Q.1 With the help of a diagram, state the algorithm:
1. Rotate Left
2. Rotate Right
Ans :
Algorithm for Rotate Right:
Algorithm rotateRight (ref root <tree pointer>)
This algorithm exchanges pointers to rotate the tree right
Pre : roots points to the tree to be rotated
Post :Node rotated and root updated
1 tempPtr = root->left
2 root -> left = tempPtr -> right
3 tempPtr -> right = root
4 root = tempPtr
5 return
endrotateRight
Algorithm for Rotate Left:
Algorithm rotateLeft (ref root <tree pointer>)
This algorithm exchanges pointers to rotate the tree left
Pre : roots points to the tree to be rotated
Post :Node rotated and root updated
1 tempPtr = root->right
2 root -> right = tempPtr -> left
3 tempPtr -> left = root
4 root = tempPtr
5 return
endrotateLeft

Example:

The following tree is an example of Right of Left case & Complex double rotation right subcase.

Original Tree:

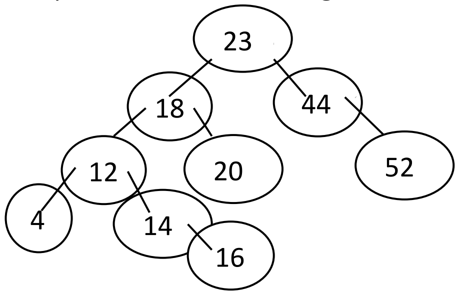
This is imbalanced tree.

$H_L = 4$, $H_R = 2$

Therefore, $H_L - H_R = 2$

Effecting node =12

Therefore, rotating 12 node to left



After Left Rotation:

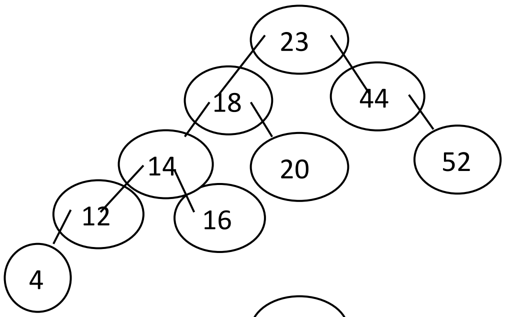
This is imbalanced tree.

$H_L = 4$, $H_R = 2$

Therefore, $H_L - H_R = 2$

Effecting node =18

Therefore, rotating 18 node to right

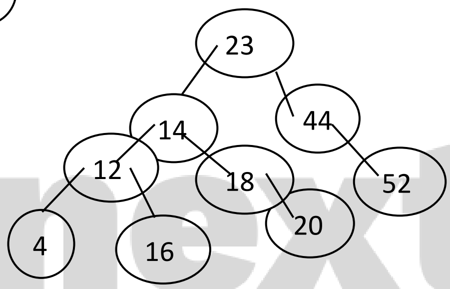


After Right Rotation:

This is balanced tree.

$H_L = 3$, $H_R = 2$

Therefore, $H_L - H_R = 1$

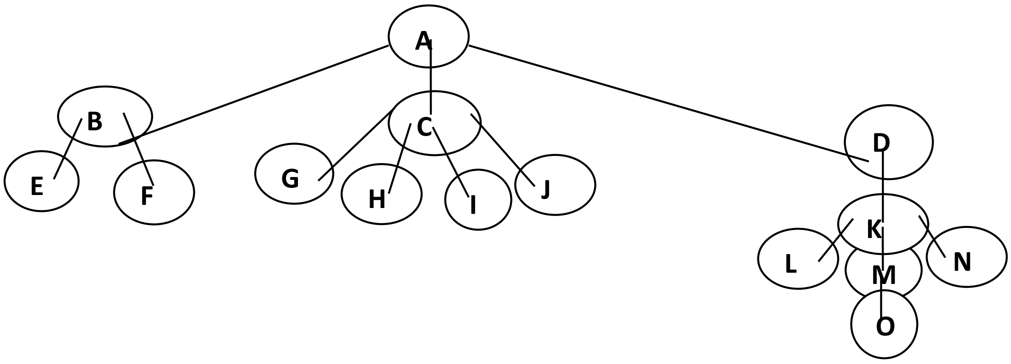


Q.2 In what way is an AVL Tree efficient that a Binary Search Tree? Why is it called a Height Balanced Tree?
Ans :
AVL Tree :-
The balanced binary tree structure called the AVL Tree, was designed and named after two Russian mathematicians AdelsonVelskii and Landis.
An AVL Tree is a search tree in which the heights of the subtrees differ by no more than 1. It is thus a balanced binary tree. Of course this advantage comes at a cost of rotating a node each time to maintain the balance. This is time consuming.

AVL Trees are sufficient than a BST :-
When the keys come in ordered sequence, the binary search tree can degenerate into a linked list, thereby greatly increasing the search time, whereas an AVL Tree will still be balanced by virtue of its ability to rotate an imbalanced node.

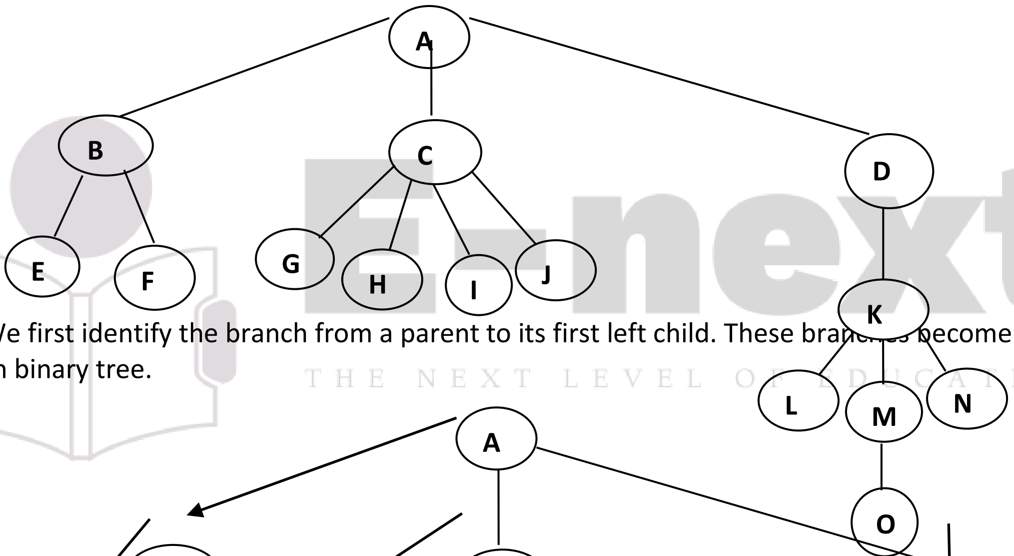
AVL Tree, an Height Balanced Tree :-
Since the heights of subtrees of every node in an AVL tree differ by no more than 1, the height of the subtree is greatly reduced, thereby shortening the search time. Hence the AVL trees are also known as "Height Balanced Trees".

Q.3 Give the definition of a general tree. What are the steps to convert a general tree to a binary tree? Implement the conversions on the given general tree.

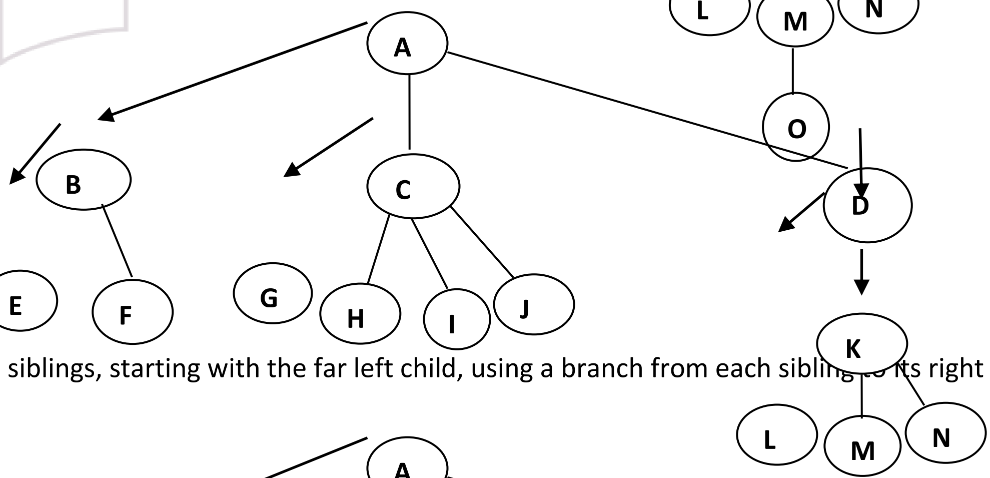


Ans:

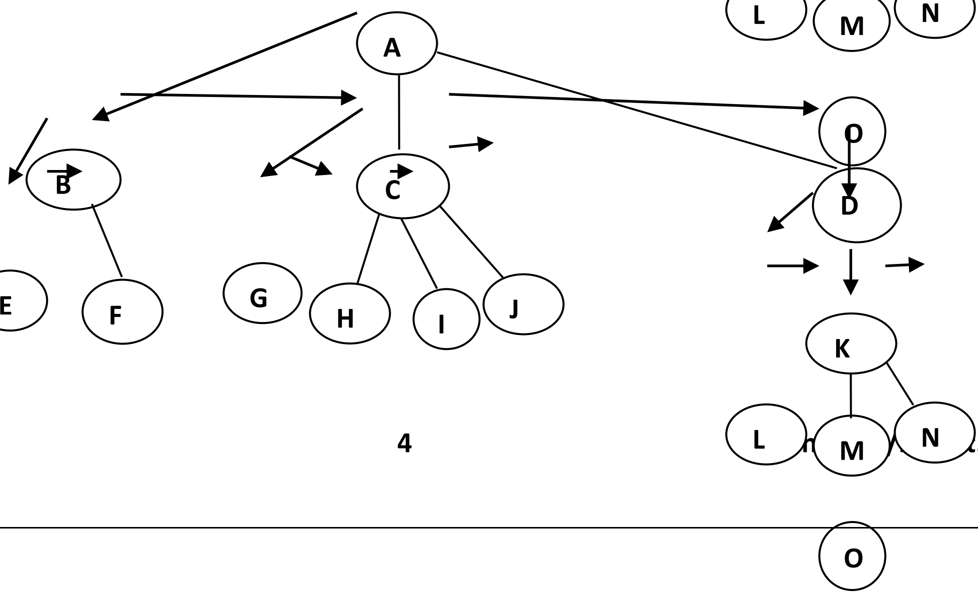
General Tree:
A general tree is a tree in which each node can have an unlimited out degree.
Changing a General Tree to a Binary Tree:
<u>A General Tree</u>



