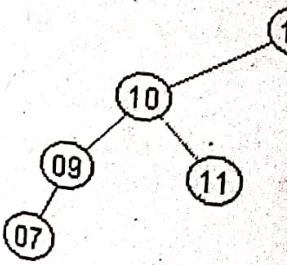
11. Construct an AVL tree using the below list. Show all the steps 12, 11, 13, 10, 9, 15, 14, 18, 7, 6, 5, 4 **[WBUT 2012, 20**

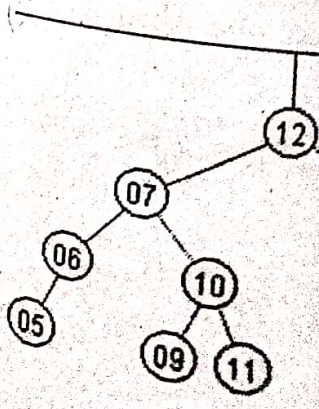
Answer:

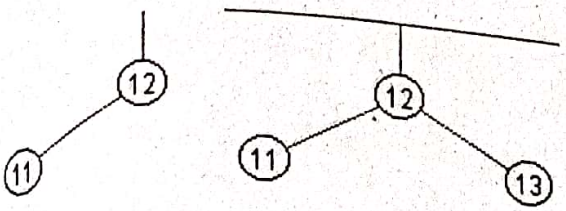
The steps are as shown below:

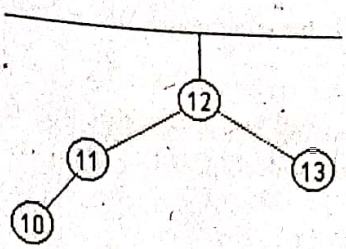


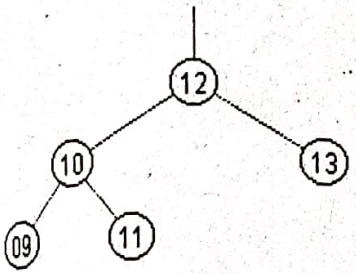


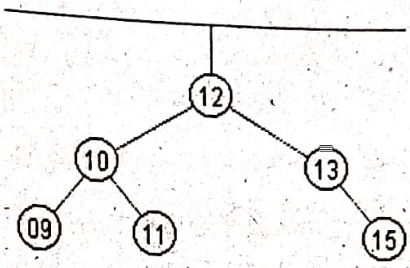


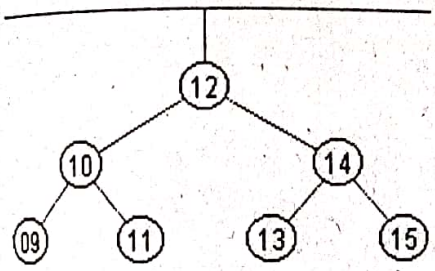


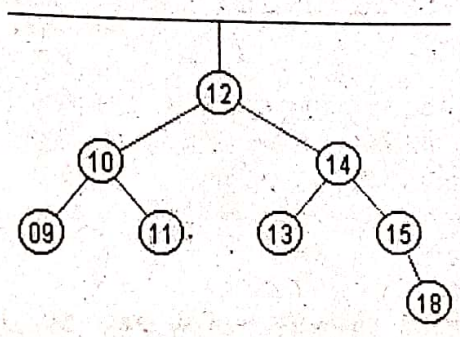


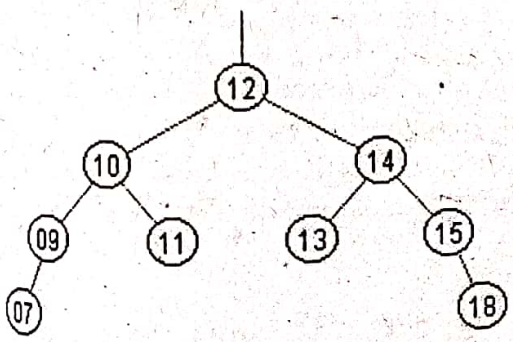


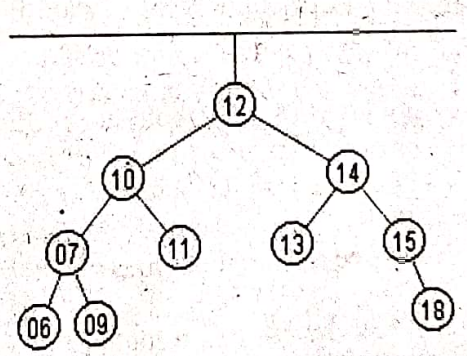


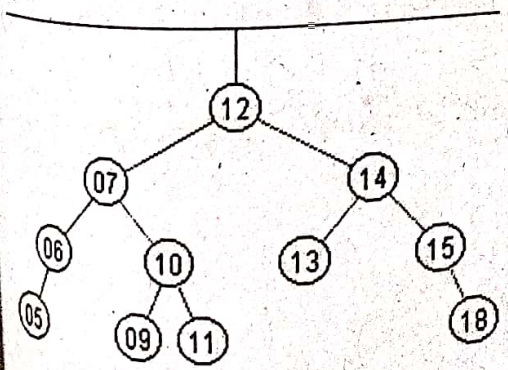


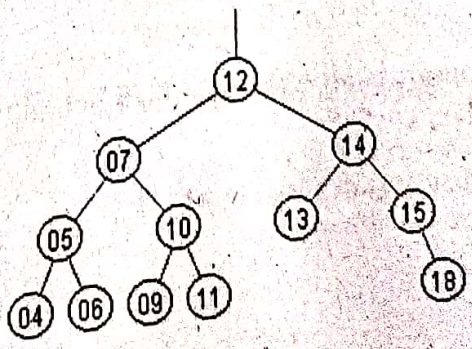












12. Prove that the maximum no. of nodes in a binary tree of depth k is $2^k - 1$. [WBUT 2014]

Answer:

The maximum number of nodes in a binary tree in level i is 2i (according theorem). Hence the maximum number of nodes in a binary tree of depth k is , summation of maximum number of nodes from level 0 upto level k – 1, i.e.,

$$\sum_{i=0}^{k-1} 2^i = 2^0 + 2^1 + 2^2 + \ldots\ldots\ldots + 2^{k-1}$$

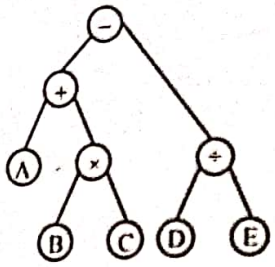
$$= 2^0 * \left(\frac{2^k - 1}{2 - 1}\right) \left[a \cdot \left(\left(r^n - 1\right)/(r - 1)\right)\right]$$

$$= 2^k - 1$$

13. For the following expression draw the corresponding expression tree: [WBUT 2015]

$a + b * c - d / e$

Answer:



14. a) Does a B tree grow at its leave or at its root? Why?
b) In deleting a key from a B tree, when it is necessary to combine nodes? [WBUT 2016]

Answer:

a) B Tree grows at its root (bottom up). B-trees are balanced search trees designed to work well on magnetic disks or other direct-access secondary storage devices. In a B-tree, new nodes are inserted into the leaf level however, if the B-tree must increase in height, it is the root level or top of the tree that changes. This automatically preserves the balance of the tree and no rotation is necessary.

b) If a node has the minimum number of keys, then deleting a key from the node will cause an underflow and it would violate the B Tree property. It needs to be merged with another node to fix the B Tree back.

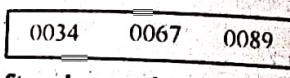
15. The post-order and in-order traversal sequences of codes in a binary tree are given below:
Post-order: D G E B H I F C A
Pre-order: D B G E A C H F I
Construct the binary tree. [WBUT 2016]

Answer:
Assuming in-order i

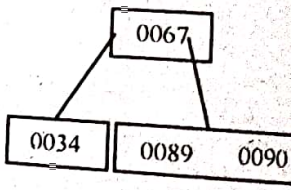
1) We first find
is root as roo
2) We search "
left of "A" in

3) We recur the

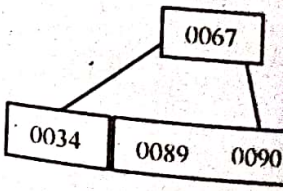
16. Construct one B
36, 76, 53, 51, 12, 10,
Answer:
After inserting 34, 67



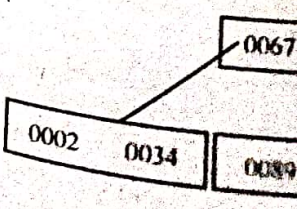
After inserting 90



After inserting 100



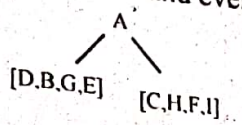
After inserting 2



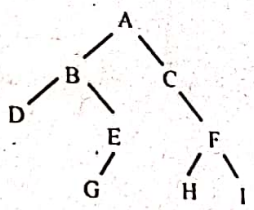
Answer:

Assuming in-order is given instead of pre-order .

1) We first find the last node in post-order. The last node is "A", we know this value is root as root always appear in the end of postorder traversal.
2) We search "A" in in-order to find left and right subtrees of root. Everything on left of "A" in inorder is in left subtree and everything on right is in right subtree.



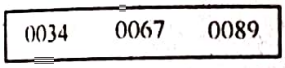
3) We recur the above process for two subtrees.



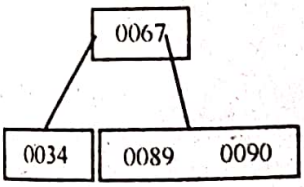
16. Construct one B-Tree of order 4 with the following data. 34, 67, 89, 90, 100, 2, 36, 76, 53, 51, 12, 10, 77, 69. **[WBUT 2016]**

Answer:

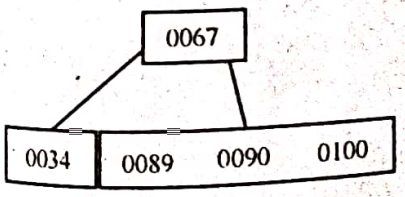
After inserting 34, 67, 89



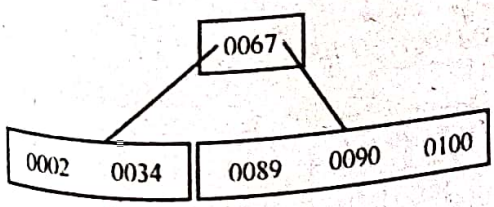
After inserting 90



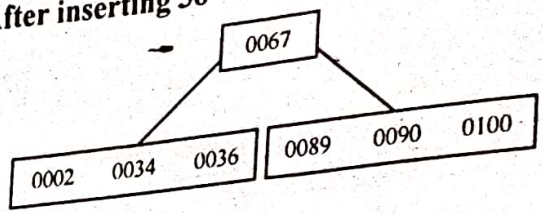
After inserting 100



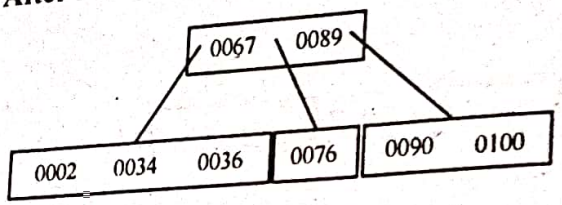
After inserting 2



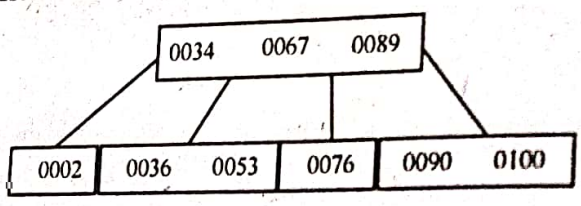
After inserting 36



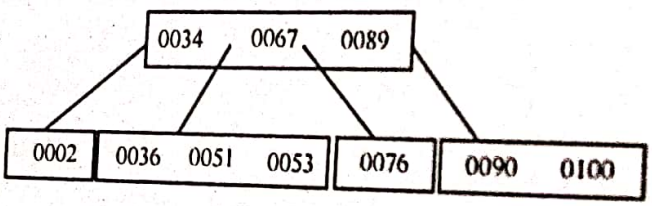
After inserting 76



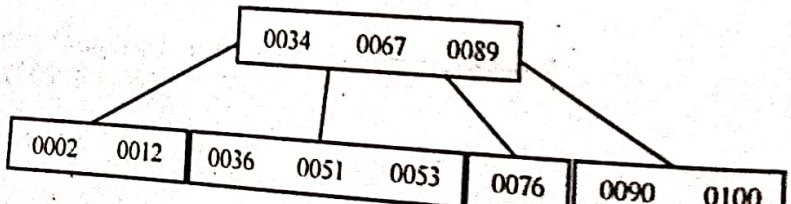
After inserting 53



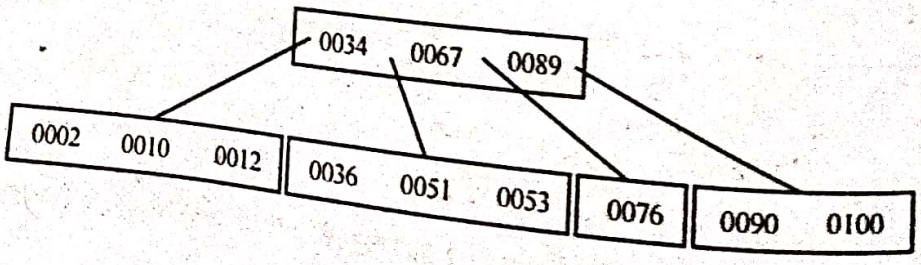
After inserting 51



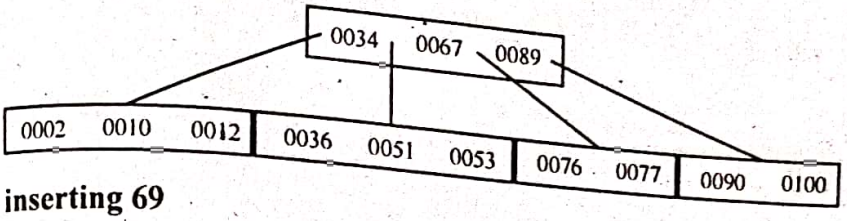
After inserting 12



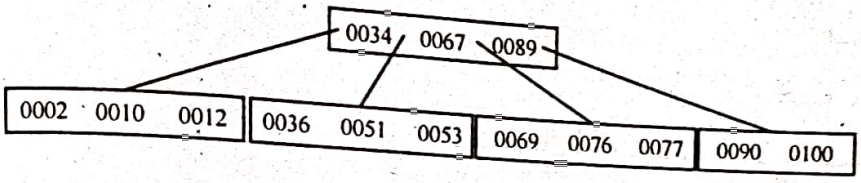
After inserting 10



After inserting 77



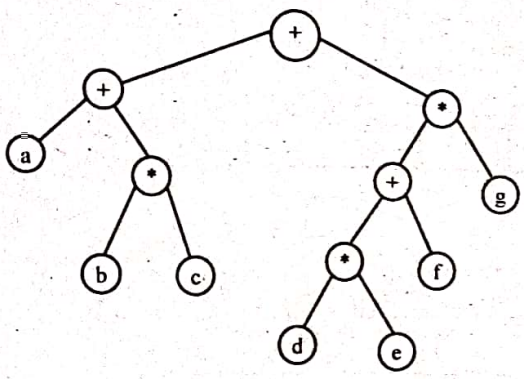
After inserting 69



17. Construct a tree from the given postfix expression $abc*+de*f+g*+$

[WBUT 2016]

Answer:



Postcode traversal: a b c * + d e *f + g * +

18. The inorder and preorder tree traversals are given. Draw the binary tree.

Inorder: ABCDEFGHI

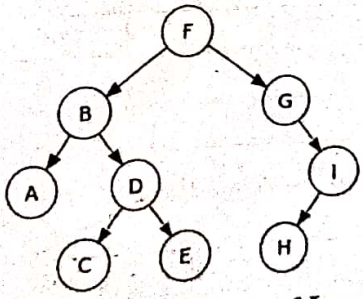
Preorder: FBADCEGIH

Is it possible to build up a unique binary tree when its preorder and postorder traversals are given?

