Chapter

13 NON LINEAR DATA STRUCTURE

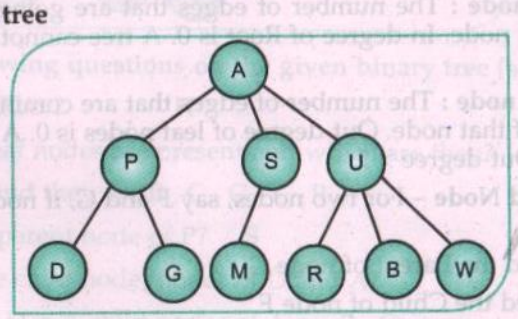
- Tree ; Binary Tree ; Complete Binary Tree
- Terminology used in Trees
- Algorithm to create
  - Complete Binary Tree
  - Binary Search Tree
- Traversal of elements in a Binary Tree
  - In-order traversal
  - Pre-order traversal
  - Post-order traversal
- Program of Binary Search Tree
  - Algorithm of the overall program
  - Diagrammatic illustration
  - Complete Program Code
  - Sample Data
  - Drawing the Binary Tree from given traversals
- Example Program

In this kind of data structure, the data items are arranged in hierarchical form. Here, one data item can be linked to more than one data items. Example of non linear data structure is Tree.

Q. 1. What is a Tree? What is a Binary Tree?

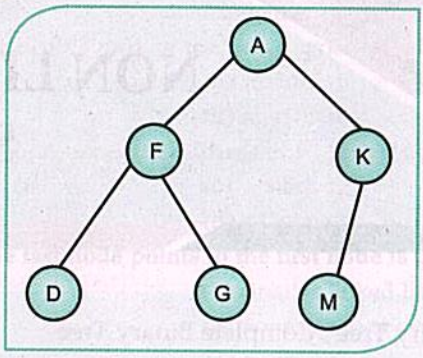
Ans. A Tree is an arrangement of elements in which each element contains the path to connect to more than one data items, without forming a loop. [A loop is formed when there is a circular connection between any two elements.]

Diagram of a **tree**

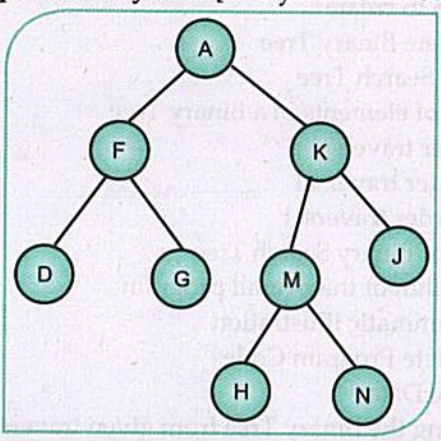


A **Binary Tree** is a special case of a Tree, in which each element contains the path to connect to (at most) two elements.

Diagram of a **binary tree**



Shown below is a **complete binary tree.** [Every node has out-degree of either 2 or 0.]



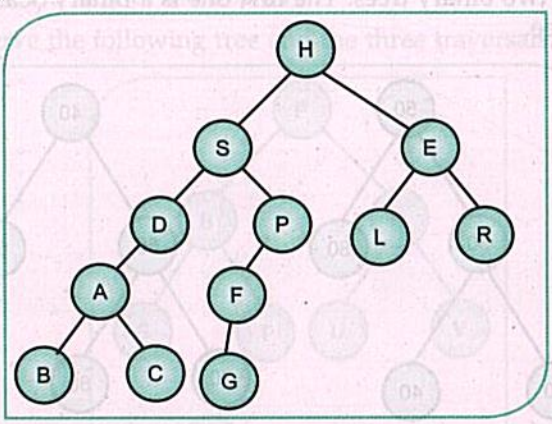
Terminology used in Trees

- **Node :** It is a structured data having two parts, "data" and "link to another node".
- **Root Node :** It is the first (or topmost) node of a tree. It does not have any parent node.
- **External Node/ Leaf Node/Terminal Node :** It is a node that does not have any child node.
- **Internal Node :** It is a node that has child node or nodes.
- **In-degree of a node :** The number of edges that are going in a node is called the in-degree of that node. In-degree of Root is 0. A tree cannot have any node with In-degree > 1.
- **Out-degree of a node :** The number of edges that are coming out of a node is called the out-degree of that node. Out-degree of leaf nodes is 0. A binary tree cannot have any node with Out-degree > 2.
- **Parent and Child Node** – For two nodes, say F and G, if node G comes from node F then,
  - ◆ node F is called the Parent of node G
  - ◆ node G is called the Child of node F