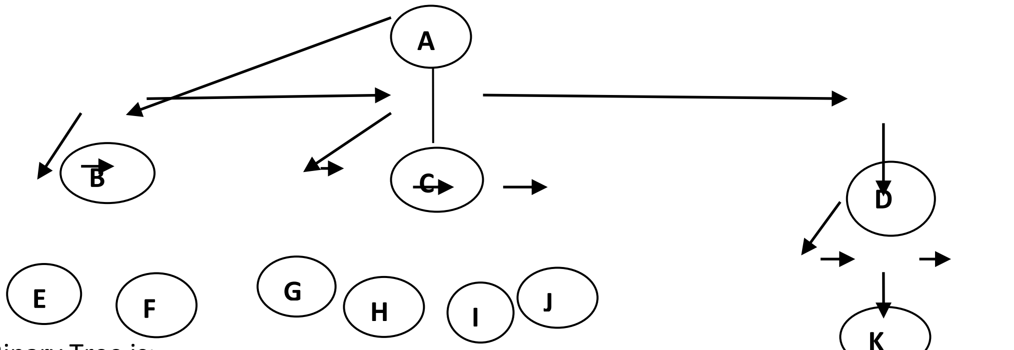
<u>Step 1</u>: We first identify the branch from a parent to its first left child. These branches become the left pointer in binary tree.



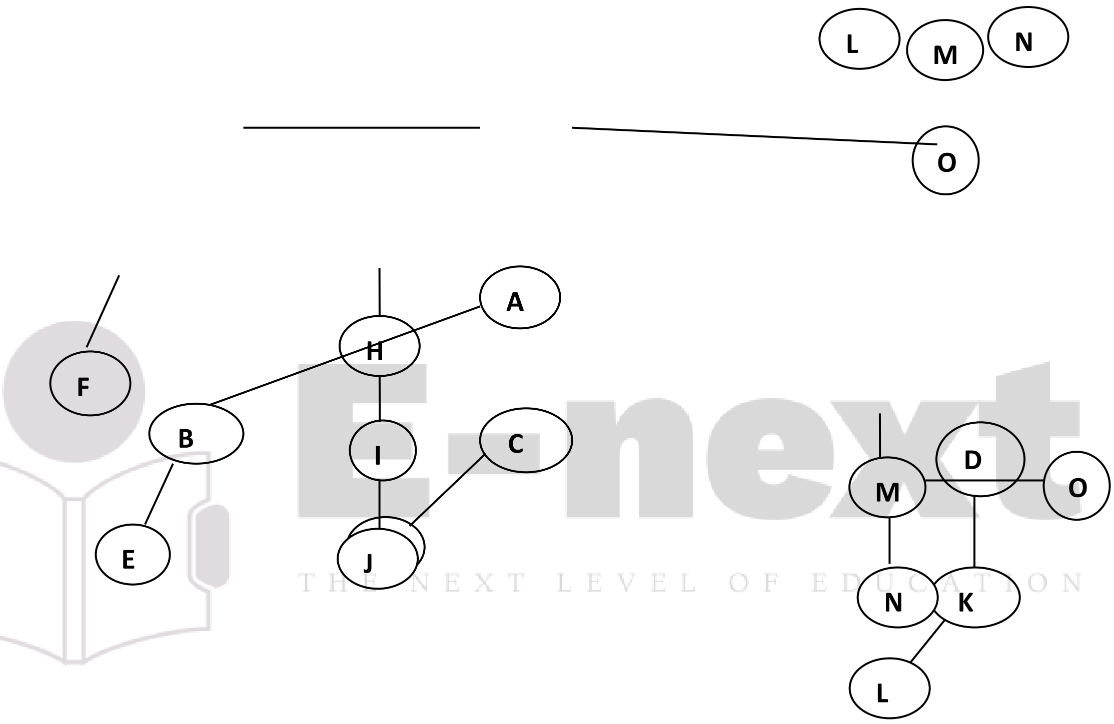
<u>Step 2</u>: Connects siblings, starting with the far left child, using a branch from each sibling to its right sibling.



Step 3: Remove all unneeded branches from the parent to its children.



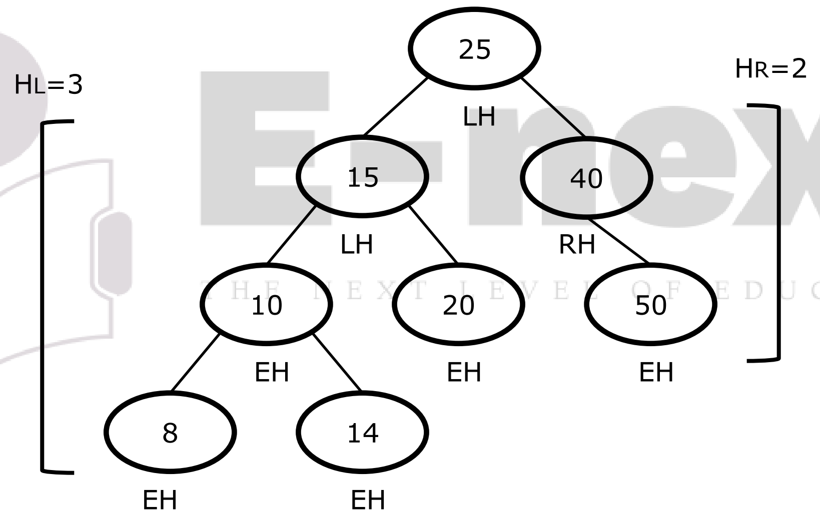
The resulting Binary Tree is:



Q.Define properties of AVL tree.

a) A AVL tree is a tree in which no nodes can have more than two subtrees, the maximum outdegree for a node is two.
b) These subtree are designated by LEFT subtree and the RIGHT subtree.
c) Each subtree is a AVL tree itself.
d) All key values at LEFT side will be less than node n RIGHT side keys are always greater than or equal to value of node.
e) An AVL tree is a binary search tree in which the balance factor of every node, which is defined as the difference between the heights of the node's left and right subtrees, is either 0 or + 1 or -1.
f) The balance factor of each node in an AVL tree is calculated as , the height of its left subtree minus the height of its right subtree.
   i.e. $|H_L-H_R|<=1$
g) If this difference is more than one, then it must so through a rotation so that the difference in their heights is not more than 1.
h) The node with a balance factor of:
i) 1 is called left high or LH
   0 is called even high or EH
   -1 is called right high or RH

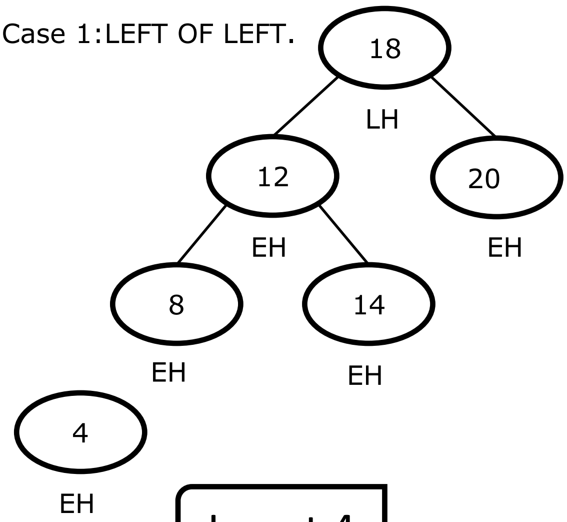
Example of AVL tree :

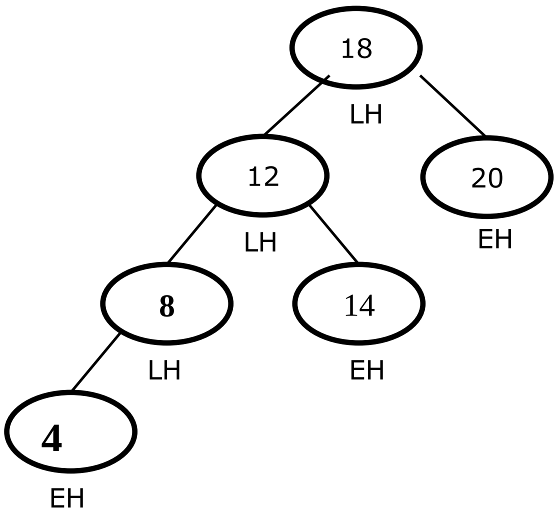


Tree Balanced $H_L-H_R=1$

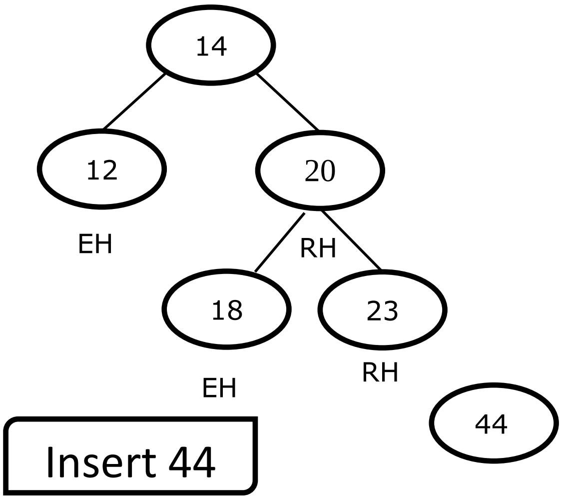
Q. define 4 major cases and 8 sub cases of AVL tree

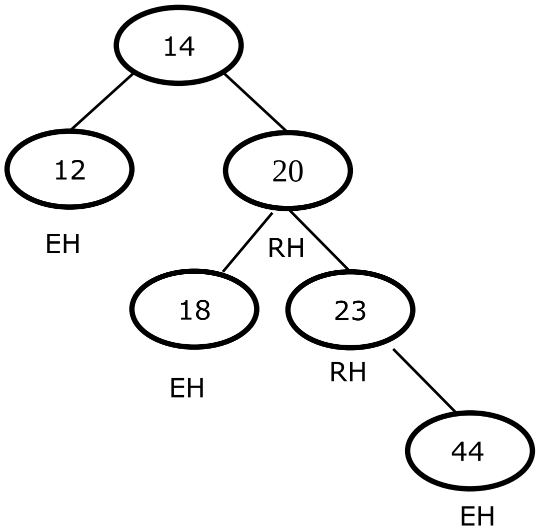
Case 1:LEFT OF LEFT.