[WBUT 2017]

Answer:

1st part:



2nd part: ***Refer to Question No. 13(b) (1st Part) of Long Answer Type Questions.***

19. Insert the following keys into a B-Tree of given order mentioned below:
a, f, b, k, h, m, e, s, r, c. (Order 3)
a, g, f, b, k, d, h, m, j, e, s, l, r, x, c, l, n, t, u, p. (Order 5) **[WBUT 2018]**

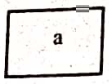
Answer:

1st part:

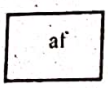
Construction of B-tree: (of order 3)

a, f, b, k, h, m, e, s, r, c

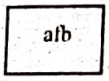
a:



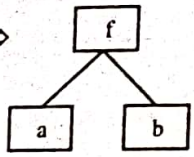
f:



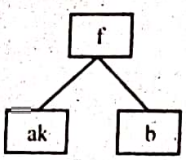
b:



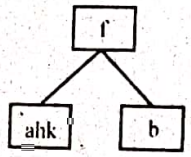
⇒



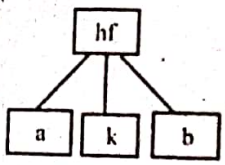
k:



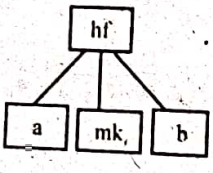
h:



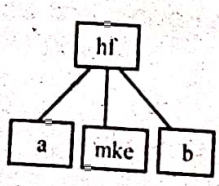
⇒



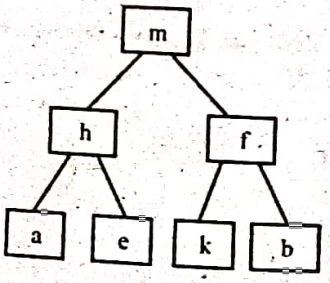
m:



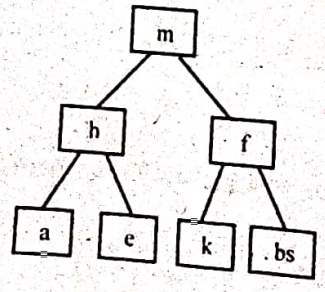
e:



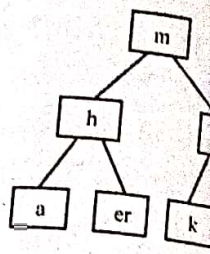
⇒



s:



r:

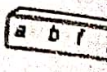


2nd Part:

The insertion steps are sho

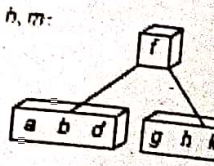
1.

a, g, f, b:



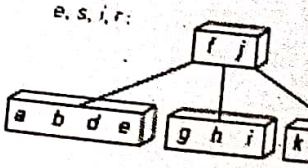
3

d, h, m:



5.

e, s, i, r:



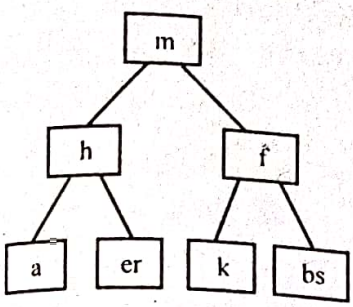
7.

c, l, n, t, u:

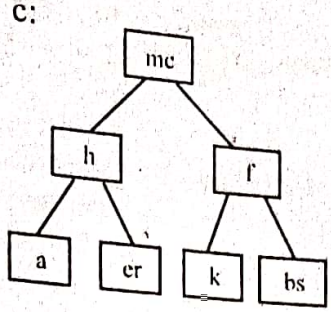
8.

p:

r:



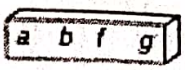
c:



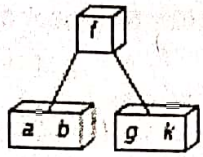
2nd Part:

The insertion steps are shown in the figure below:

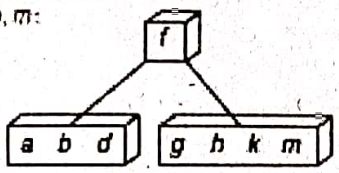
1. a, g, f, b:



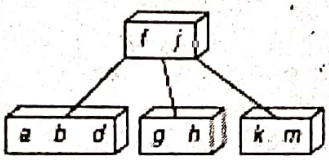
2. k:



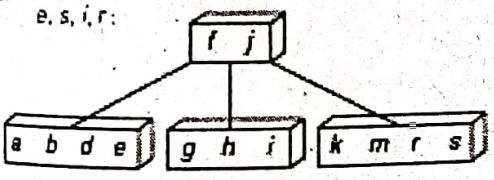
3. d, h, m:



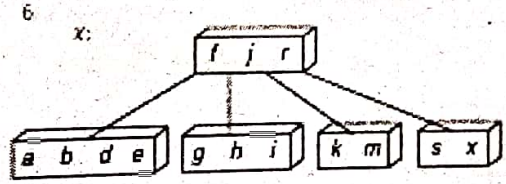
4. j:



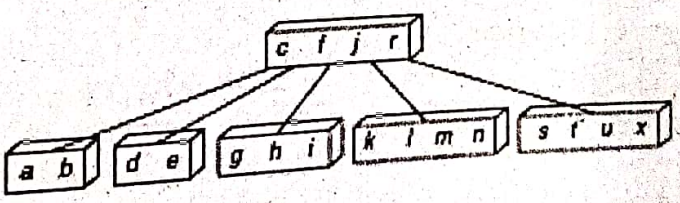
5. e, s, i, r:



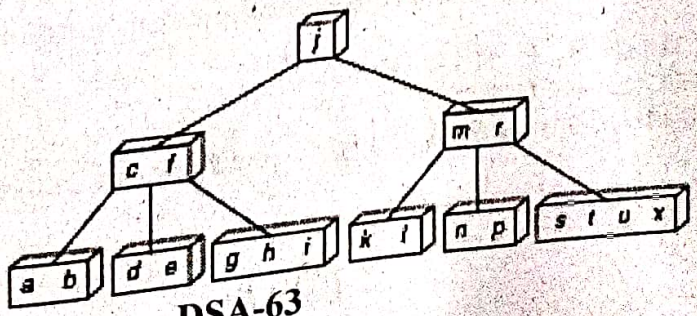
6. x:



7. c, l, n, t, u:



8. p:



20. What is expression tree? Draw the expression tree and write the In, Pre & Post. Order traversals for the given expression tree: E = (2x +y) (5a – b)^3. [WBUT 2018]

Answer:

1st Part:

An expression tree is a representation of expressions arranged in a tree like data structure. It is a tree with leaves as operands of the expression and nodes contains the operators. Data interaction is also possible in an expression tree.

2nd Part: ***Refer to Question No. 2(c) of Long Answer Type Questions.***

Long Answer Type Questions

1. Write an algorithm to insert a node in a binary search tree. [WBUT 2007, 2014]

Answer:

Implementing binary search tree.

We assume that a tree node has left and right pointers and a key element.

```
SearchTree Insert( ElementType X, SearchTree T )
{
     if( T == NULL )
   {
      /* Create and return a one-node tree */
          T = malloc( sizeof( struct TreeNode ) );
           if( T == NULL )
               FatalError( "Out of space!!!" );
      else
       {
            T->Element = X;
          T->Left = T->Right = NULL;
       }
    }
    else
      if( X < T->Element )
          T->Left = Insert( X, T->Left );
       else
     if( X > T->Element )
          T->Right = Insert( X, T->Right );
    /* Else X is in the tree already; we'll do nothing */
      return T;  /* Do not forget this line!! */
}
```

2. a) The inorder and preorder traversal sequence of nodes in a binary tree are given below. [WBUT 2007]

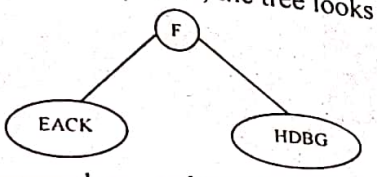
Inorder: E A C K F H D B G

Preorder: F A E K C D H G B

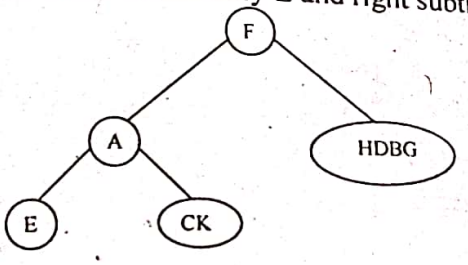
Draw the binary tree. State briefly the logic used to construct the tree.

Answer:

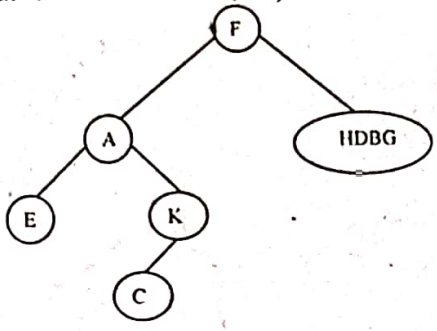
From the preorder traversal, we know that the first node is always the root node of the binary tree. So, F is the root node of the binary tree. Now, from the inorder traversal, we have to find the position of F. All the nodes of F are in the left subtree and all the nodes in right will constitute the right subtree. So, the tree looks like:



Again from the preorder traversal, next element is A. So, A is the root of the left subtree. Again if we find the position of A in the inorder traversal, left element is E & CK are right elements. So, the left subtree consists only E and right subtree consists CK.

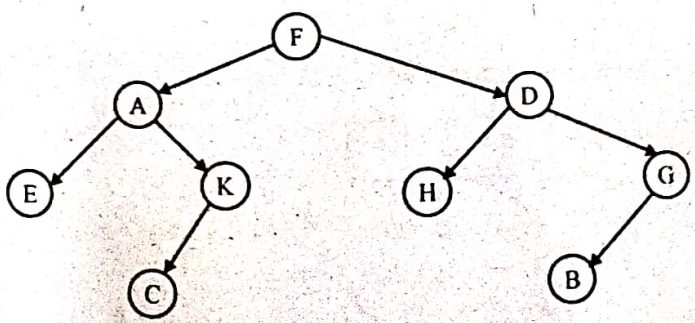


Again for CK, K occurs before C in preorder traversal, so K is the root of right subtree and from the inorder traversal C is at left of K, so, C is the left child of K.



Now, for the right subtree, HDBG of F, the first element in the preorder traversal is D and from the inorder traversal, we say that H is the left child and BG is the right subtree. Again among BG, G is the first element, so G is the root and from the inorder traversal B is the left child.

So, the final tree is



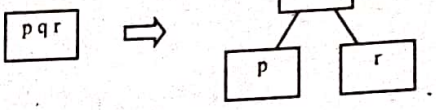
b) Insert the following keys into a B-Tree of order 3: **[WBUT 2007]**
p, q, r, d, h, m, l, s, k, n.

Answer:

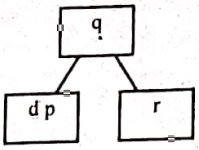
p:

q:

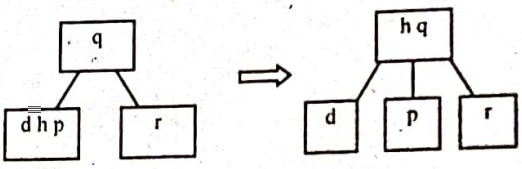
r:



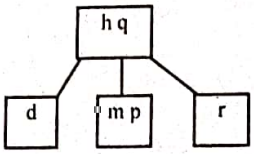
d:



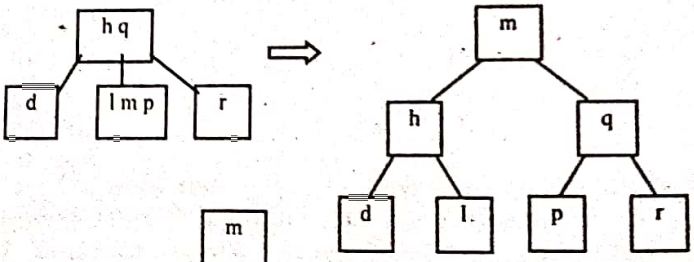
h:



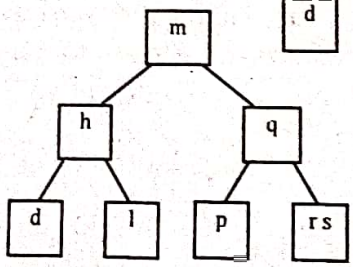
m:



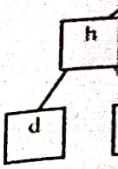
l:



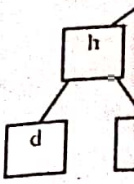
s:



k:

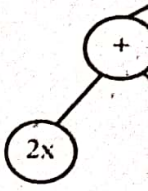


n:



c) Construct an expres

Answer:



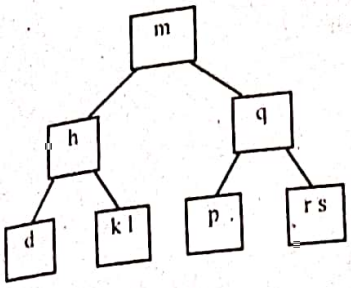
d) Write a non-recursive

Answer:

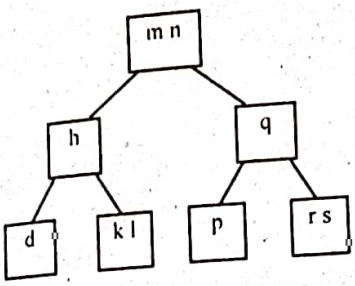
Assume that a stack ADT
input is the root pointer of
the elements.

```
print_inorder (Node
{
  Node temp;
while(left(root) !=nu
  push(stack,left(ro
```

k

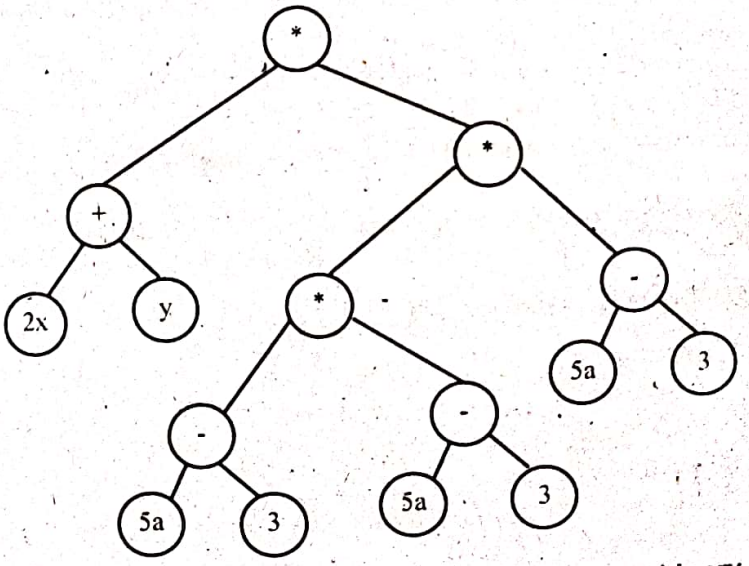


m



c) Construct an expression tree for the expression $E = (2x + y) * (5a - b)^3$. **[WBUT 2007]**

Answer:



d) Write a non-recursive algorithm for in-order traversal of a binary tree. **[WBUT 2007, 2010, 2016]**

Answer:

Assume that a stack ADT exists. The following is the non recursive C function whose input is the root pointer of the binary tree stored as a linked list with left, right and key as the elements.

```
print_inorder(Node *root)
{
 Node temp;
while(left(root)!=null)
  push(stack,left(root));
```

```
while(isEmpty(stack))
 {
  temp = pop(stack);
   print temp->key;
   push(stack,right(temp));
}
```

3. a) In a 2-tree, if E be the external path length, P be the internal path length and Q be the number of vertices that are not leaves, then prove that $E = P + 2Q$ [WBUT 2008]

Answer:

If a tree contains only its root and no other vertices, then E = P = Q = 0, and the first case of the above proof is trivially correct. Now let us consider a larger tree, & let v be some vertex that is not a leaf, but for which both the children of the vertex v are leaves. Let k be the number of branches on the path from the root to v.

Now let us delete the two children of v from the 2-tree. Since b is not a leaf but its children are, the number of non-leaf nodes goes down from Q to Q – 1. The internal path length I is reduced by the distance to v, that is, to P - k. The distance to each child of v is k + 1, so the external path length is reduced from E to E – 2 * (k+1), but v is now a leaf so its distance, k, must be added, given a new external path length of

$$E - 2 * (k+1) + k = E - k - 2.$$

Since the new tree has fewer vertices then the old one; by the induction hypothesis we know that

$$E - k - 2 = (P - k) + 2 * (Q - 1)$$

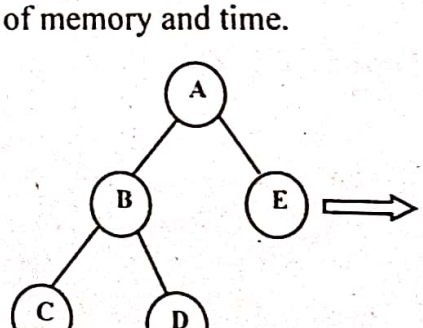
or, $E - k - 2 = P - k + 2q - 2$

or, $E = P + 2Q$. Proved.

b) What is threaded binary tree? [WBUT 2008, 2011]

Answer:

A Threaded Binary Tree is a binary tree in which every node that does not have a right child has a THREAD (in actual sense, a link) to its INORDER successor. By doing this threading we avoid the recursive method of traversing a Tree, which makes use of stacks and consumes a lot of memory and time.



The tree has 6 null links 2 each for the leaf nodes. In-order traversal of the tree is C B D A E.

Let us consider the

- The immediate is a null link, head node. (traversal)
- The right nul point to successor whic order traversal
- The left null li to the imme which is B (as
- The right null order traversal

The above proced binary tree.

The node structure

```
struct NODE
{
 struct NODE
 int node_val
 struct NODE
 struct NODE
}
```

c) Write an algor

Answer:

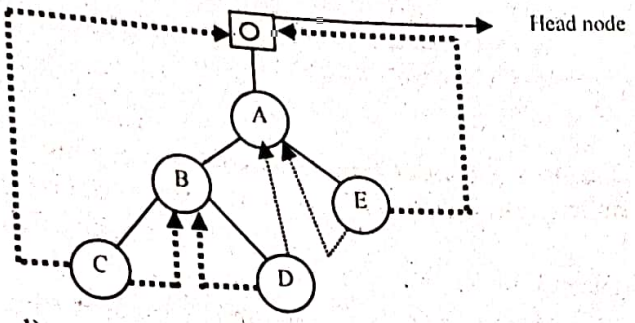
In order to delete

1. Node to be de
2. The node to b
3. The node to b

```
node *delete(
{
  node *q,
  q = NULL;
  root = p;
  while (p!=
  {
    q = p;
    if (x < p
      p = p->
    else
      p = p->
  }
  if (p == NU
  {
```

Let us consider the tree show above.

- The immediate predecessor of C will be the head node. So the left child of C, which is a null link, will point to the head node. (as per in-order traversal)
- The right null link of C will point to the immediate successor which is B (as per in-order traversal).
- The left null link of D will point to the immediate predecessor which is B (as per in-order traversal).
- The right null link of D will point to the immediate successor which is A (as per in-order traversal).



The above procedure is followed for the node E also and we get the resultant threaded binary tree.

The node structure for a threaded binary tree varies a bit and its like this --

```
struct NODE
{
struct NODE *leftchild;
int node_value;
struct NODE *rightchild;
struct NODE *thread;
}
```

c) Write an algorithm to delete a node from a binary search tree. [WBUT 2008]

Answer:

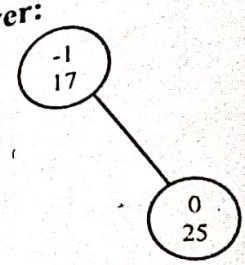
In order to delete a node from a binary search tree there may be three possible scenarios:

1. Node to be deleted has no children
2. The node to be deleted has only one sub tree
3. The node to be deleted has both left & right subtrees

```
node *delete(node *p, int x)
{
 node *q,  *rp,  *a,  *b,  *root;
 q = NULL;
 root = p;
 while (p!= NULL & &x != p->data)
 {
  q = p;
  if (x < p->data)
    p = p->lc;
  else
    p = p->rc;
 }
 if (p == NULL)
 {
```

```
    printf("key not present");
    exit(0);
  }
  if (p->lc == NULL) /* node p has no left child*/
    rp = p->rc;
  else if (p->rc == NULL)      /* p has no right child*/
    rp = p->lc;
  else
  {
    a = p; /* a contains the parent node p */
    rp = p->rc;
    b = rp->lc; /* b is always left child of rp */
    while (b ! = NULL) /* to find the inorder successor with no
left child */
    {
      a = rp;
      rp = b;
      b = rp->lc;
    }
        /* At this point rp is the inorder successor of p */
    if (a ! = p)
    {
      a->lc = rp->rc;
      rp->rc = p->rc;
    }
    if (q == NULL)
      nroot = rp;
    else if (p == q->lc)
    {
      q->lc = rp;
      rp->lc = p->lc;
    }
    else
    {
      q->rc = rp;
      rp->lc = p->lc;
    }
    free(p); // delete the node
    return (root);
  }
```


d) Create a AVL tree
are given: 17 25
Answer:



4. a) Prove that, the h
n and at least [log (
b) The order of node
under:

In order: D B F E
Pre-order: A B D

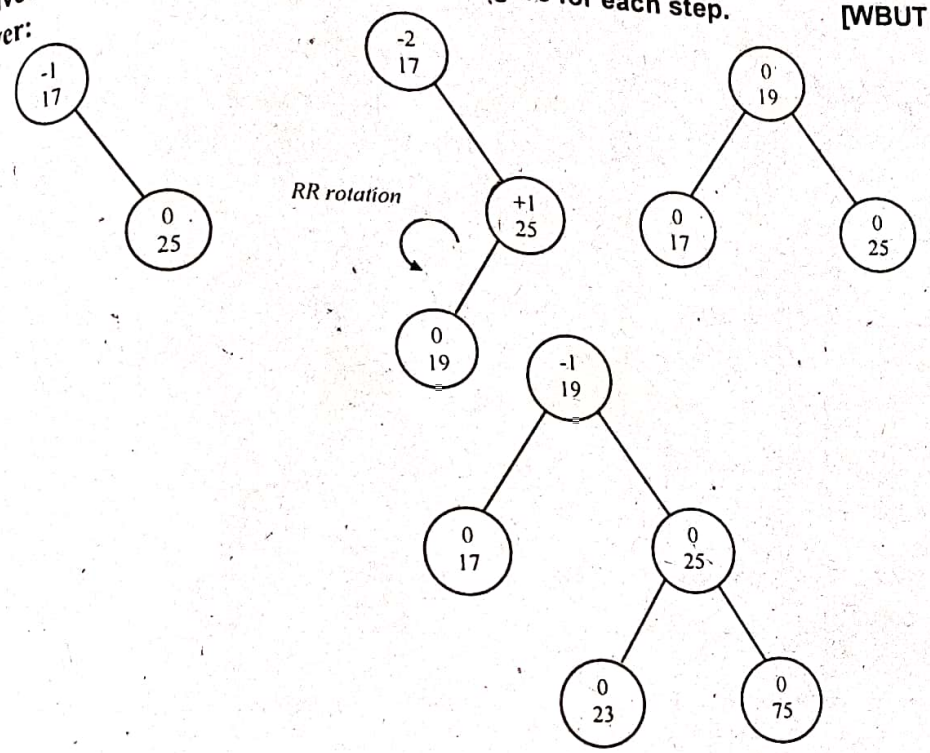
Draw the correspond
c) How does static al

Answer:

a) The worst case is $h \leq O(n+1)$. The best
the bottom level (the
bottom level can effec
the left-most pair of n
height of the tree. Thu
internal nodes: $2^{h-1} < n$

d) Create a AVL tree by inserting the following numbers in the order in which they are given: 17 25 19 23 75. Draw figure for each step. [WBUT 2008]

Answer:



4. a) Prove that, the height of a binary tree that contains n elements, n≥0, is at most n and at least [log (n + 1)].

b) The order of nodes of a binary tree in Preorder and in order traversal are as under:

In order: D B F E G H I A C

Pre-order: A B D E F G H I C

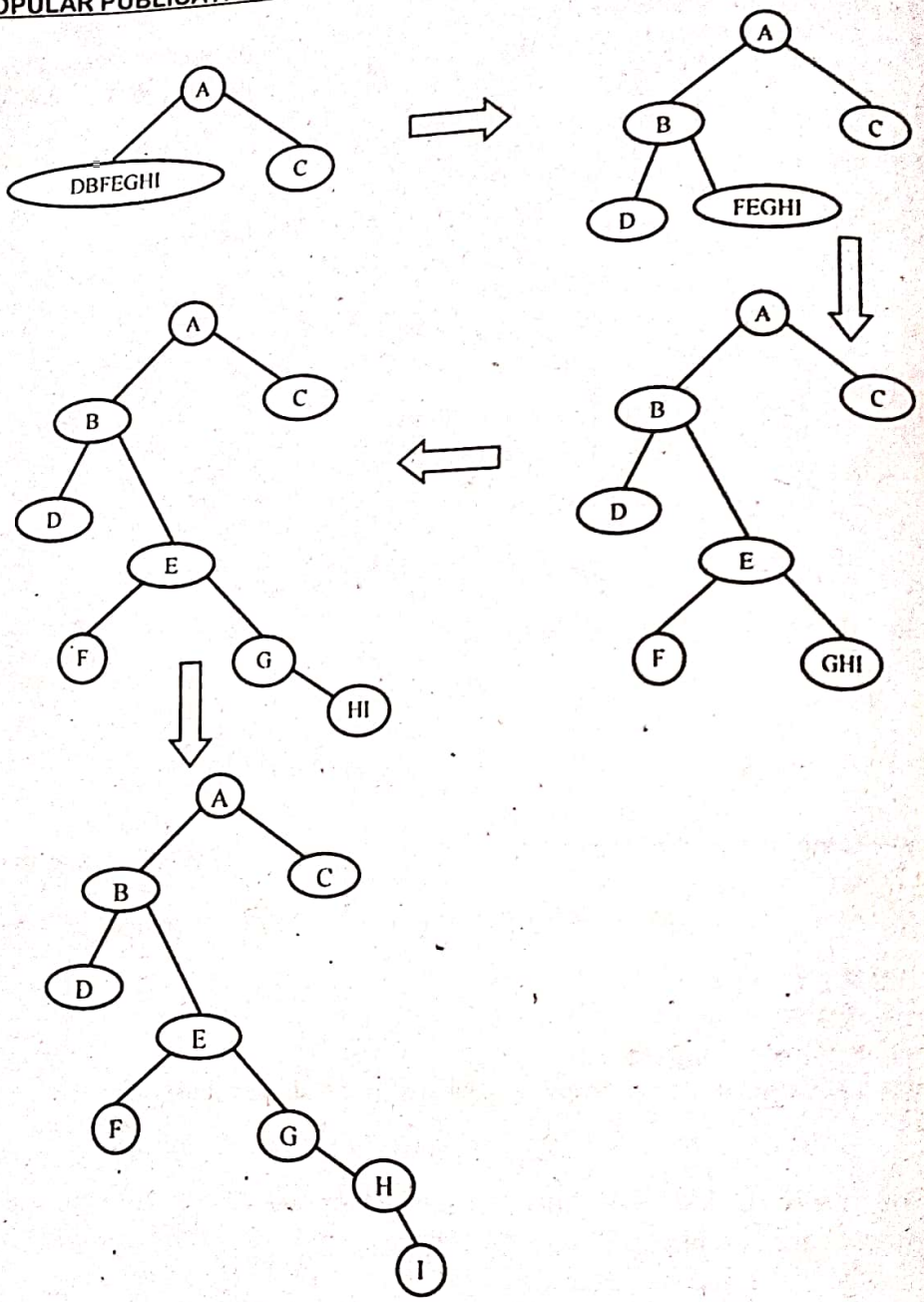
Draw the corresponding binary tree.

c) How does static allocation differ from dynamic allocation of memory? [WBUT 2009]

Answer:

a) The worst case is a linear tree with only one leaf, and thus has n + 1 nodes, that is $h \le O(n+1)$. The best base is a binary tree, with 2^k internal nodes at each level, except for the bottom level (they are all leaf nodes). However, the completion of nodes at the bottom level can effect the number of external nodes from 2^{h-1} (the bottom level has only the left-most pair of nodes) to 2^h (a complete binary tree) external nodes, where h is the height of the tree. Thus, we can have this inequality for external nodes for a true with n internal nodes: $2^{h-1} < n + 1 \le 2^h \Rightarrow h - 1 < \lg (n + 1) \le h$. Thus, $\lg n \le h \le n + 1$.