- **Edge :** The connecting line between two nodes is called an Edge. An edge is denoted by using the two respective nodes. E.g., AF, FG, KM etc.
- **Path :** It is the sequence of continuous edges between two nodes. E.g., the path of node A to node G is AFG.
- **Sibling Node :** Nodes that appear from the same parent are called sibling nodes.
- **Sub-tree :** It is a part of a tree in which a node behaves as the root having its own set of descendents (or child nodes). Every node in a tree is capable of becoming a root.
- **Depth of a Node :** It is the number of edges from Root to that Node. Depth of Root is 0.
- **Level of a Node :** It is same as the Depth of a Node. The Level of Root is 0.
- **Height of a Node :** It is the number of edges from the Node to its deepest leaf of its Sub-tree. Height of each leaf is 0. Height of Root is the number of edges from Root to the deepest leaf.
- **Height of Root == Height of Tree == Depth of Tree ==** number of edges in the longest path of the tree from Root to a leaf.
- **Size of a tree :** It is the total number of nodes present in the tree.
- **Binary Tree :** A tree in which each node can have at most two child nodes. Here each node contains one data part and two link parts.
- **Complete Binary Tree :** It is a binary tree in which every node has an out-degree of either 2 or 0.
- **Binary Search Tree :** A binary tree in which the value of the left child is smaller than the value of the parent and the value of the right child is larger than the value of the parent node.

Example :

Given a binary tree :



Answer the following questions on the given binary tree [solved] :

(i) Which is the Root? H

(ii) How many leaf nodes are present and which are they?

5 leaf nodes and they are B , C , G , L , R

(iii) Which is the parent node of P? S

(iv) Which are the child node/nodes of D and A?

child node of D is A child nodes of A are B , C

(v) What is the In-degree and Out-degree of R?
In-degree of R = 1 and Out-degree of R = 0

(vi) What is the height of the tree? 4

(vii) Write the path from node H to G. Path : H S P F G

(viii) Which is the sibling node of L? R

(ix) What is the In-Degree and Out-degree of H?
In-degree of H = 0 and Out-degree of H = 2

(x) What is the height of node A and node S?
Height of node A is 1 and height of node S is 3.

Algorithm to Create a Complete Binary Tree

Step 1 : The first node in a binary tree is the root of the tree.

Step 2 : The next two nodes will be the child nodes of the root, at depth 1.

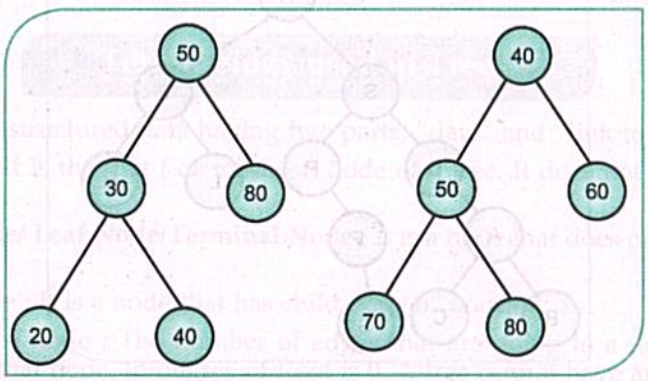
Step 3 : The next 4 nodes will be at depth 2, where each node at depth 1 will get 2 nodes.

Step 4 : In the same way, as the depth increases by 1, the number of nodes in that becomes double the number of nodes in the previous level.

Binary Search Tree

It is a binary tree in which the value of the left child is smaller than the value of parent and the value of the right child is larger than the value of the parent node. A bi search tree may not be a complete binary tree.

Shown below are two binary trees. The first one is a Binary Search Tree and the se one is just a Binary Tree.



Binary Search Tree **Binary Tree**

Algorithm to create a Binary Search Tree :

Step 1 : The first node in any binary tree is the root of the tree.

Step 2 : Each next node is first compared with the root.
If it is smaller, then it is sent to the left sub-tree.
If larger, then it is sent to the right sub-tree.

Step 3 : The process is repeated in the same way for each sub-tree.

Note :

- *Binary Search Tree contains unique values.*

Traversal of Nodes (Elements) in a Binary Tree

Since a tree is a non-linear structure, we cannot get the address of the parent node from its child node. This is why, in case of traversal of trees, the memory stack is used. Traversal of a tree is done with the help of recursive function.