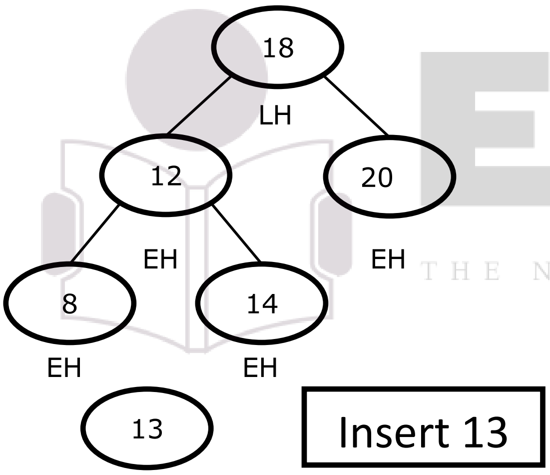


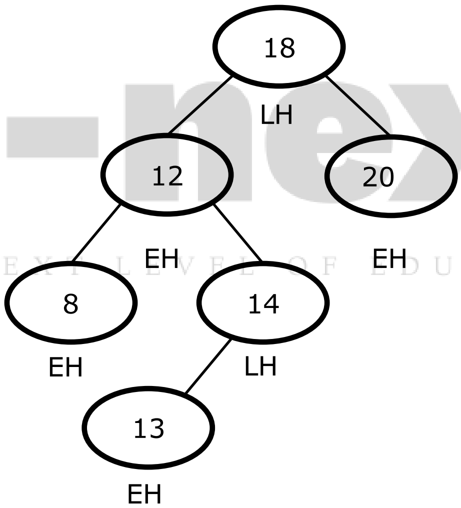
CASE 2:RIGHT OF RIGHT



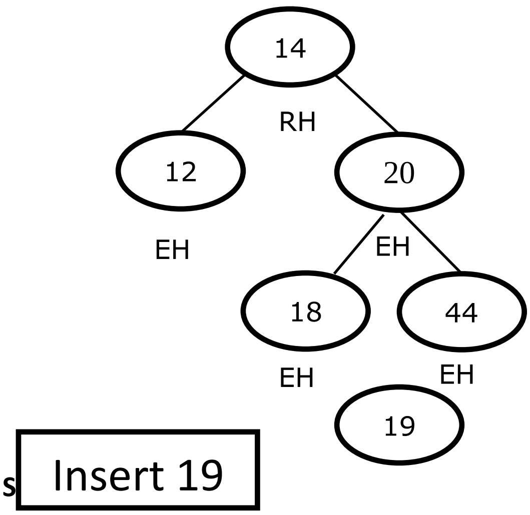


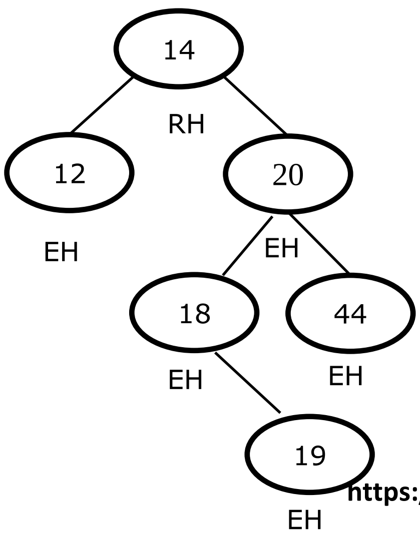
CASE 3:RIGHT OF LEFT



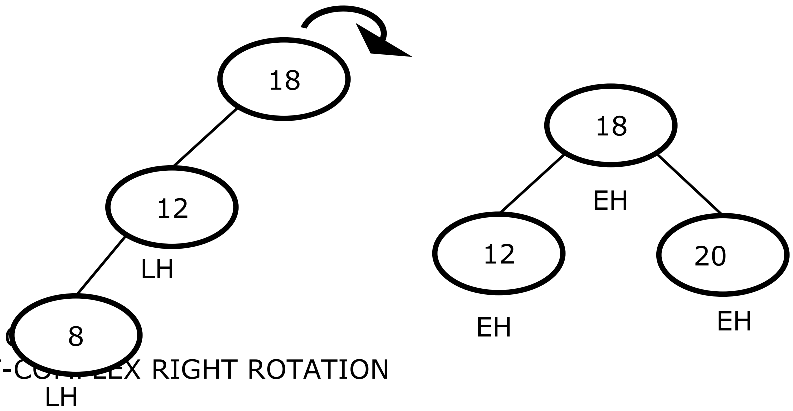


CASE 4:LEFTOF RIGHT

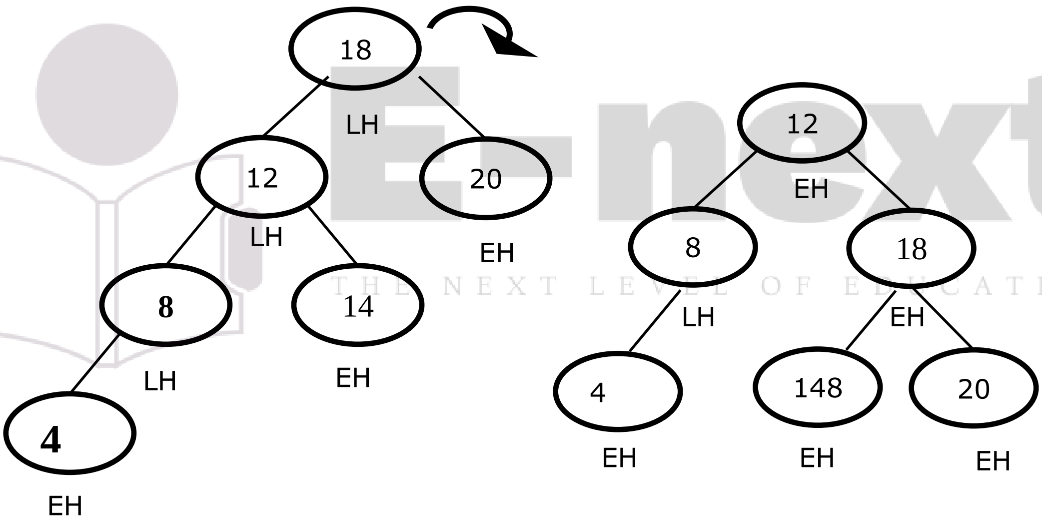




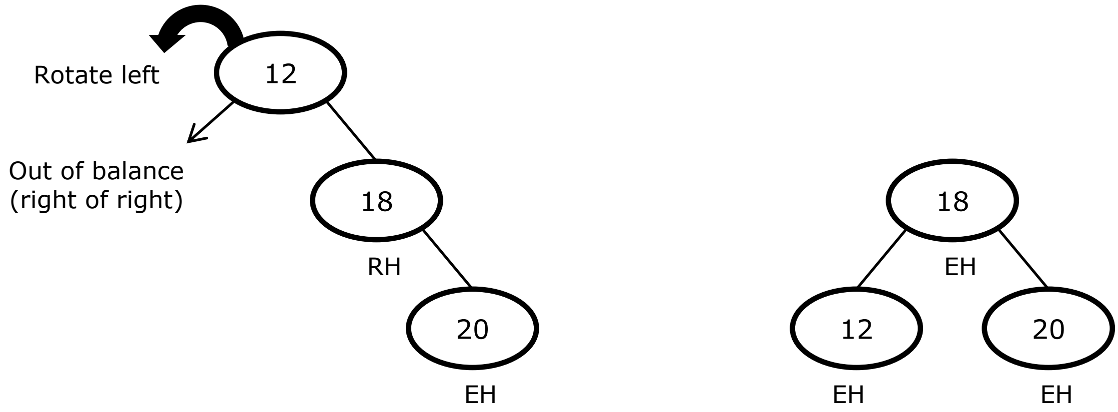
CASE 1: SUB CASE A)
LEFT OF LEFT-SIMPLE RIGHT ROTATION



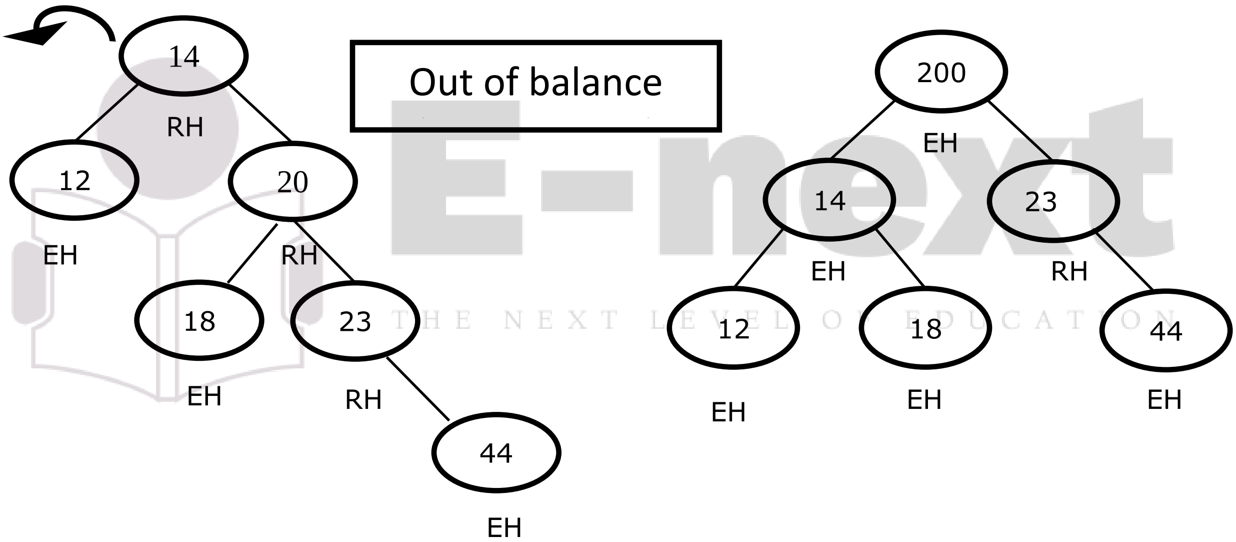
CASE 1: SUB C
LEFT OF LEFT-COMPLEX RIGHT ROTATION