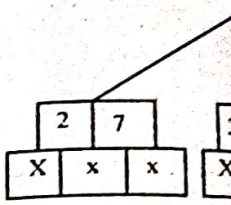
b)



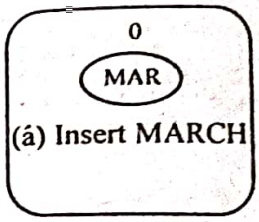
c) Static allocation is done at the time the program is written. The programmer reserves a set amount of memory for each use. That setting will be used each time the program runs. Dynamic allocation is done while the program runs, adjusting the amount of memory for each use according to how much memory required at that time.

5. a) Show the steps in cre
insertion of items in the follo
(March, May, November, Aug
October, September)
b) What do you mean by a
structures?

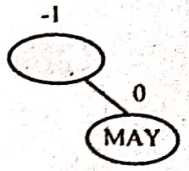
For what purposes are B tree
c) Consider a B-Tree of orde
this order in the B-Tree.



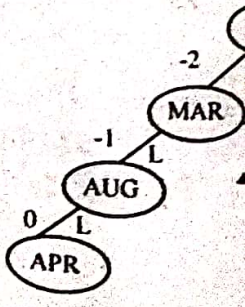
Answer:



(á) Insert MARCH



(b) Insert MAY



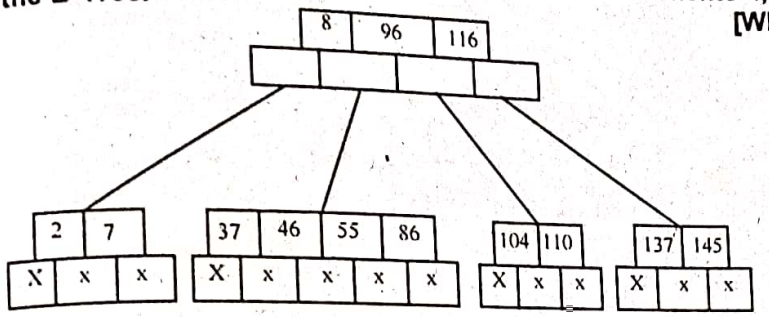
5. a) Show the steps in creation of a height balanced binary AVL TREE using insertion of items in the following order – Show the balancing steps required. (March, May, November, August, April, January, December, July, February, June, October, September) [WBUT 2009]

b) What do you mean by a B-Tree and what are the uses of such a tree in data structures? [WBUT 2009]

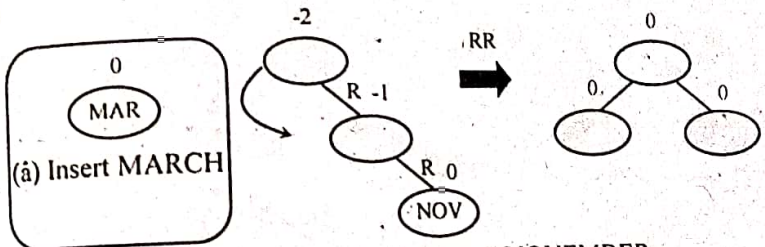
OR,

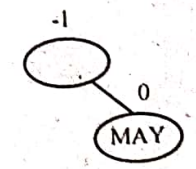
For what purposes are B trees especially appropriate? [WBUT 2016]

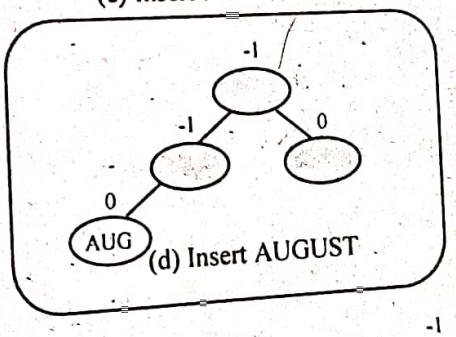
c) Consider a B-Tree of order 5 as shown below – insert the elements 4, 5, 58, 6 in this order in the B-Tree. [WBUT 2009]

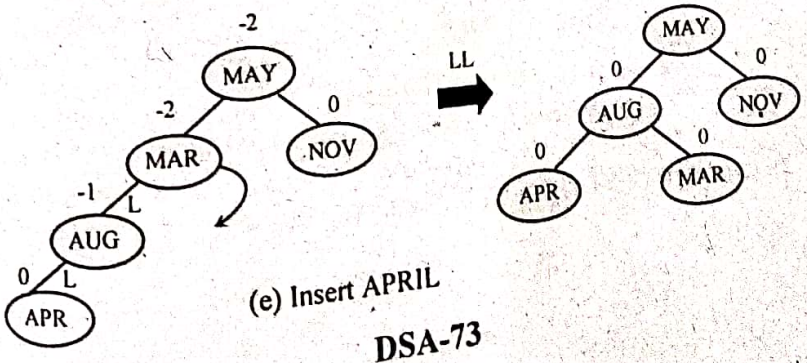


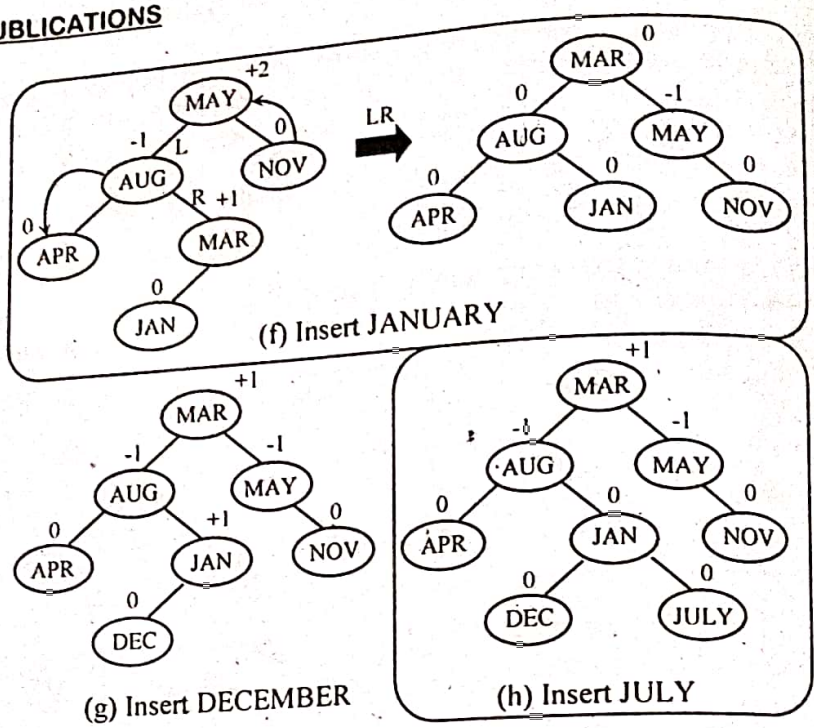
Answer:







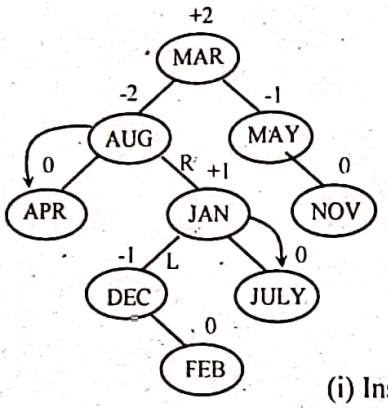


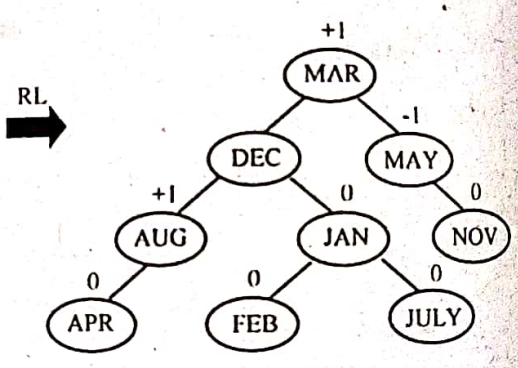


(f) Insert JANUARY

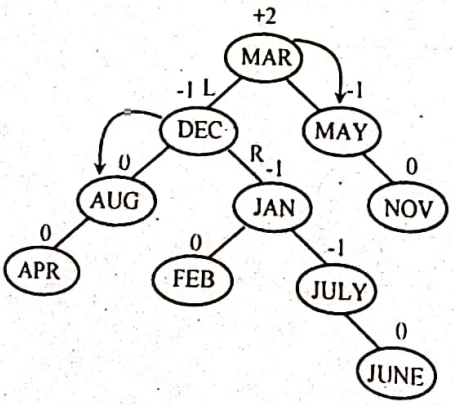
(g) Insert DECEMBER

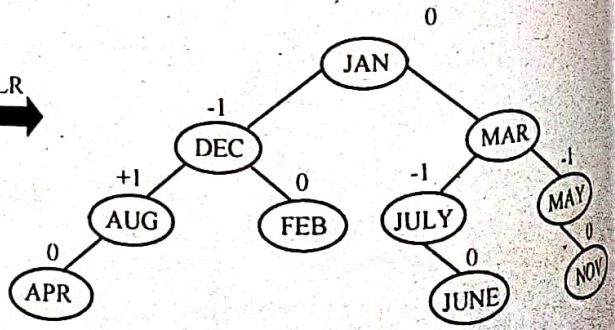
(h) Insert JULY

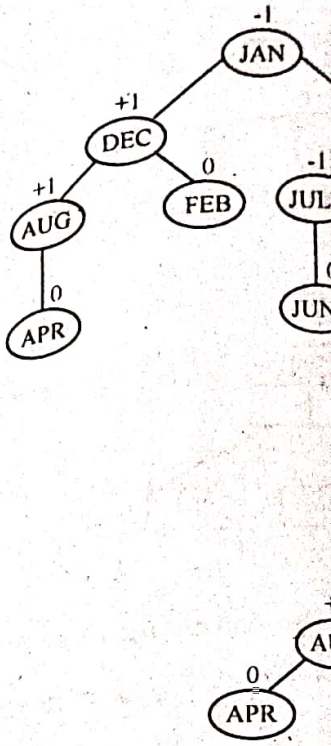




(i) Insert FEBRUARY



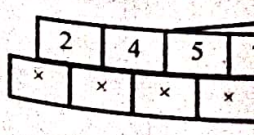


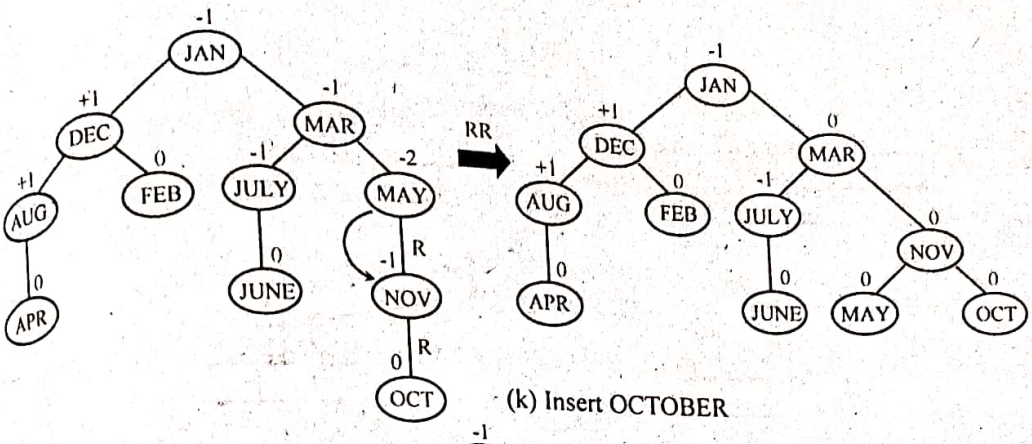


b) B-trees are balanced
must be maintained in s
expensive (time consur
accesses. Each node of
keys are stored in non-d
of a subtree containing
the preceding key. A n
subtree containing all ke
Magnetic disks (second
But they are much slow
information as possible

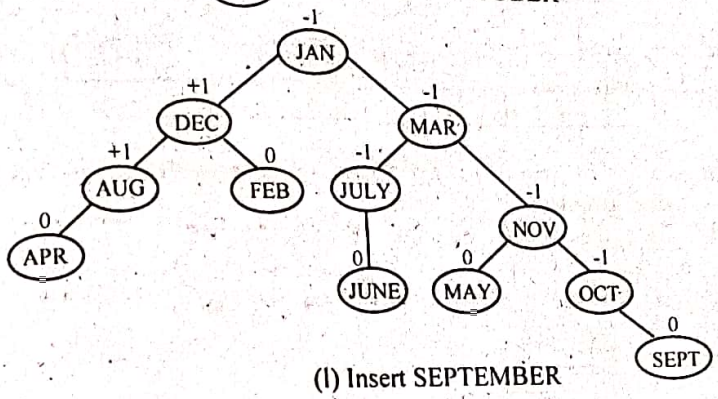
c)

Insert 4, 5





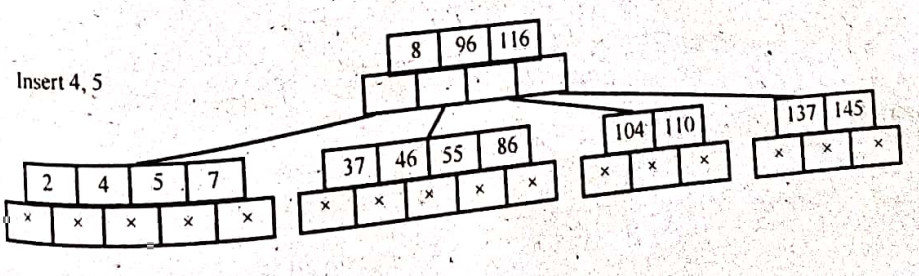
(k) Insert OCTOBER



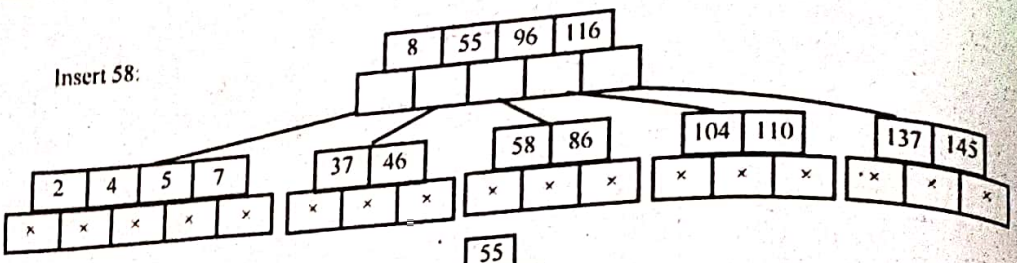
(l) Insert SEPTEMBER

b) B-trees are balanced trees that are optimized for situations when part or the entire tree must be maintained in secondary storage such as a magnetic disk. Since disk accesses are expensive (time consuming) operations, a B-tree tries to minimize the number of disk accesses. Each node of a b-tree may have a variable number of keys and children. The keys are stored in non-decreasing order. Each key has an associated child that is the root of a subtree containing all nodes with keys less than or equal to the key but greater than the preceding key. A node also has an additional rightmost child that is the root for a subtree containing all keys greater than any keys in the node.
Magnetic disks (secondary memory) are cheaper than RAM and have higher capacity. But they are much slower because they have moving parts. B-trees try to read as much information as possible in every disk access operation.