There are three main ways of traversal :

(i) In-order traversal :

- Read the Root of the Left sub-tree (L)
- Read the Root Node of the tree (N)
- Read the Root of the Right sub-tree (R)

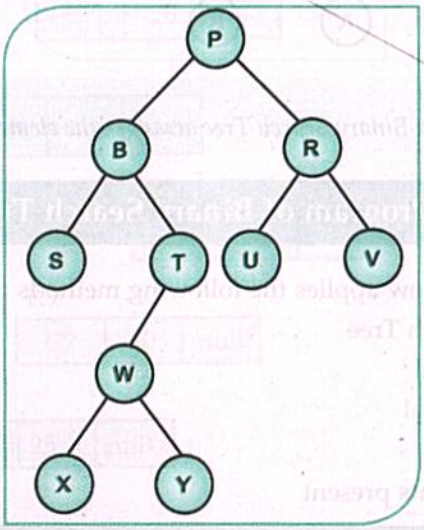
(ii) Pre-order traversal :

- Read the Root Node of the tree (N)
- Read the Root of the Left sub-tree (L)
- Read the Root of the Right sub-tree (R)

(iii) Post-order traversal :

- Read the Root of the Left sub-tree (L)
- Read the Root of the Right sub-tree (R)
- Read the Root Node of the tree (N)

Example : Observe the following tree and the three traversals.



Inorder traversal (L Node R) : S B X W Y T P U R V

Preorder traversal (Node L R) : P B S T W X Y R U V

Postorder traversal (L R Node) : S X Y W T B U V R P

Exercise : Draw the **Binary Search Tree** for the input list :

(i) 70 40 50 90 80 20

Ans : The first value is the node.

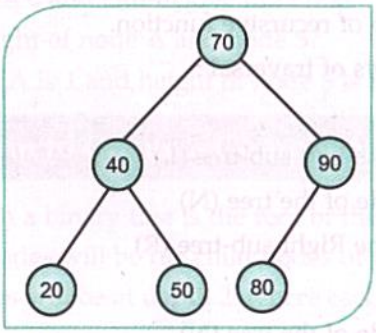
Take each next value and place it in the

left sub-tree if it is smaller than the current node

right sub-tree if it is larger than the current node

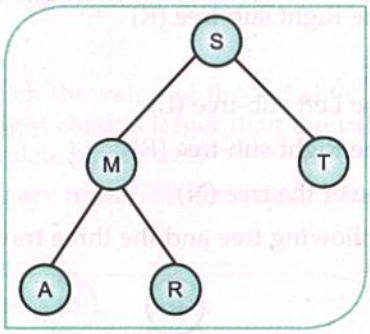
The process repeats for the sub-trees

[The given list contains numeric values.]



(ii) S M A R T

[The given list contains alphabetical values.]



Note :

- *The in-order traversal of a Binary Search Tree arranges the elements in order.*

Program of Binary Search Tree

The program given below applies the following methods :

(i) Make Binary Search Tree

(ii) Pre-order Traversal

(iii) Post-order Traversal

(iv) In-order Traversal

(v) Count total elements present

Algorithm of the Program

A Self Referential Structure called **clBinTree** is created to store the data and address.

It can store one data and two addresses, named leftChild and rightChild.

The first element in the tree is set to the Root. Then on each time a new data has to be entered in the tree, it is compared with the element in the current Root.

- If the new element is smaller, it is sent to the left-sub-tree of the current tree.
- If the new element is larger, it is sent to the right-sub-tree of the current tree.

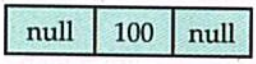
A sub-tree behaves like a tree in itself when it has to fill data in it.

Data enters a tree only when the subtree under it is vacant. The following diagram demonstrates the entry of the following set of data in a Binary Search Tree.

Diagrammatic Illustration

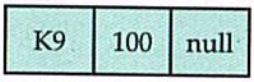
Data : 100 50 25 15 30 150

(i) D5



(ii)

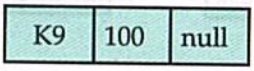
D5



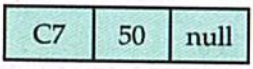
K9

(iii)

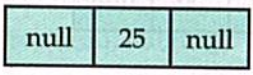
D5



K9

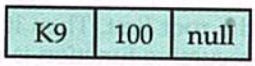


C7

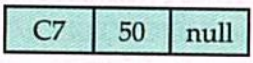


(iv)