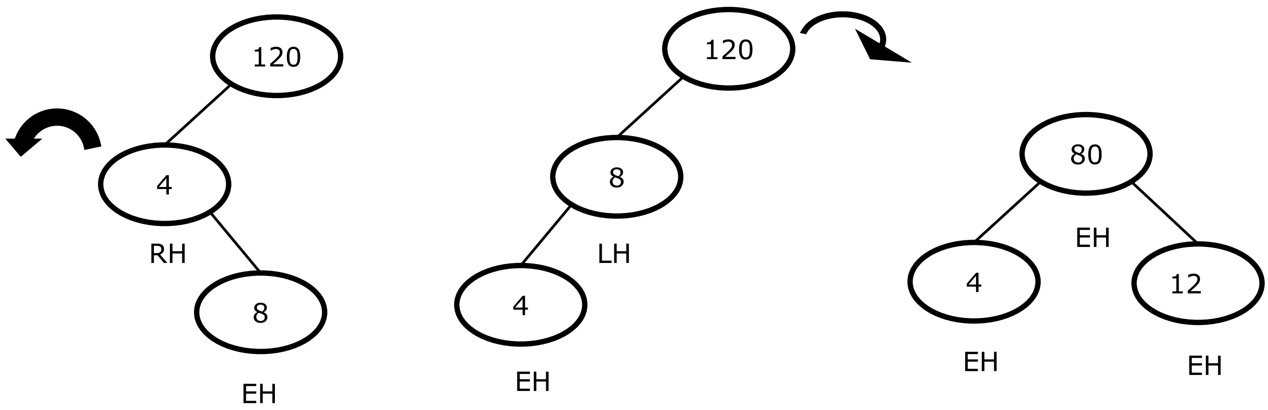
CASE 2: SUB CASE A)
LEFT OF LEFT-SIMPLE RIGHT ROTATION



CASE 2: SUB CASE B)
LEFT OF LEFT-COMPLEX RIGHT ROTATION

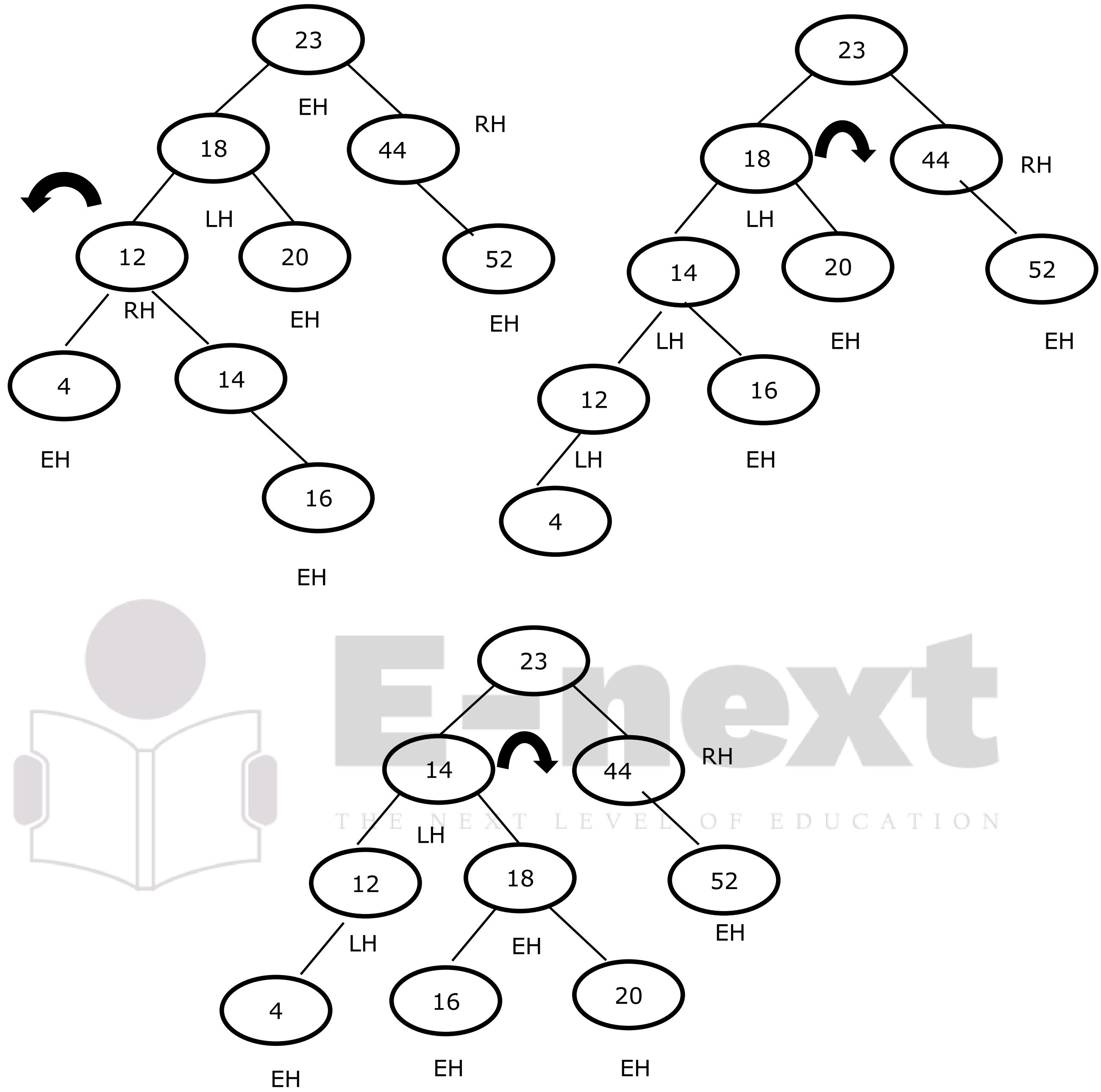


CASE 3: SUB CASE A)
LEFT OF LEFT-SIMPLE RIGHT ROTATION

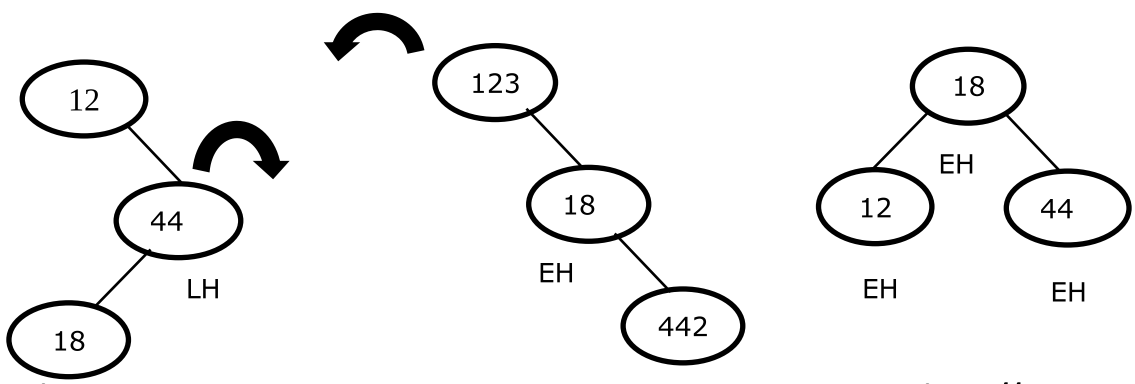


CASE 3: SUB CASE A)

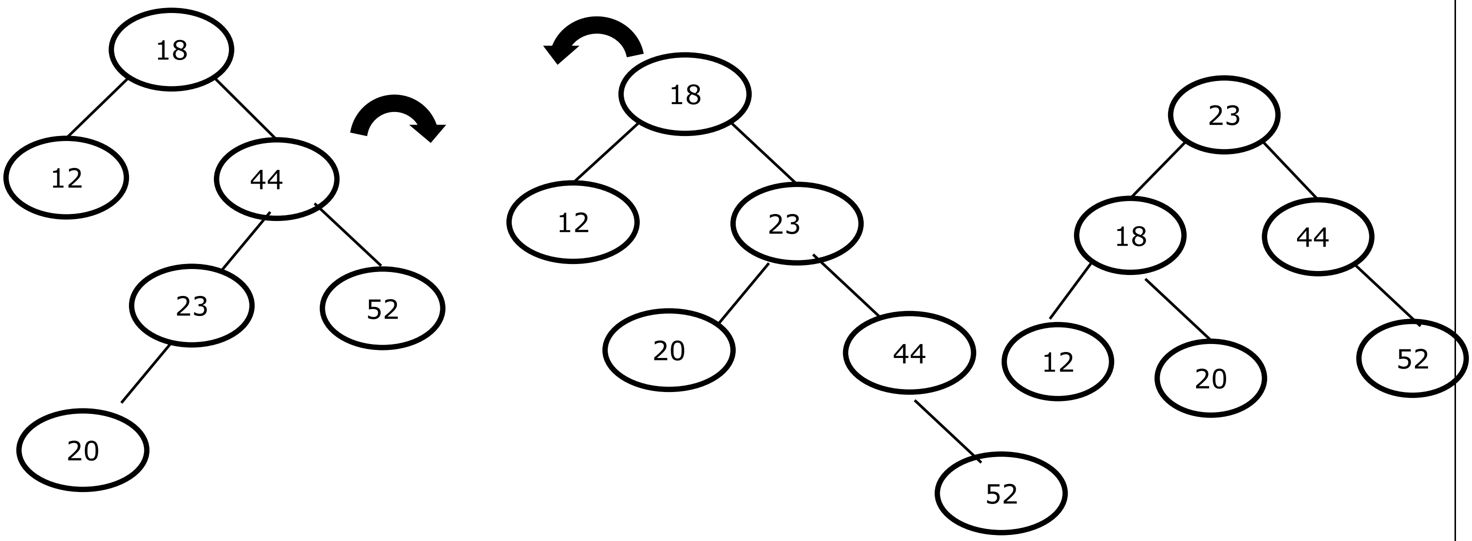
LEFT OF LEFT-COMPLEX RIGHT ROTATION



CASE 4: SUB CASE A)
LEFT OF LEFT-SIMPLE RIGHT ROTATION



CASE 4: SUB CASE B)
LEFT OF LEFT-SIMPLE RIGHT ROTATION

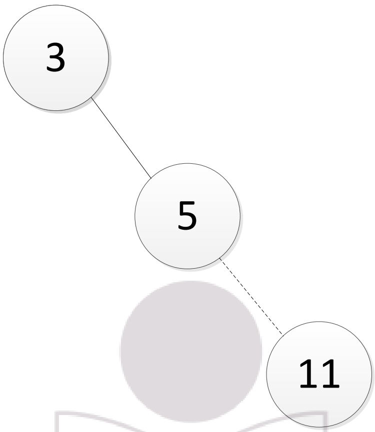


Q.What is meant by Balanced tree(AVL)? Draw an AVL tree for the following data arriving in sequence