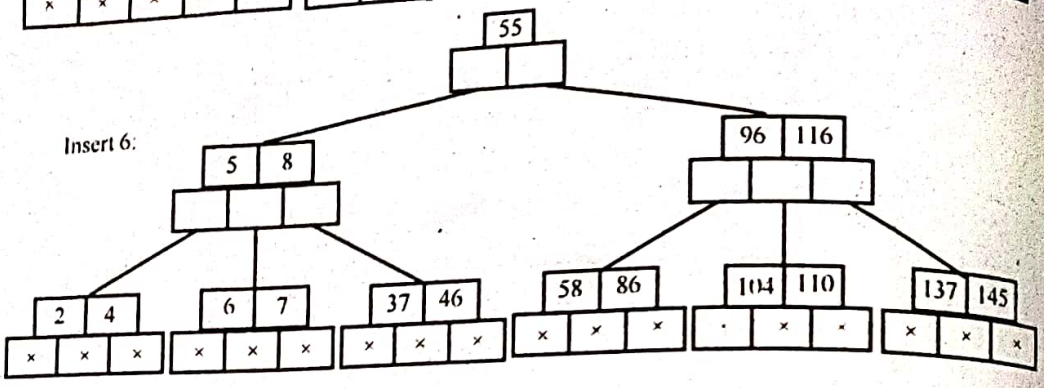
c)



Insert 58:



Insert 6:



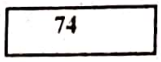
6. a) Show the stages in growth of an order – 4 B-tree when the following keys are inserted in the order given:

74 72 19 87 51 10 35 18 39 60 76 58 19 45 **[WBUT 2010]**

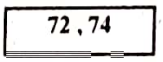
Answer:

74 72 19 87 51 10 35 18 39 60 76 58 19 45

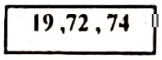
Insert 74:



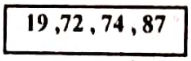
Insert 72:



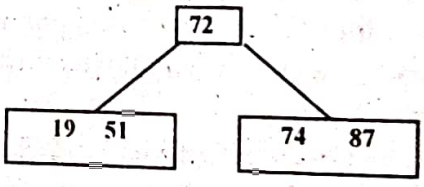
Insert 19:



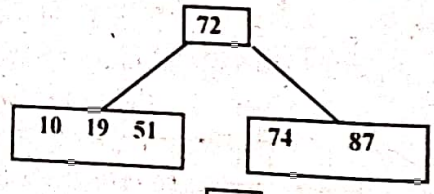
Insert 87:



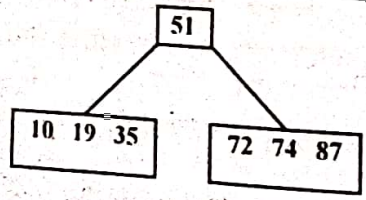
Insert 51:



Insert 10:



Insert 35:



Insert 18:

Insert 39:

Insert 60,76,58,19:

Insert 45:

b) How an AVL tree diff

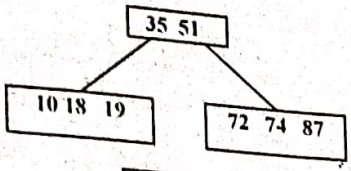
What are the difference

Answer:

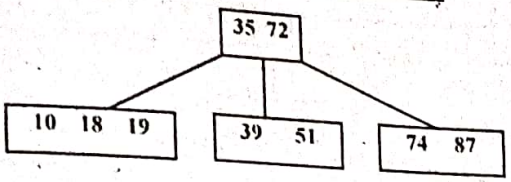
A binary search tree is
the following conditions
i) The left child ha
ii) The right child
Now, an AVL tree is a
as its left and right subtr
i) T_L & T_R are hei
ii) $|h_L - h_R| <= 1$,
Now, an AVL tree is al
tree. In case of binary se
O(n), where n is the nur
be O(n). But since AVL
we can perform searchi
perform some extra oper

c) Insert the following
8 12 9 11 7
Clearly mention differe

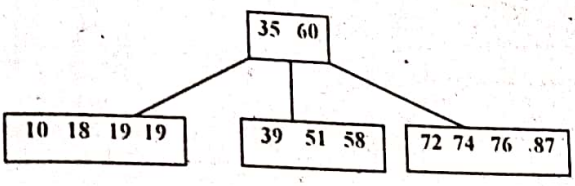
Insert 18:



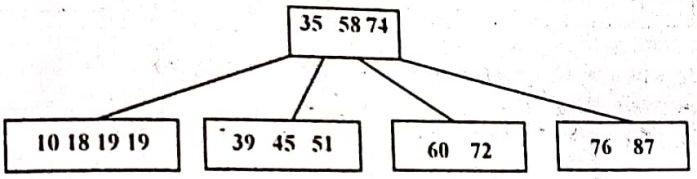
Insert 39:



Insert 60,76,58,19:



Insert 45:



b) How an AVL tree differs from binary search tree? [WBUT 2010]

OR,

What are the differences between AVL Tree & Binary Search Tree? [WBUT 2012]

Answer:

A binary search tree is a binary tree that is either empty or in which each node satisfies the following conditions:

i) The left child has a value smaller than the parent

ii) The right child has a value greater than the parent.

Now, an AVL tree is a height-balanced tree. If T is a non-empty binary tree with T_L & T_R as its left and right subtrees respectively, then T is height balanced if

i) T_L & T_R are height balanced

ii) $|h_L - h_R| <= 1$, where h_L & h_R are the heights of T_L & T_R respectively.

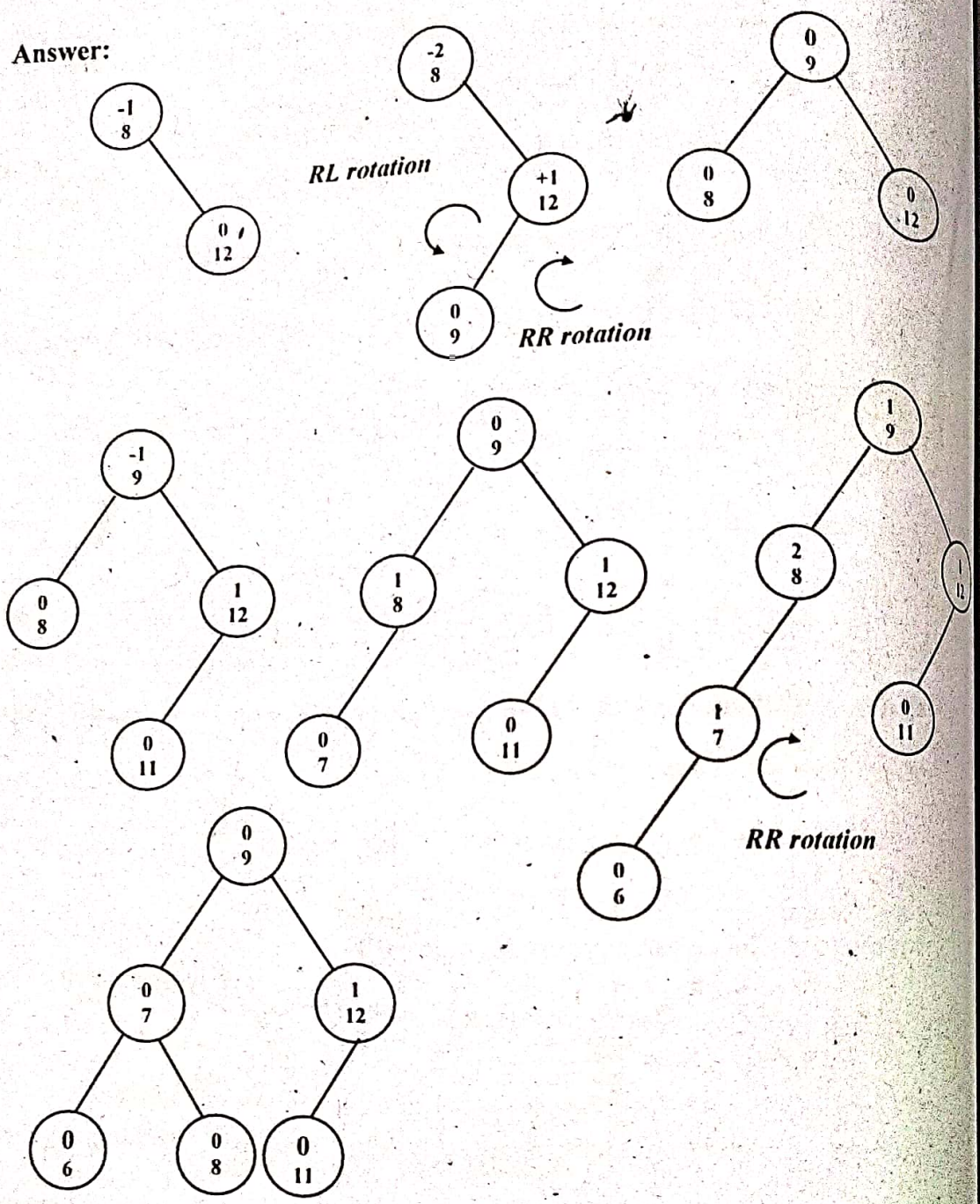
Now, an AVL tree is also a binary search tree, but it is a height balanced binary search tree. In case of binary search tree, if all the data are already sorted, then the height will be O(n), where n is the number of elements in tree. So, then the worst case search time will be O(n). But since AVL tree is height balanced, this type of case will never occur. Here we can perform searching in O (Lgn) time. Now, to construct an AVL tree, we have to perform some extra operations to get the tree height balanced.

c) Insert the following keys in the order given below to build them into an AVL tree.

8 12 9 11 7 6

Clearly mention different rotations used and balance factor of each node. [WBUT 2010]

Answer:



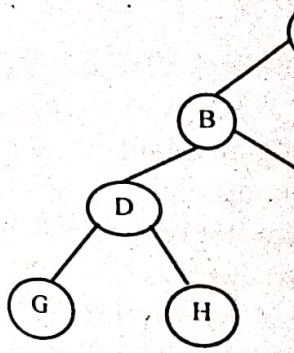
7. a) Given the preorder and inorder sequence and draw the resultant binary tre and write its postorder traversal:

Pre-order: A B D G H E I C F J K
In-order: G D H B E I A C J F K

b) Write an algorithm to search a node in a binary search tree. [WBUT 2016

Answer:

a) In preorder travers
We look for the posi
node is a pivot arou
example all nodes lyi
lying to the right of A
The tree looks like.



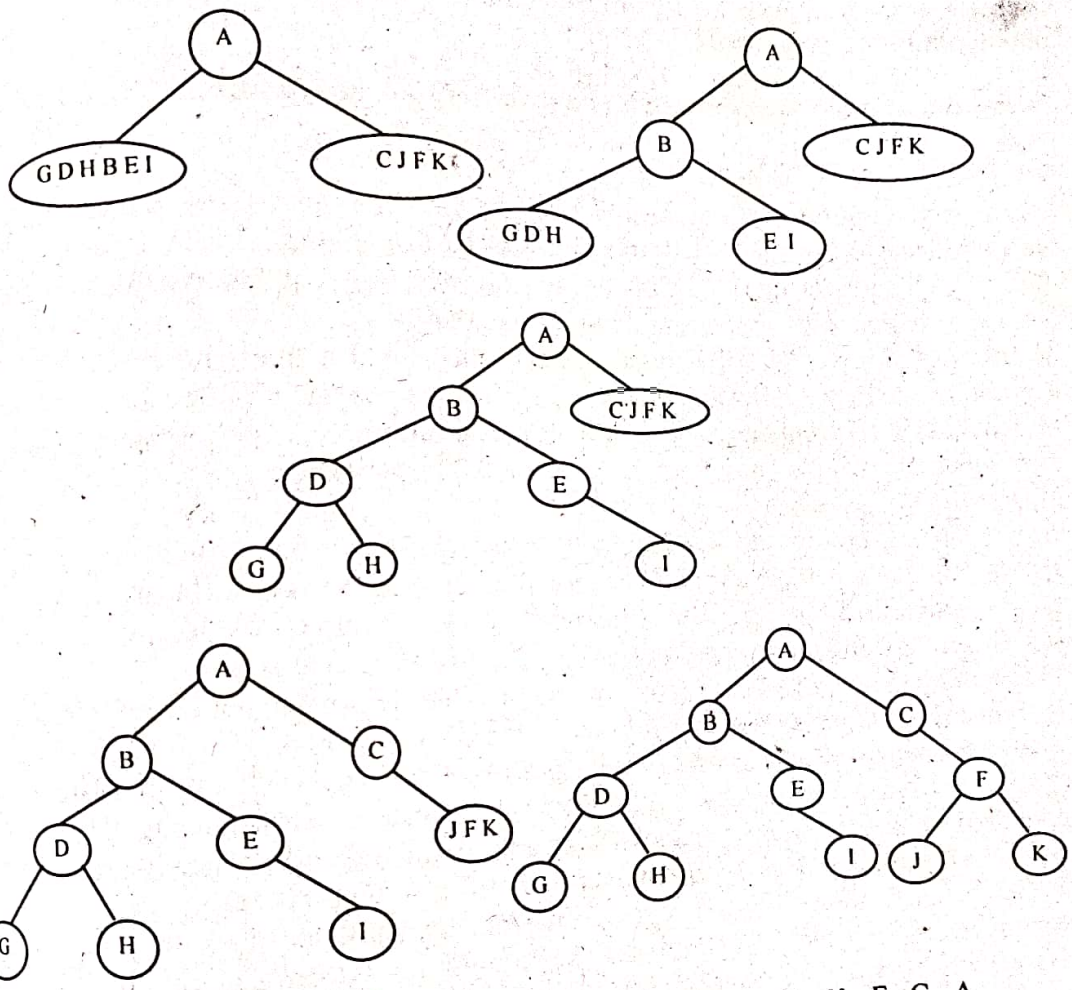
The Post Order trave

b) p is either the roo

```
void search(nod
{
  if (p == NULL
    printf("Err
  else if (digi
    printf("Num
  else if (digi
```

Answer:

a) In preorder traversal, the first node is always the root of the binary tree. Here it is A. We look for the position of A in the corresponding in-order traversal. In order the root node is a pivot around which the left and right subtrees are constructed. So in our example all nodes lying to the left of A will represent the left subtree, & all other nodes lying to the right of A will represent the right sub-tree.
The tree looks like.