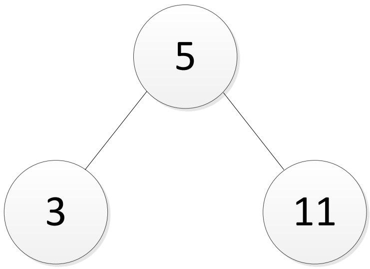
3, 5, 11, 8, 4, 1, 12, 7, 2, 6, 10

The balanced binary tree structure called the AVL tree, was designed and named after two Russian mathematicians **AdelsonVelskii and Landis(AVL).**
An AVL tree is a search tree in which the heights of the sub-trees differ by no more than 1. It is thus a **balanced binary tree.**
Since the heights of the sub-trees of every node in an AVL tree differ by no more than 1, the height of the tree is greatly reduced, thereby shortening the search time. Hence they are also known as **height balanced trees.**

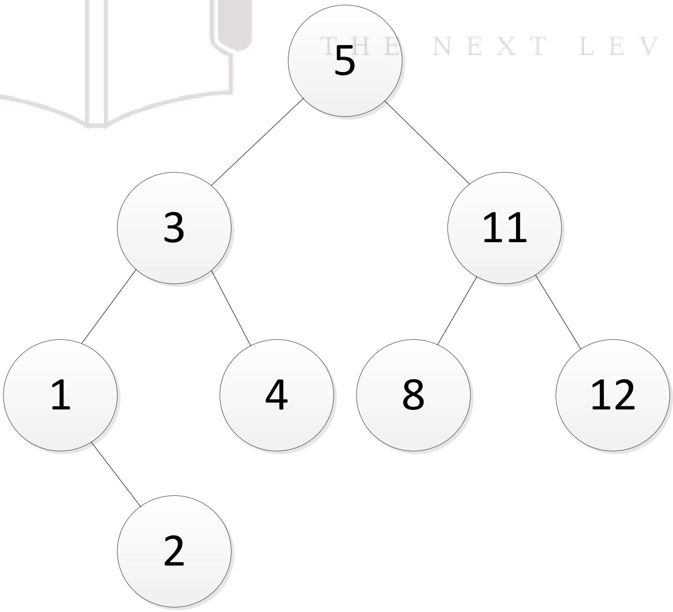
Step 1:



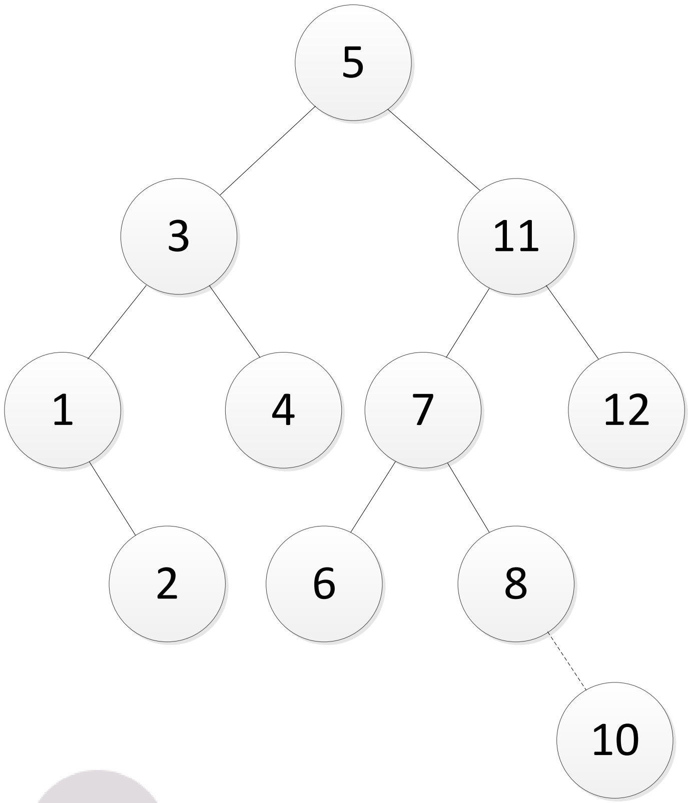
Step 2:



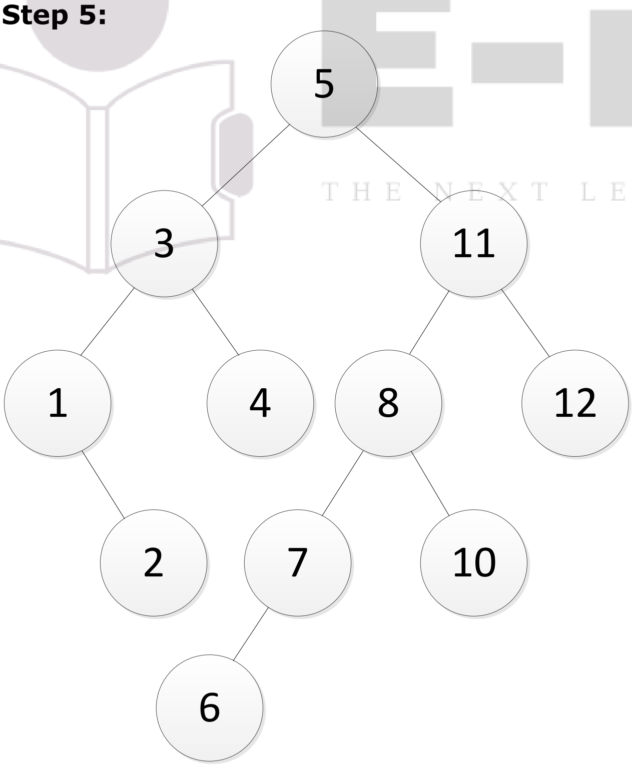
Step 3:



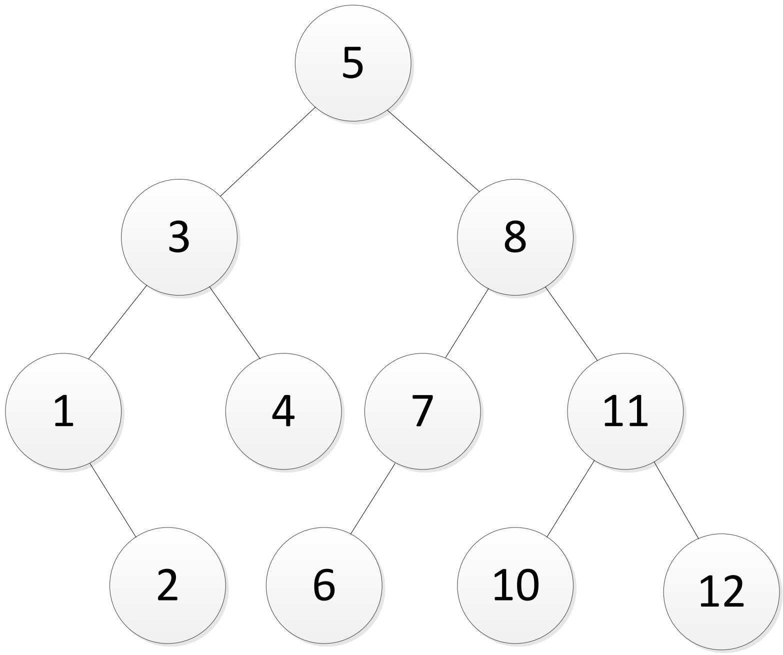
Step 4:



Step 5:

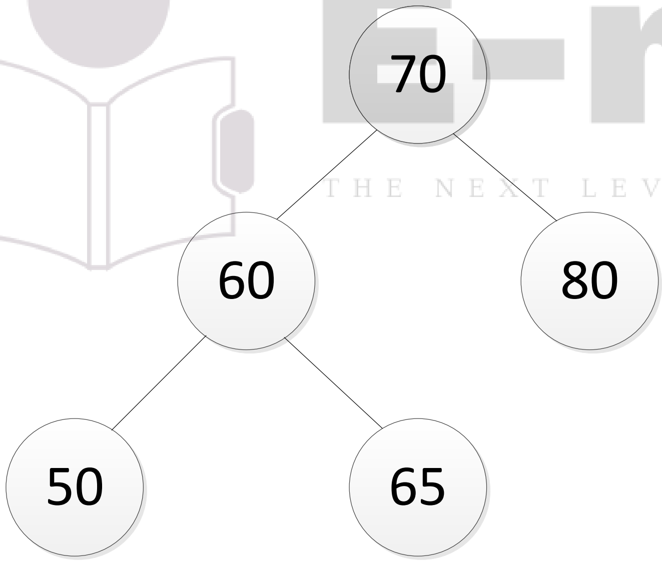


Step 6:

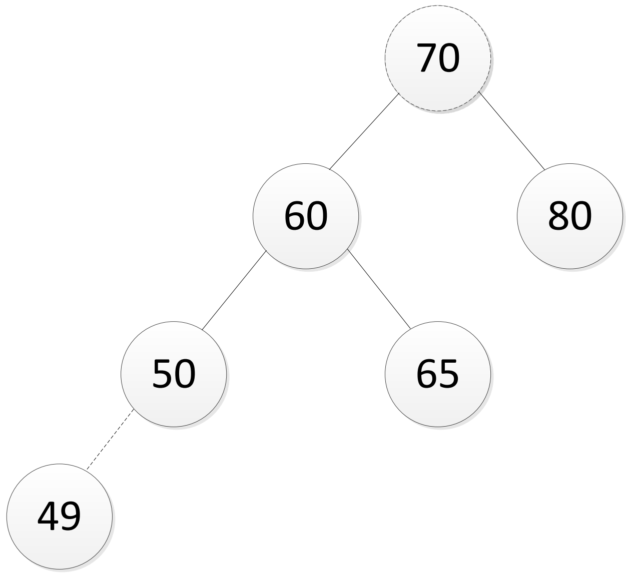


1. **Crate an AVL tree for the following data occurring in sequence:**

   70, 60, 80, 50, 65

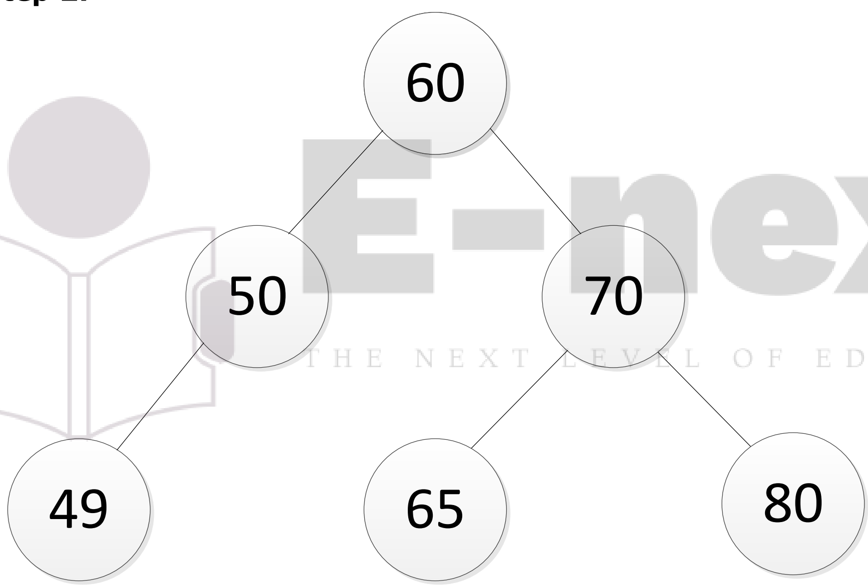


A] Now Insert 49 and convert it to an AVL tree clearly starting the main case, sub case and the necessary rotations.
Step 1:

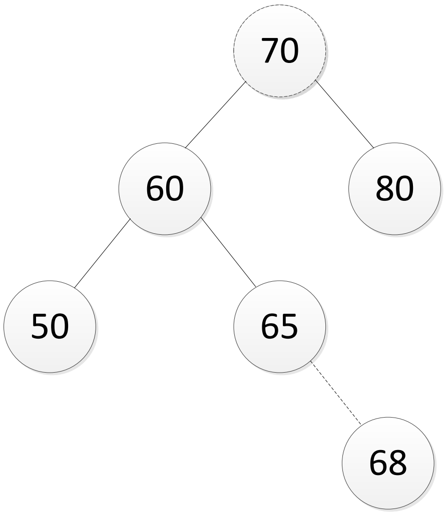


Here, 70 imbalanced
Left of Right
Complex Right of Right

Step 2:

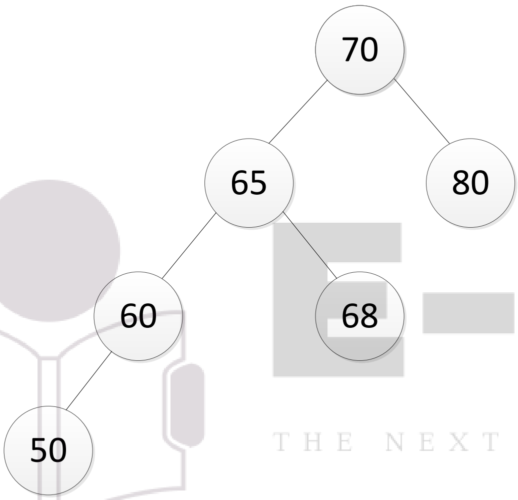


B] Now insert 68 to the original tree above and do the necessary rotations mentioning all the details.
Step 1:



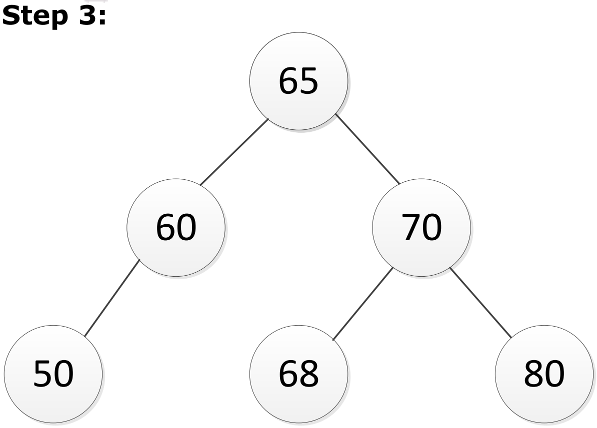
Here, 70 imbalanced
Right of Left
Simplex double rotation Right

Step 2:



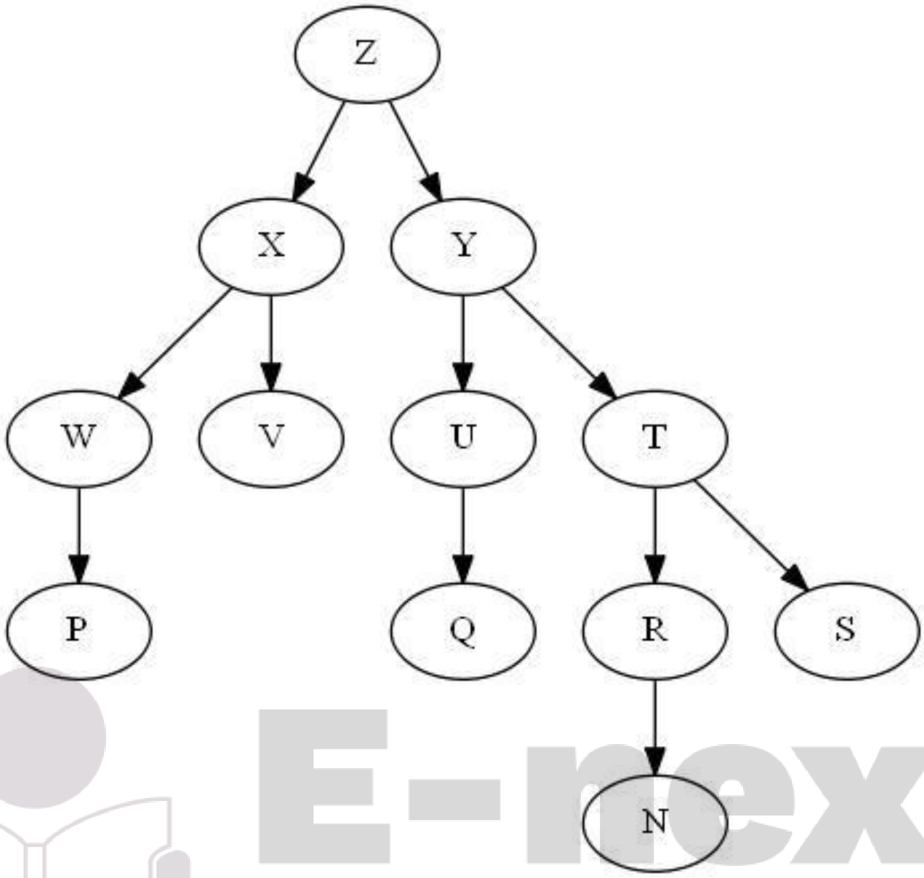
fter Left rotation

Step 3:



After Right rotation

May2011– Q6 B– 10 Marks
Q. Explain the preorder, postorderand inorder traversal of a tree with their algorithms. Givethe preorder, postorder and inorder listing ofthenodes of thefollowing trees.



Preorder:

Thedepth-first traversal method is called preorder traversal. Preorder traversal is defined recursively as follows. To do a preorder traversal of a generaltree:

1. Visit the root first;&then
2. do a preorder traversal each of thesubtrees ofthe root one-by-one inthe order given

Algorithm Prefix(Preorder)Tree Traversal:
Print the prefix expression for an expression tree.
Pre tree is a pointer to an expression tree.
Post thepostfix expression has beenprinted

1. if (treenotempty)
   i. print (tree token)
   ii. prefix (treeleft subtree)
   iii. prefix (treeright subtree)
2. end if

end postfix

Postorder:

In contrastwith preorder traversal, whichvisits theroot first, postorder traversal visits the root last. To do a postorder traversal of ageneraltree:

1. Do a postordertraversal each of thesubtrees of the root one-by-one in theorder given; &then
2. Visit the root.

Algorithm Postfix (Postorder)TreeTraversal:

Print the postfix expression for anexpression tree.
Pre tree is a pointer to an expression tree.
Post the postfix expression has beenprinted

1. if (treenotempty)
    i. postfix (tree left subtree)
    ii. postfix (tree right subtree)
    iii. print (tree token)
2. end if

end postfix

Inorder:

Inorder traversal only makes sensefor binarytrees. Whereas preorder traversal visits the root first and postorder traversalvisits theroot last, inorder traversalvisits the root in between visiting the left and right subtrees:

1. Traversethe left subtree; &then
2. Visit the root; &then
3. Traversethe right subtree.

Algorithm Infix(Inorder)TreeTraversal:

Print the infix expression for an expression tree.
Pre tree is a pointer to an expression tree.
Post thepostfix expression has beenprinted

1. if (treenotempty)
    i. prefix (treeleft subtree)
    ii. print (tree token)
    iii. prefix (treeright subtree)
2. end if

end infix

Notationsof the tree:

Preorder:Z X WPVYU QT RNS

Postorder:PWV XQU NRS TYZ

Inorder: PWX VZQ U YNRT S

May2011– Q5 B– 10 Marks

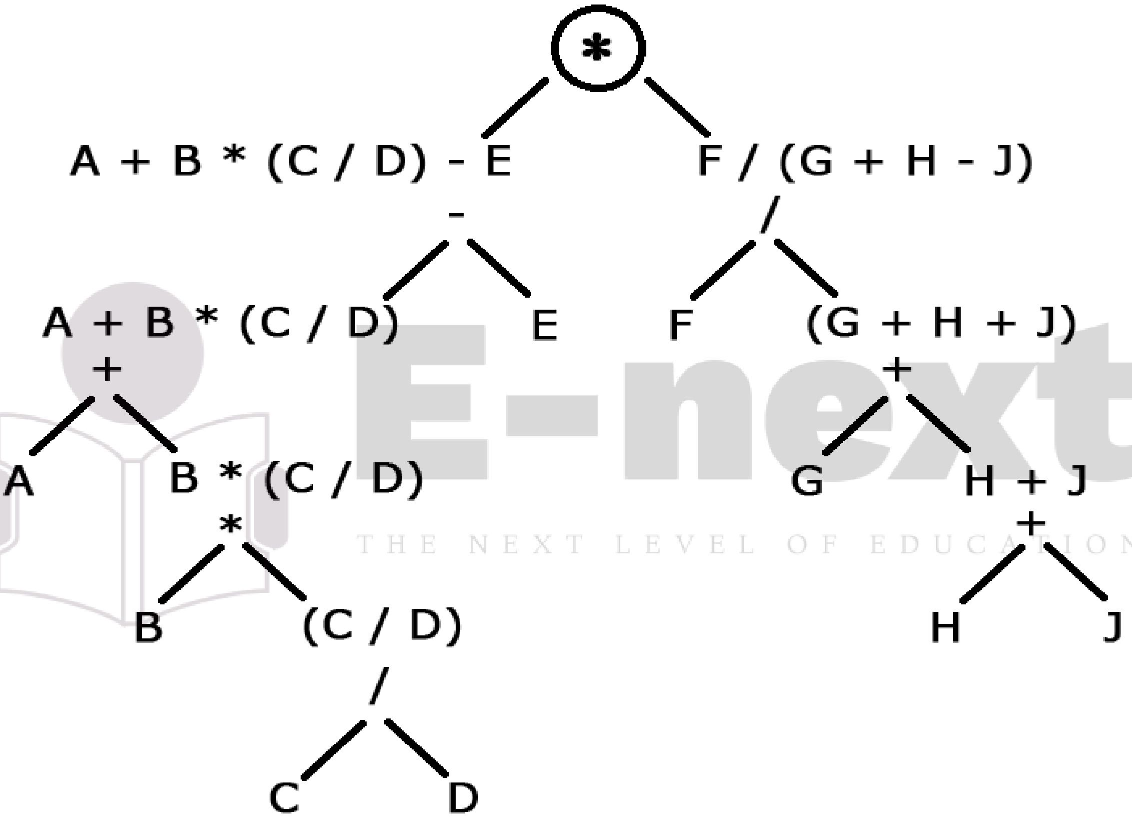
Q. Definean expression tree.Thefollowing infix expression is given.Drawthe expression tree and findtheprefix and thepostfix expression.
(A +B *(C/ D) –E)*(F /(G+H– J))

ExpressionTree:

An expression tree is aspecialkindof tree. Inthis thenodes contain one, two or zero children.

Some of the**properties**oftheExpression tree are:

- Each leaf is an operand.
- The root &the internalnodes areoperators
- Subtrees aresubexpressions with the root being an operator.

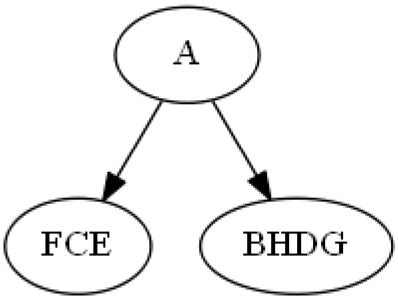


Prefix(Preorder) Expression: *** -+A * B/C D E/F + -GH J**

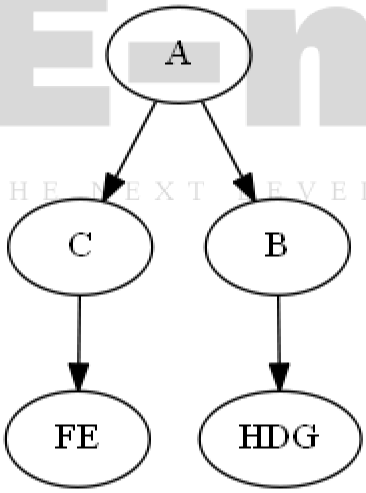
Postfix(Postorder) Expression: **A BCD /* + E– FGH J -+/***

Dec 2010– Q3A– 10 Marks
A binary treehas 8nodes. The inorder andthepostorder traversal of thetree is given below:
Postorder: F ECHGDBA
Inorder: F CEA BHD G
Show astepwise reconstruction ofthe binary tree along with its Preorder traversal

Step1:
By thepostorder expression wecan determinethe root. Inthis case, the root is A
Taking A asthe root&seeing theInorder expression, weget FCE atthe left of it &
BHD Gto its right.

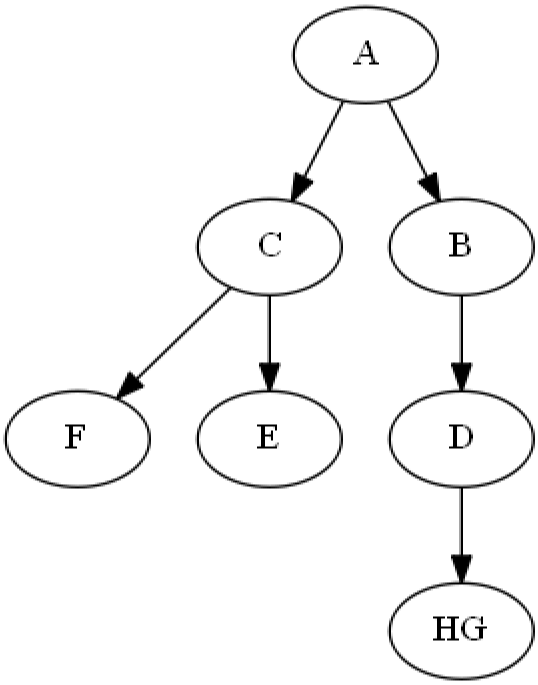


Step2:
Going further now,wewill get.



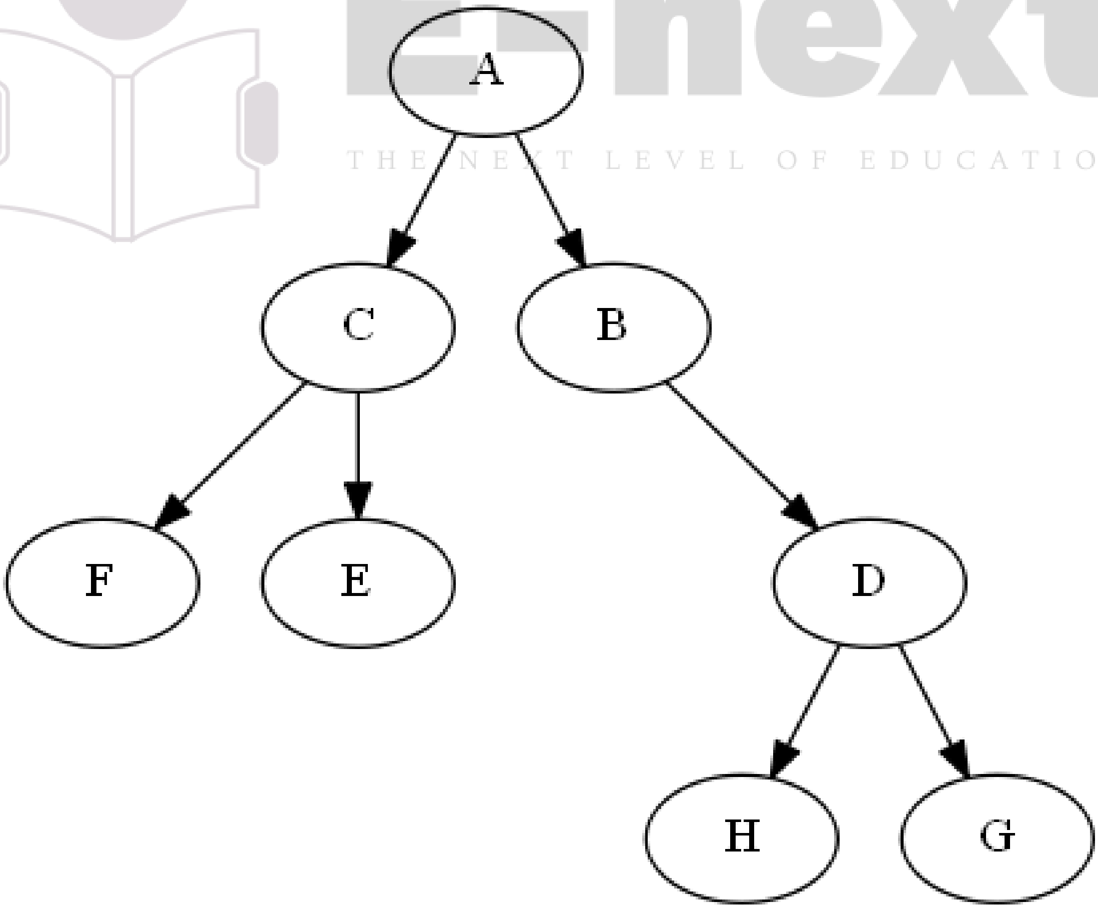
Step3:

Splitting thenodes in the left subtree&in theright subtree, we get.