The Post Order traversal of the above tree is: G , H , D , I , E , B , J , K , F , C , A

b) p is either the root node or the node from where the search has to begin

```
void search(node *p, int digit)
{
 if (p == NULL)
  printf("Error! Tree structure not found\n");
 else if (digit == p->data)
  printf("Number=%d\n", digit);
 else if (digit < p->data)
```

```
    search(p->lc, digit); //lc = left child
  else
    search(p->rc, digit); // rc =  right child
}
```

8. What are the problems of binary tree? Explain the improvement of performance by the use of height-balanced tree.
Explain how a height-balanced tree can be formed by inserting the following elements in the given order:
1, 2, 3, 4, 5, 6, 8, 9, 10, 7, 11
Show the root element the can be deleted from the above tree. **[WBUT 2011]**

Answer:

1st part:

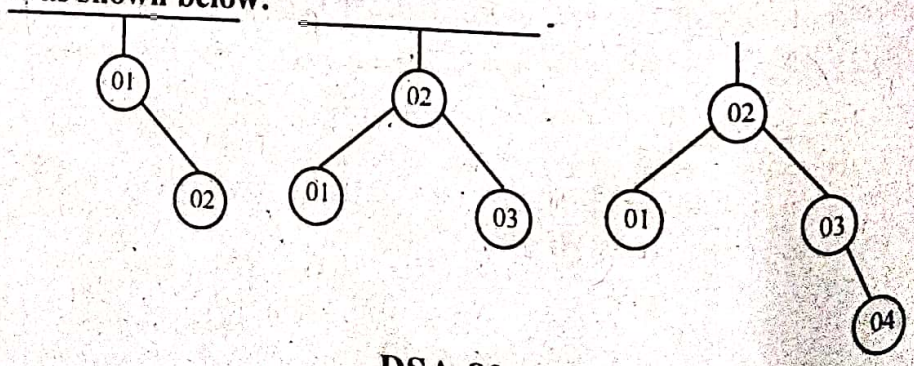
Most operations on a binary search tree (BST) take time directly proportional to the height of the tree, so it is desirable to keep the height small. Since a binary tree with height h contains at most $20+21+\cdots+2h = 2h+1-1$ nodes, it follows that the minimum height of a tree with n nodes is $\log_2(n)$, rounded down; that is, $[\log_2 n]$. However, the simplest algorithms for BST item insertion may yield a tree with height n in rather common situations. For example, when the items are inserted in sorted key order, the tree degenerates into a linked list with n nodes. The difference in performance between the two situations may be enormous: for n = 1,000,000, for example, the minimum height is $[\log_2(n)]-19$.

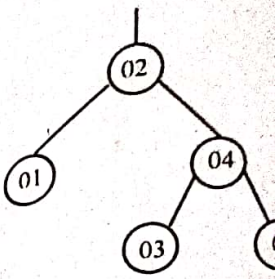
Self-balancing binary trees or height balanced tree solve this problem by performing transformations on the tree (such as tree rotations) at key times, in order to keep the height proportional to $\log_2(n)$. Although a certain overhead is involved, it may be justified in the long run by ensuring fast execution of later operations.

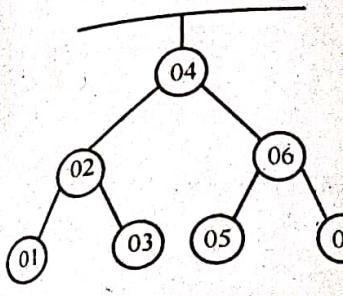
Maintaining the height always at its minimum value $[\log_2(n)]$ is not always viable; it can be proven that any insertion algorithm which did so would have an excessive overhead. Therefore, most self-balanced BST algorithms keep the height within a constant factor of this lower bound. In the asymptotic ("Big-O") sense, a self-balancing BST structure containing n items allows the lookup, insertion, and removal of an item in O(log n) worst-case time, and ordered enumeration of all items in O(n) time. For some implementations these are per-operation time bounds, while for others they are amortized bounds over a sequence of operations. These times are asymptotically optimal among all data structures that manipulate the key only through comparisons.

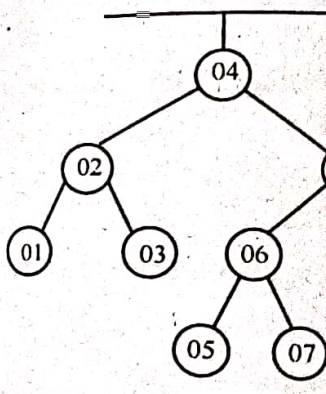
2nd part:

The steps are as shown below:



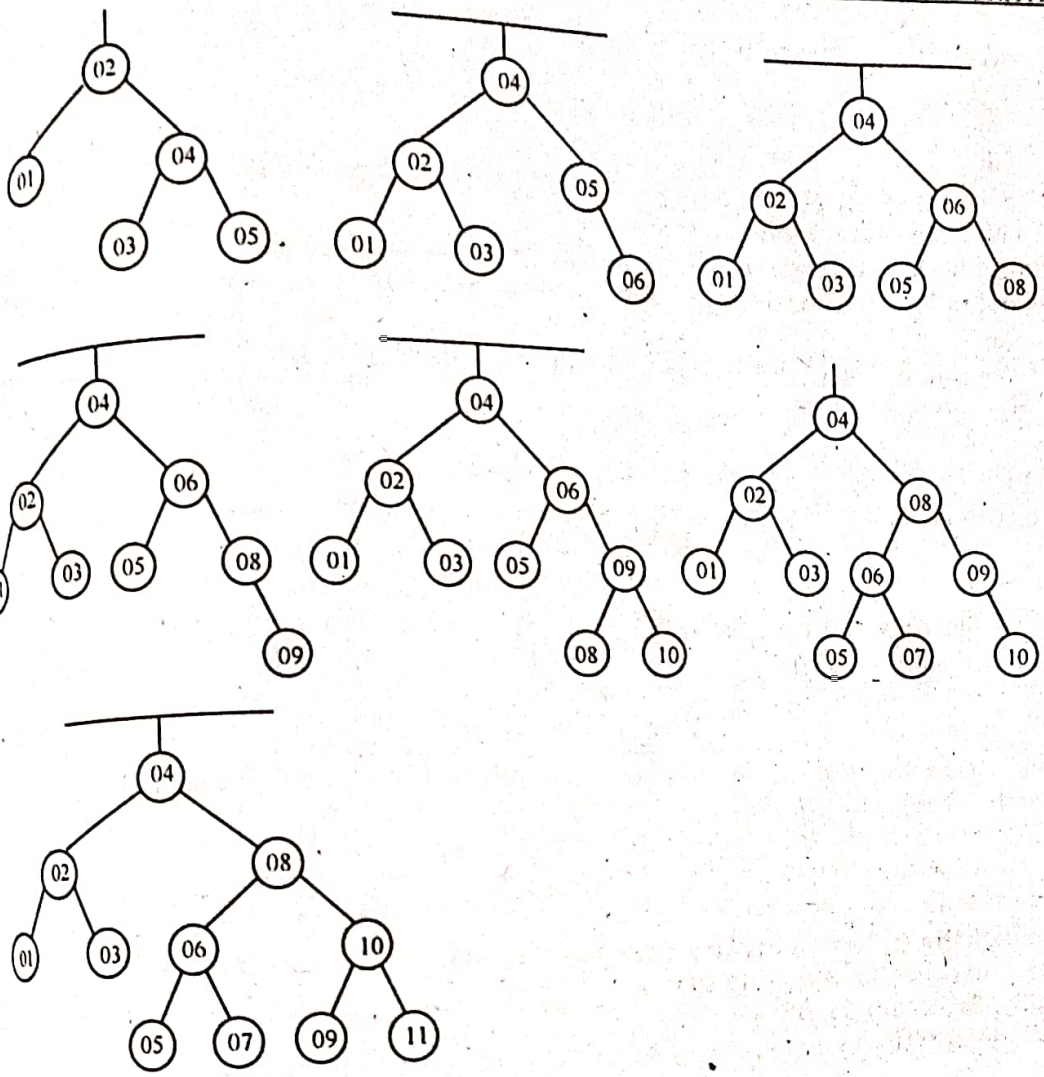




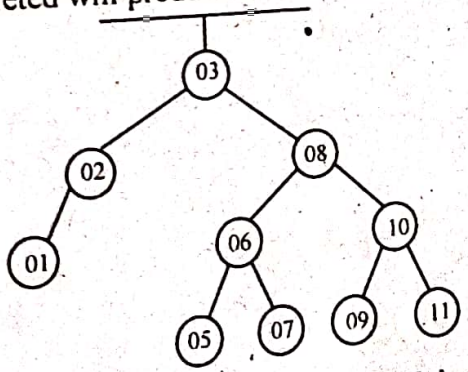


The root element 4 whe

9. What is a B-tree? S
into a B-tree of order



The root element 4 when deleted will produce the following tree:



9. What is a B-tree? Show how the letters A to P of English alphabet can be entered into a B-tree of order 4. **[WBUT 2011]**

Answer:

1st Part:

A B-Tree of order m is an m-way tree in which

i) All leaves are on the same level.

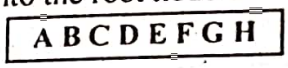
ii) All internal nodes except the root have at most m nonempty children, and at least $\lceil m/2 \rceil$ nonempty children.

iii) The number of keys in each internal node is one less than the number of its nonempty children, and these keys partition the keys in the children in the fashion of a search tree.

iv) The root has at most m children, but may have as few as 2 if it is not a leaf, or none if the tree consists of the root alone.

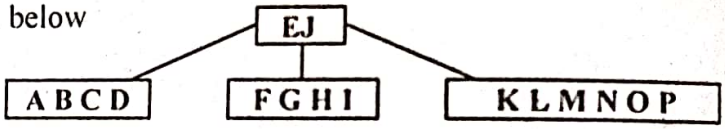
2nd Part:

First A to H would be inserted into the root node one at a time as shown below.



When I has to be inserted, the node will be split at E and then all the alphabets up to M would be inserted as below

When J has to be inserted, the nodes will be split at E and then all the alphabets up to P would be inserted as below



10. Insert the following number into a binary search tree in the order that they are given and draw the resulting tree.

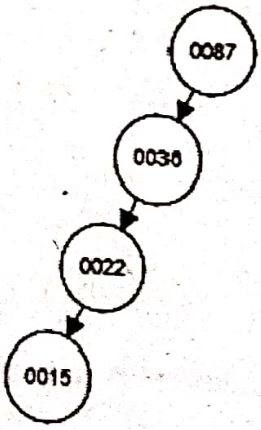
87; 36; 22; 15; 56; 85; 48; 91; 72; 6

Delete 48 and draw the resulting tree. Delete 15 and draw the resulting tree.

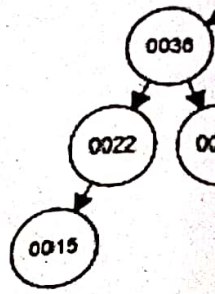
[WBUT 2014]

Answer:

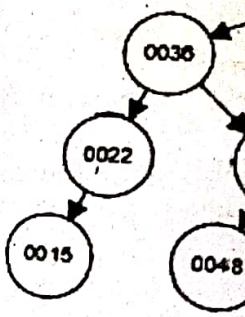
The first 4 nodes are inserted as follows:



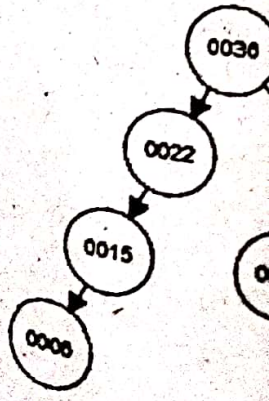
Insert 56, 85



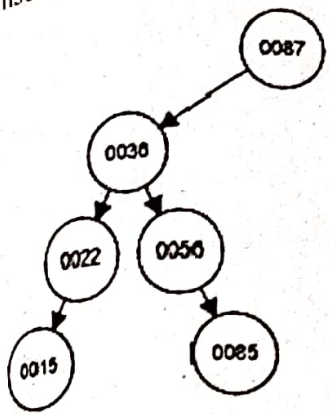
Insert 91



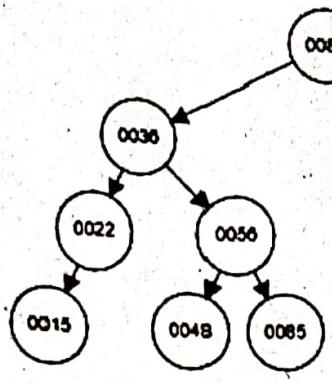
Insert 6



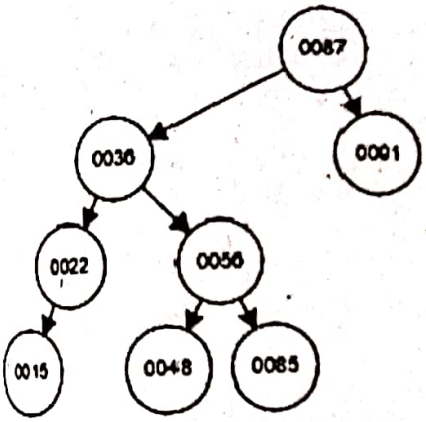
Insert 56, 85



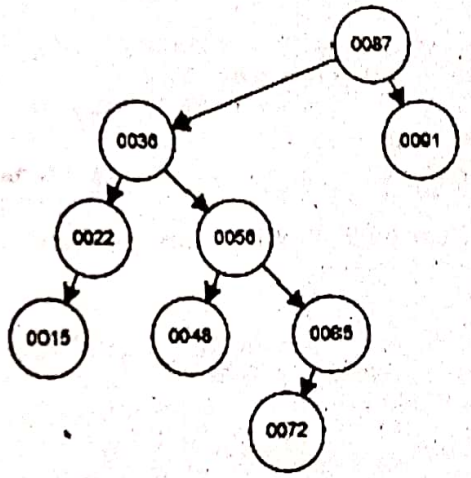
Insert 48



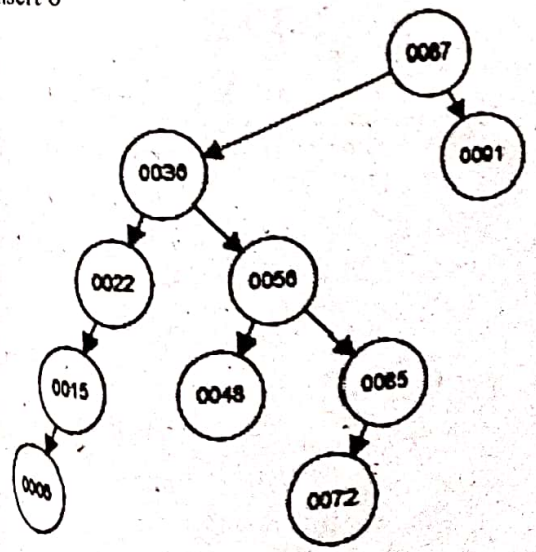
Insert 91



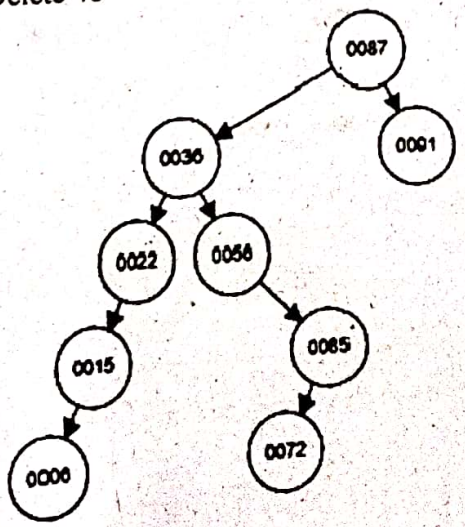
Insert into 72



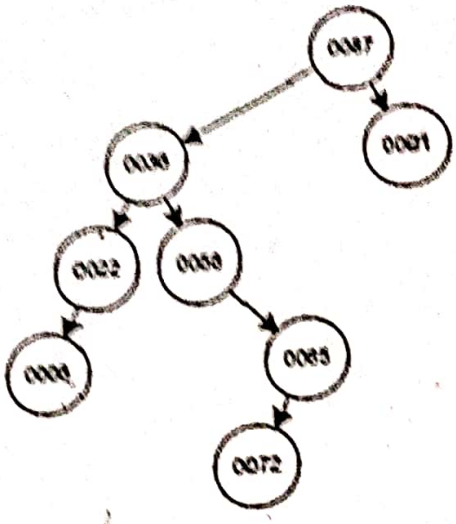
Insert 6



Delete 48



Delete 15



11. a) Show the stages in growth of an order -4B- Tree when the following keys are inserted in the order given:

84 82 29 97 61 10 45 28 49 70 86 68 19 55 22 11 55 77 16

b) How does an AVL tree differ from a binary search tree? Insert the following keys in the order given below to build them into an AVL tree:

8 12 9 11 7 6 66 2 1 44

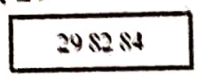
Clearly mention different rotation used and balance factor of each node.

[WBUT 201

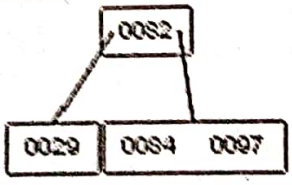
Answer:

a) As order of B-tree is 4, therefore maximum keys in one node is 3 and each node can have maximum of 4 children.

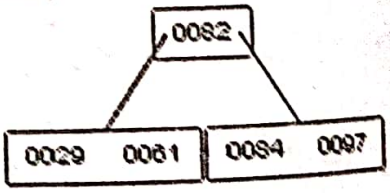
B-tree after inserting 84, 82, 29



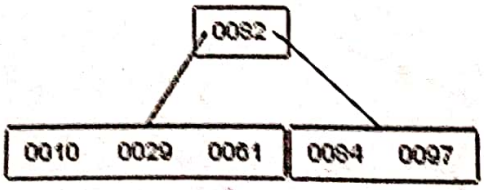
B-tree after inserting 97



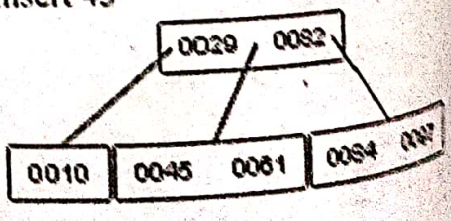
Insert 61



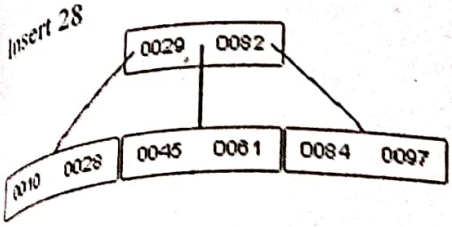
Insert 10



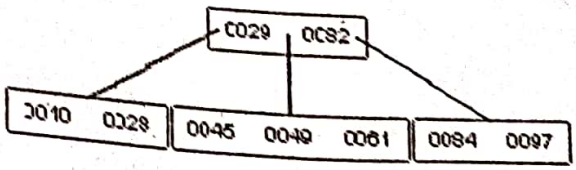
Insert 45



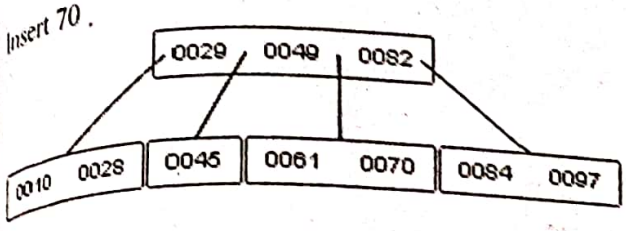
Insert 28



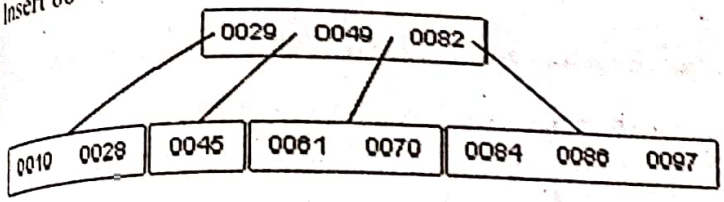
Insert 49



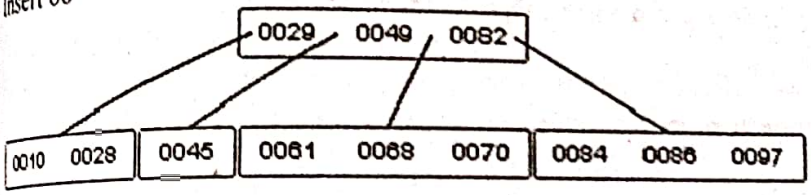
Insert 70 .



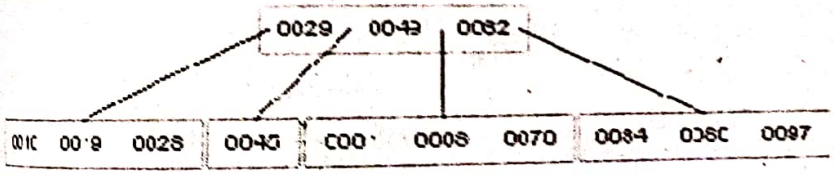
Insert 86



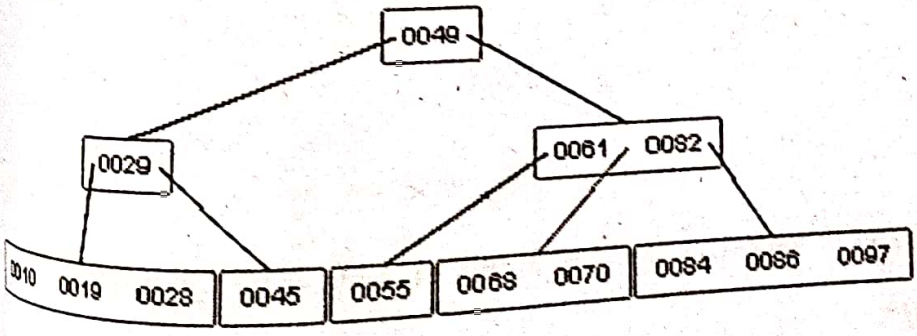
Insert 68



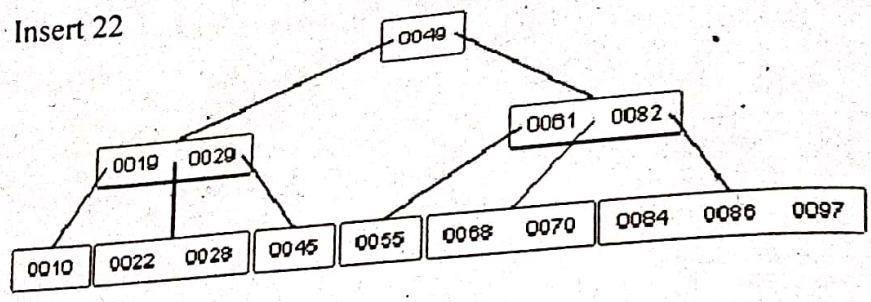
Insert 19



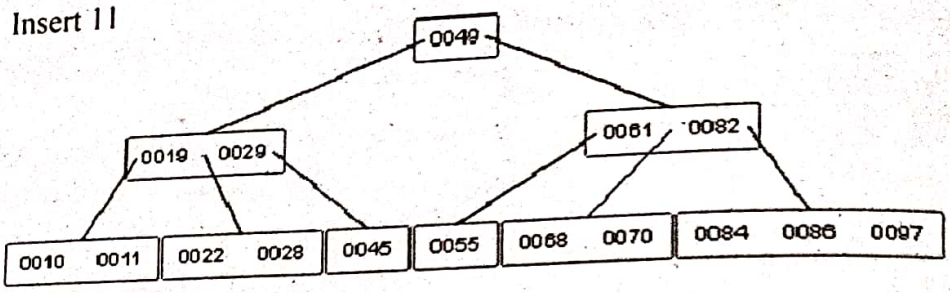
Insert 55



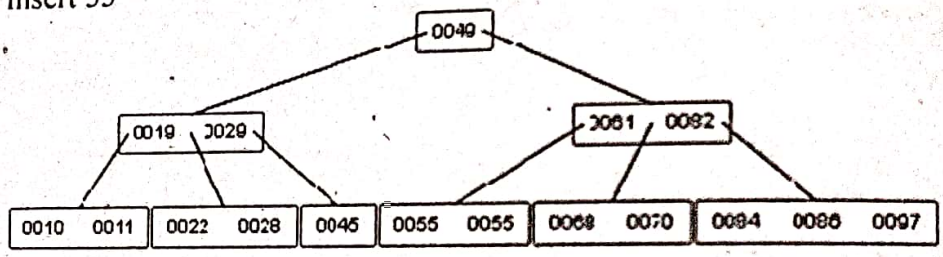
Insert 22



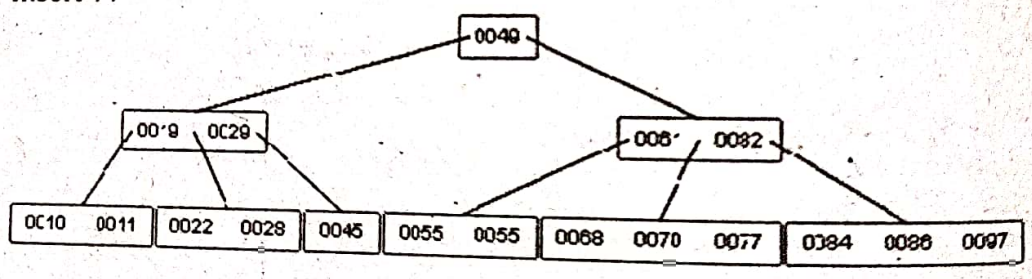
Insert 11



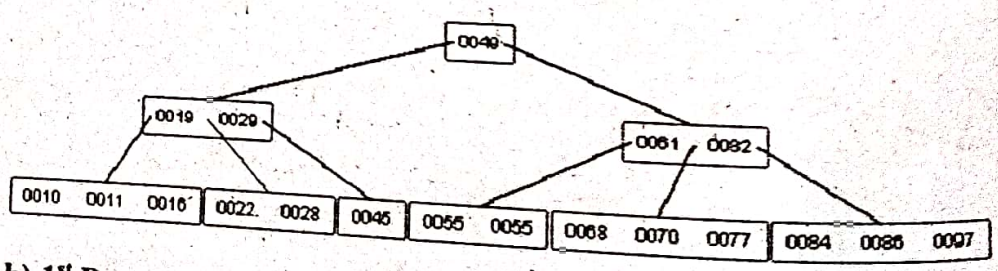
Insert 55



Insert 77



Insert 16



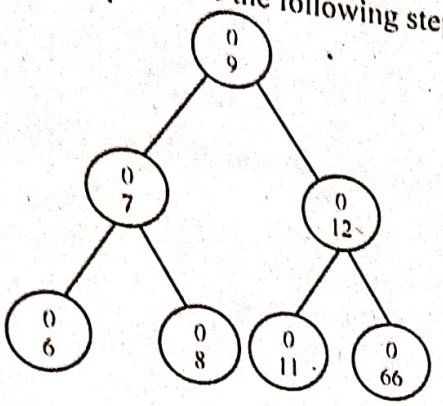
b) 1st Part: ***Refer to Question No. 6(b) of Long Answer Type Questions.***

2nd Part:

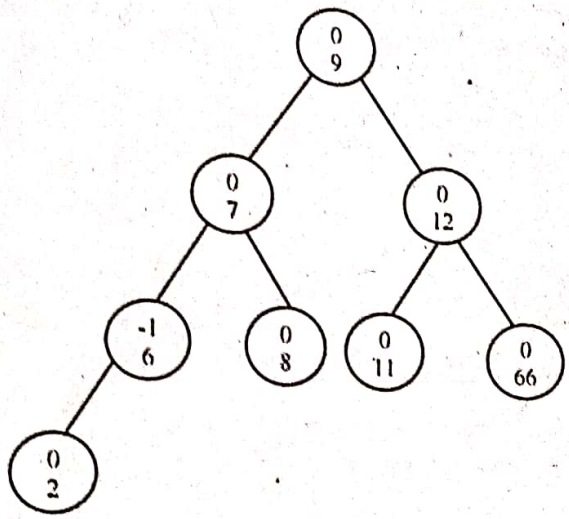
Up to insertion of 8 12 9 11 7 6:

Refer to Question No. 6(c) of Long Answer Type Questions.

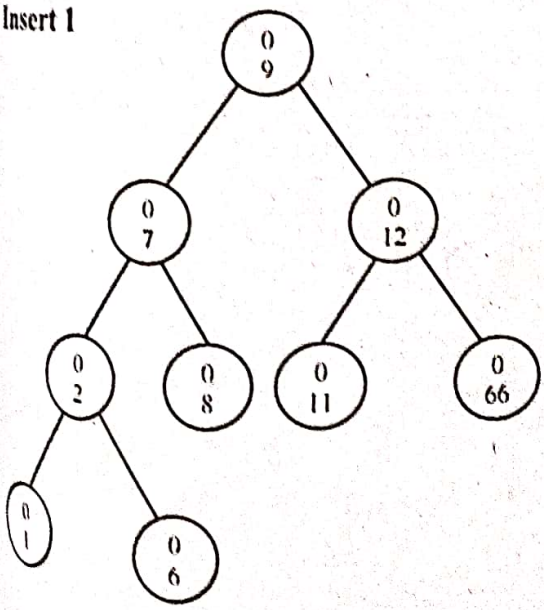
For insertion of 66, 2, 1, 44 in the above tree the following steps are needed.