Step4:

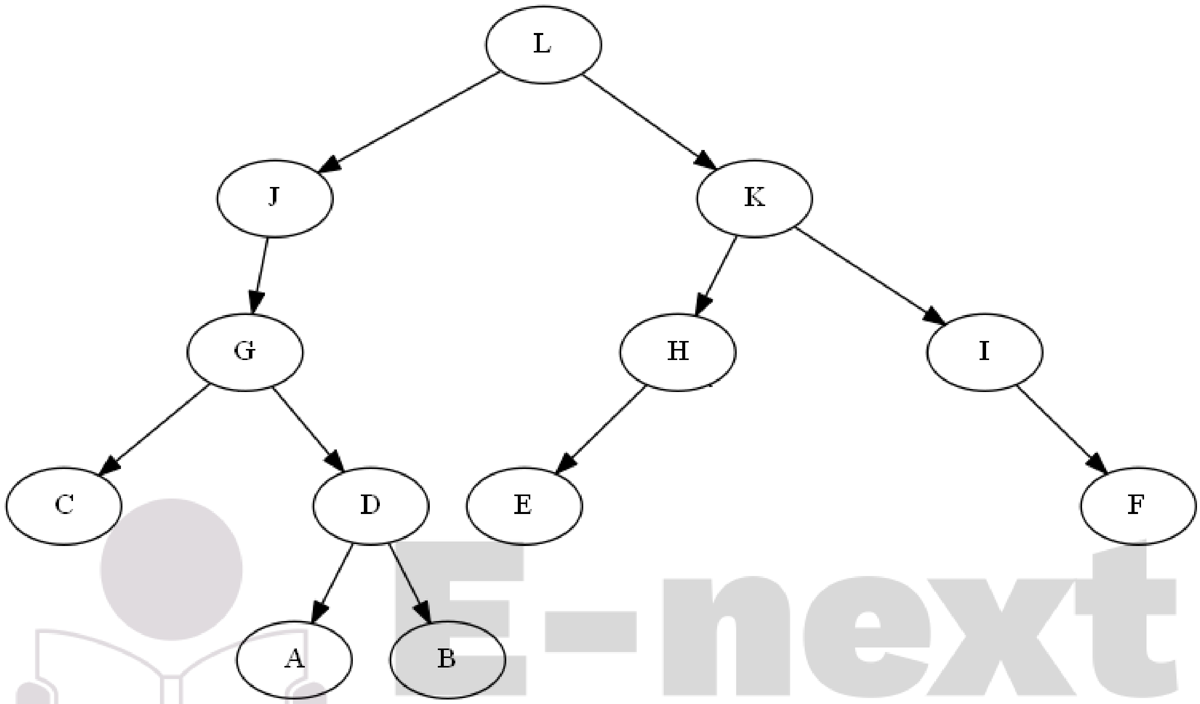
In this wesplit up thefinal remainingnodeHG in theroots ofD.**TheFinal Expressiontree**is given as:



Prefix(Preorder) Expression: **A CF EBDH G**

May2010– Q7 B– 10 Marks
Drawthetree given Preorder & the Inorder traversal below:
Preorder:	L J GCDABKHE IF
Inorder:	C GADBJLEHK IF
Also GivethePostordertraversal of thetree.
Write Algorithmfor traversing a tree.

ExpressionTree:



Postfix(Postorder) Expression:	**C A BD GJ EHF IKL**

Algorithm Prefix(Preorder)Tree Traversal:
Print the prefix expression for an expression tree.
Pre	tree is a pointer to an expression tree.
Post	thepostfix expression has beenprinted

3. if (treenotempty)
    i. print (tree token)
    ii. prefix (treeleft subtree)
    iii. prefix (treeright subtree)
4. end if

end postfix

Algorithm Postfix (Postorder)TreeTraversal:
Print the postfix expression for anexpression tree.
Pre	tree is a pointer to an expression tree.
Post	thepostfix expression has beenprinted

- if (treenotempty)
    i. postfix (tree left subtree)
    ii. postfix (tree right subtree)
    iii. print (tree token)
- end if

end postfix

Algorithm Infix(Inorder)TreeTraversal: Print the infix expression for an expression tree. Pre tree is a pointer to an expression tree. Post thepostfix expression has beenprinted

- if (treenotempty)
  - i. prefix (treeleft subtree)
  - ii. print (tree token)
  - iii. prefix (treeright subtree)
- end if

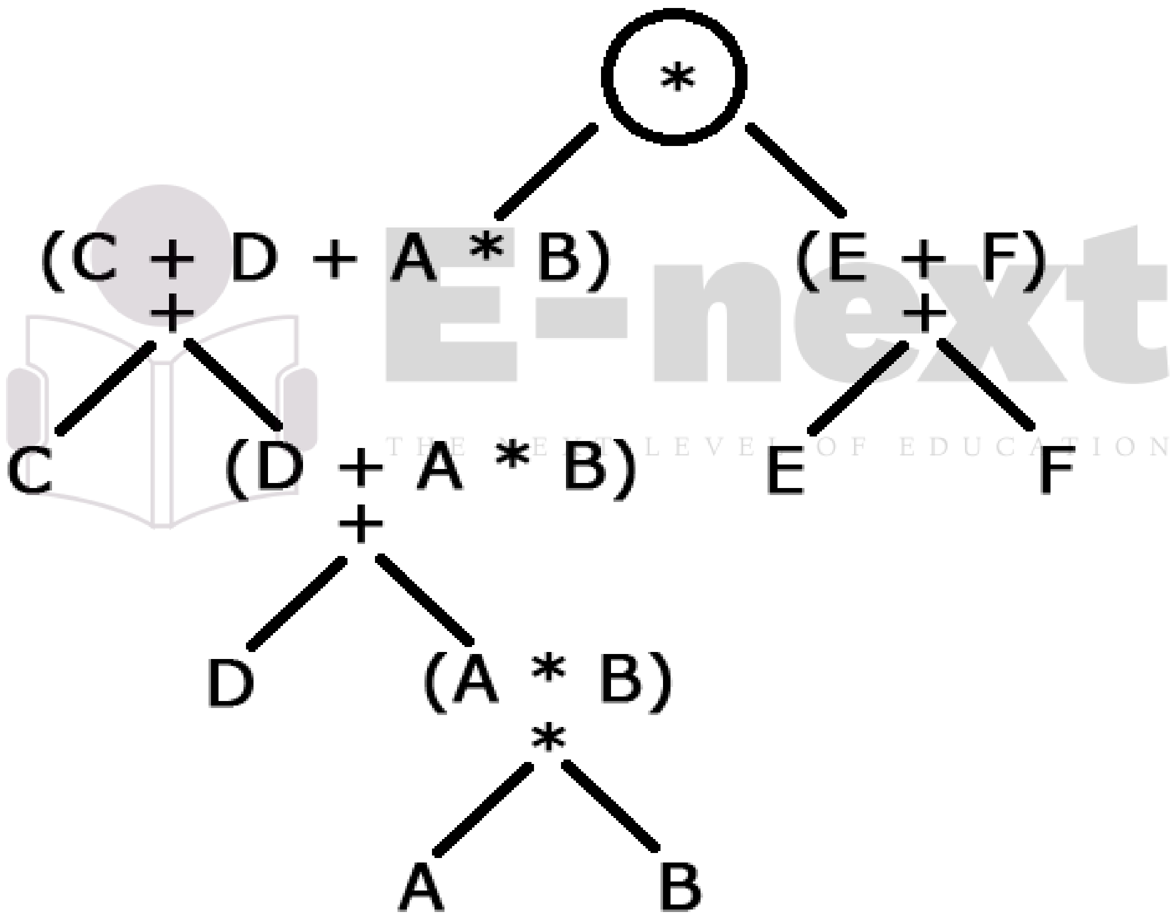
end infix

Dec 2009– Q7A– 10 Marks

Define an expression tree. Following Infix expression is given.Drawthe expression tree. Find thePrefix & Postfix expression

(C+D+ A* B) *(E+F)

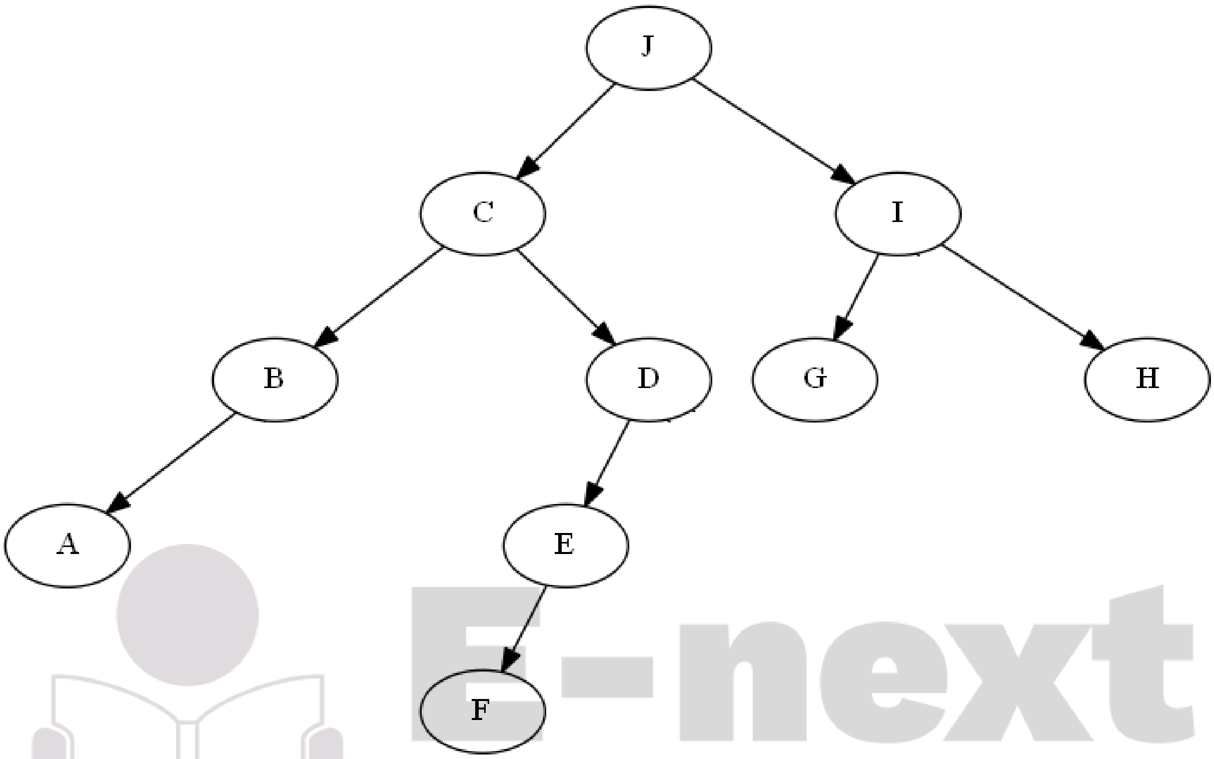
ExpressionTree:



Prefix(Preorder) Expression: *** + C+D* A B+ EF**

Postfix(Postorder) Expression: **C DA B*+ + EF +***

May 2009– Q5A– 10 Marks& May 2006 – Q1 A – 10 Marks
A binary tree has 10 nodes. The inorder and preorder traversal are shown below:
Inorder: A B C D E F J G I H
Preorder: J C B A D E F I G H
Show the binary tree, along with its Postorder traversal



Postfix(Postorder) Expression: A B F E D C G H I J