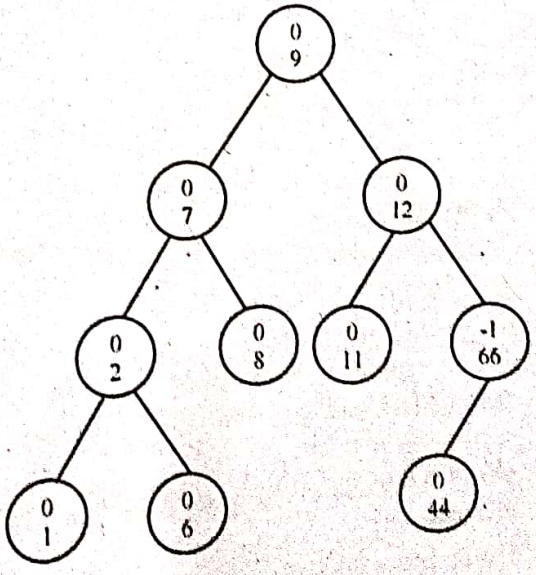
Insert 2



Insert 1



Insert 44



12. a) The in-order and pre-order traversal sequence of nodes in a binary tree are given below:

Post-order:	I	E	J	F	C	G	K	L	H	D	B	A
In-order:	E	I	C	F	J	B	G	.D	K	H	L	A

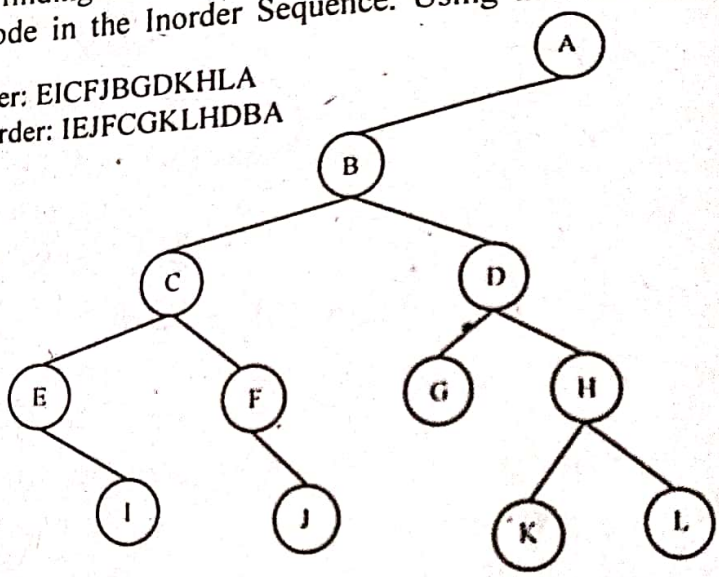
Construct the tree.

b) Write an algorithm for inserting an element into a Binary tree with example. **[WBUT 2015]**

Answer:

a) The last node in the postorder sequence is the root. We traverse right to left in the postorder sequence, finding each node's position (left or right) with respect to the previously located node in the Inorder Sequence. Using this we construct the tree as shown below.

inorder: EICFJBGDKHLA
postorder: IEJFCGKLHDBA



b) The following C program does the insertion into a binary tree

```
/* Structure */
struct bin_tree {
 int data;
 struct bin_tree * right, * left;
 };
 typedef struct bin_tree node;
/*insert function*/
1 void insert(node ** tree, int val) {
2 node *temp = NULL;
3 if(!(*tree)) {
4  temp = (node *)malloc(sizeof(node));
5  temp->left = temp->right = NULL;
6  temp->data = val;
7  *tree = temp;
8  return;
9 }
10
```



```
11 if(val < (*tr
12    insert(&
13  } else if(va
14    insert(&(
15  }
16 }
```

Explanation:
[Lines 3-9] Ch
[Line 11] Chec
- a. [Lin
- b. [Lin

[Line 13] Chec
- a. [Lin node
- b. [Lin

13. a) Write a binary search
b) Given the reconstruct th
Answer:
a) Assume the

```
struct node
{
struct node
int info;
struct node
};
```

/*Min function

```
struct node
{
if(ptr==NULL
return NULL;
else if(ptr-
        retu
else
return Min(p
}/*End of mi
```

/*Max function

```
struct node
{
if(ptr==NULL
return NULL;
```

```
11 if(val < (*tree)->data) {
12    insert(&(*tree)->left, val);
13  } else if(val > (*tree)->data) {
14    insert(&(*tree)->right, val);
15  }
16 }
```

Explanation:

[Lines 3-9] Check first if tree is empty, then insert node as root.

[Line 11] Check if node value to be inserted is lesser than root node value, then

- a. [Line 12] Call insert() function recursively while there is non-NULL left node
- b. [Lines 3-9] When reached to leftmost node as NULL, insert new node.

[Line 13] Check if node value to be inserted is greater than root node value, then

- a. [Line 14] Call insert() function recursively while there is non-NULL right node
- b. [Lines 3-9] When reached to rightmost node as NULL, insert new node.

13. a) Write a C function to find out the maximum and the minimum elements in a binary search tree.

b) Given the pre-order sequence and the post-order sequence, why cannot you reconstruct the tree? **[WBUT 2016]**

Answer:

a) Assume the binary search tree nodes are defined as follows:

```
struct node
{
struct node *lchild;
int info;
struct node *rchild;
};
```

/*Min function -recursive */

```
struct node *Min(struct node *ptr)
{
if(ptr==NULL)
return NULL;
else if(ptr->lchild==NULL)
      return ptr;
else
return Min(ptr->lchild);
}/*End of min()*/
```

/*Max function -recursive */

```
struct node *Max(struct node *ptr)
{
if(ptr==NULL)
return NULL;
```

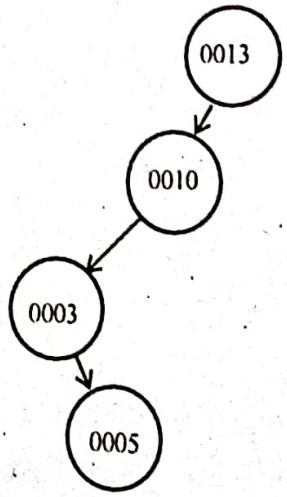
```
else if(ptr->rchild==NULL)
        return ptr;
else
return Max(ptr->rchild);
}/*End of max()*/
```

b) It is not possible to construct a binary tree just from pre and post order traversals of a binary tree because only inorder traversals give the position of the sub-tree elements with respect to the root.
If one has inorder along with pre or post order one can always construct a unique binary tree because the pre/post order information gives you the root of the tree. Once the root of the tree is known one can recursively construct the left and right sub-trees which are binary trees themselves thereby making the final product unique.

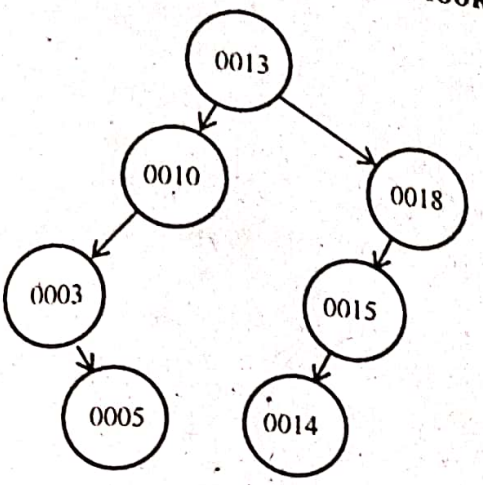
14. a) What do you mean by a binary search tree?
b) Construct a binary search tree by inserting the list of elements one by one:
13, 10, 3, 5, 18, 15, 14
c) Write an algorithm for pre-order traversal of a tree represented by a linked-list.
[WBUT 2017]

Answer:
a) A binary search tree is a binary tree that is either empty or in which each node satisfies the following conditions:
i) The left child has a value smaller than the parent
ii) The right child has a value greater than the parent.

b) After inserting 13, 10, 3, 5 the tree looks like



Now, after inserting 18, 15, 14 the tree looks like



c) //C program implementation

```
/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Given a binary tree, print its nodes in preorder*/
void printPreorder(struct node* node)
{
    if (node == NULL)
        return;

    /* first print data of node */
    printf("%d ", node->data);

    /* then recur on left subtree */
    printPreorder(node->left);
    /* now recur on right subtree */
    printPreorder(node->right);
}
```

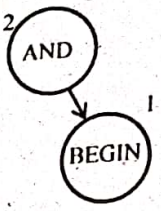
[WBUT 2017]

5. a) What is an AVL tree?
b) Construct an AVL search tree for the data list:
AND, BEGIN, CASE, DO, END, FOR GOTO.
c) For the AVL tree you have constructed delete the following keys in the order:
DO, FOR END.

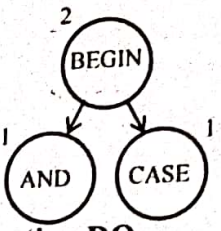
Answer:

a) Refer to *Question No. 16(a) of Long Answer Type Questions.*

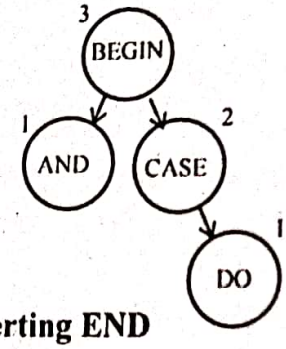
b) After inserting AND and BEGIN:



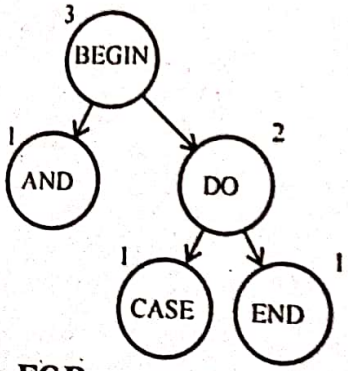
After inserting CASE



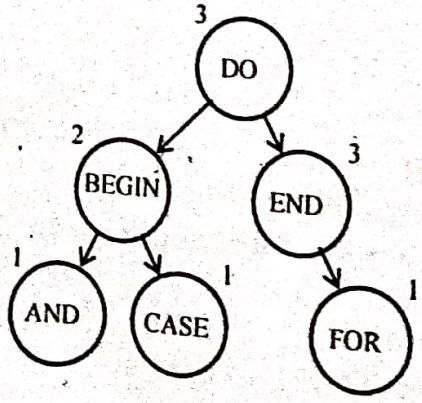
After inserting DO



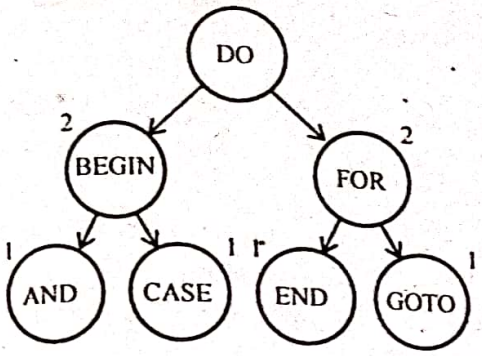
After inserting END



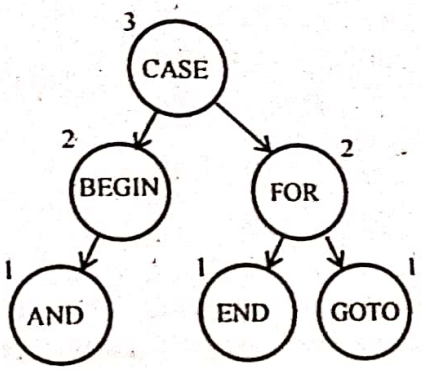
After inserting FOR



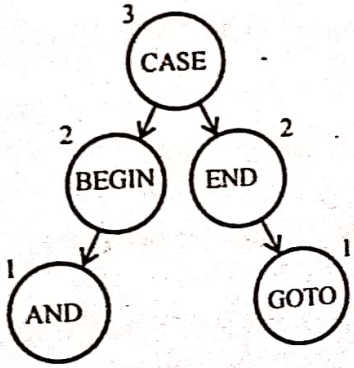
After inserting GOTO



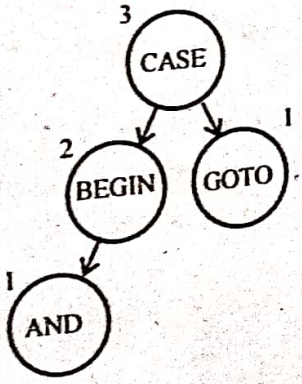
c) After removing DO



After removing FOR



After removing END



16. Write short notes on the following: [WBUT 2013, 2015, 2018] [WBUT 2010, 2014, 2015] [WBUT 2012, 2013] [WBUT 2018]

a) AVL Tree
b) Threaded Binary Tree
c) B – tree
d) Binary Search Tree

Answer:

a) AVL Tree:

An AVL Tree is a form of binary tree, however unlike a binary tree, the worst case scenario for a search is O (log n). The AVL data structure achieves this property by placing restrictions on the difference in height between the sub-trees of a given node, and re-balancing the tree if it violates these restrictions.

The complexity of an AVL Tree comes from the balance requirements it enforces on each node. A node is only allowed to possess one of three possible states (or balance factors):

Left-High (balance factor -1)
The left-sub tree is one level taller than the right-sub tree

Balanced (balance factor 0)
The left and right sub-trees are both the same heights

Right-High (balance factor +1)
The right sub-tree is one level taller than the left-sub tree.

If the balance of a node becomes -2 (it was left high and a level was lost from the left sub-tree) or +2 (it was right high and a level was lost from the right sub-tree) it will require re-balancing. This is achieved by performing a rotation about this node.

b) Threaded Binary Tree:

Refer to Question No. 3(b) of Long Answer Type Questions.

c) B – tree:

Definition: A B-Tree of order m is an m-way tree in which

i) All leaves are on the same level.

ii) All internal nodes except the root have at most m nonempty children, and at least $\lceil m/2 \rceil$ nonempty children.

iii) The number of keys in each internal node is one less than the number of its nonempty children, and these keys partition the keys in the children in the fashion of a search tree.

iv) The root has at most m children, but may have as few as 2 if it is not a leaf, or none if the tree consists of the root alone.

There are some disadvantages also

- The nodes of a B-tree do not normally contain the same number of value entries.
- They normally do contain a certain amount of free space.
- Difficulty in traversing the keys sequentially.

Note: B-tree is not a binary tree. A B-tree is also known as the balanced sort tree.

d) Binary Search Tree:

Refer to Question No. 14(a) & (b) of Long Answer Type Questions.

Chapt

- A graph is model netw where the
- Types of g
  1. Undirec
  2. Directed
- Graph Tra
  Depth first
  Breadth fi
  tree.
- Spanning T graph which different sp representing computing t
- Shortest Pa between two edges is min The most im Dijkstra's a Bellman-Fo A* search al the search.

1. The vertex, r
   a) Pendant
   c) articulati
   Answer: (c)

2. A vertex of in
   a) articulatio
   c) isolated v
   Answer: (c)

3. Adjacency m
   a) identity
   c) asymmetr
   Answer: (b)