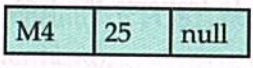
D5



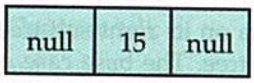
K9



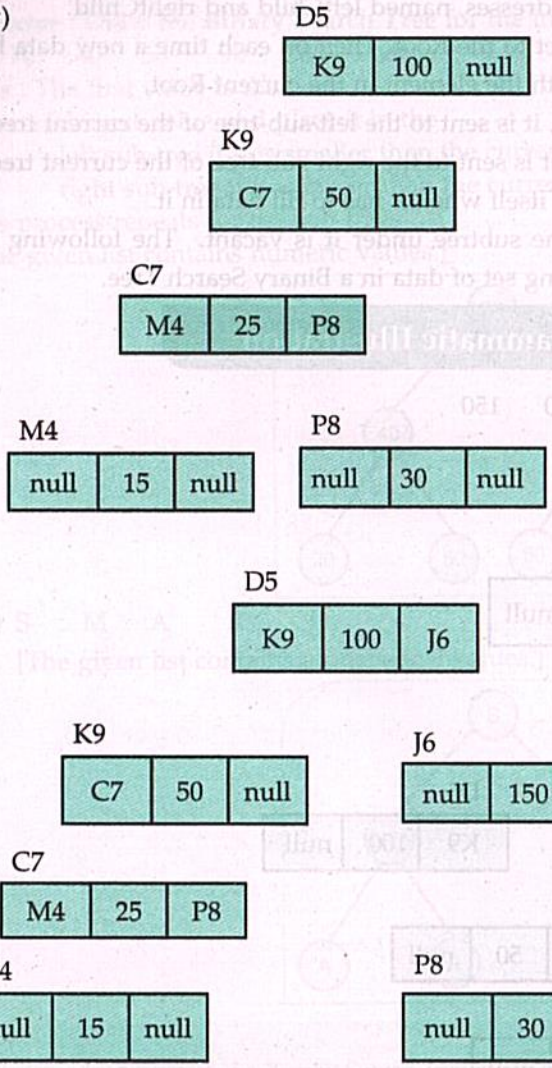
C7



M4



(v)



(vi)

Pre-order Traversal

It reads the current data of the root first. Then it reads the data on its left sub-tree and then data on its right sub-tree. A recursive function is used to traverse the tree. The base case of the recursion is at **null.**

In-order Traversal

It reads the leftmost data first. Then it reads the data on its current root and then data on its right sub-tree. A recursive function is used to traverse the tree. The base case of the recursion is at **null.**

Post-order Traversal

It reads the leftmost data first. Then it reads the data on its right sub-tree and then data on its root. A recursive function is used to traverse the tree. The base case of the recursion is at **null.**

Count Items

It traverses the tree in any order and keeps a counter that increments with each non-null element in the tree.

Program 1 : Binary Search Tree

```
import java.io.*;
import java.util.*;
public class clBinTree
{
    public int data;
    clBinTree leftChild;  //memory allocation cannot be done here
    clBinTree rightChild;
    clBinTree(int ini)
    {
        data = ini;
        leftChild = null;
        rightChild = null;
    }
    public void main()
    {
        clBinTree head = new clBinTree(100);
        Scanner sc = new Scanner (System.in);
        int dataval = 0;
        int m = 1 ;
        do
        {
            System.out.print("\n Enter number for the tree : ");
            dataval = sc.nextInt();
            fnMakeTree(head, dataval);
            System.out.print("\n Any more data ( 1 for Yes)  ");
            m = sc.nextInt() ;
        } while ( m== 1) ;
        // TRAVERSALS – PreOrder, PostOrder, InOrder
        System.out.print("\n PREORDER NodeLR ");
        fnPreOrder(head);
        System.out.print("\n POSTORDER LRNode");
        fnPostOrder(head);
        System.out.print("\n INORDER LNodeR");
        fnInOrder(head);
        //count
        System.out.print("\n Total elements  = " + fnCountItems(head));
```

```
}
void fnMakeTree( clBinTree mov, int d)
{
    if(d <mov.data)
    {
        if(mov.leftChild == null)
        {
            System.out.print("\n left .. ");
            clBinTree cur = new clBinTree(d);
            mov.leftChild = cur;
        }
        else
        {
            System.out.print("\n Changing level (going down) .. ");
            fnMakeTree(mov.leftChild, d);
        }
    }
    else if (d >mov.data)
    {
        if(mov.rightChild == null)
        {
            System.out.print("\n right .. ");
            clBinTree cur = new clBinTree(d);
            mov.rightChild = cur;
        }
        else
        {
            System.out.print("\n Changing level (going down) .. ");
            fnMakeTree(mov.rightChild, d);
        }
    }
    else
    {
        System.out.print("\n Equal data not allowed. ");
    }
}
void fnPreOrder(clBinTree mov)
{
    if(moy == null)
    {
        return;
    }
```

```
        System.out.print("  ** " + mov.data);
        fnPreOrder(mov.leftChild);
        fnPreOrder(mov.rightChild);
    }
    void fnPostOrder(clBinTree mov)
    {
        if(mov == null)
        {
            return;
        }
        fnPostOrder(mov.leftChild);
        fnPostOrder(mov.rightChild);
        System.out.print("   $$ " + mov.data);
    }
    void fnInOrder(clBinTree mov)
    {
        if(mov == null)
        {
            return;
        }
        fnInOrder(mov.leftChild);
        System.out.print("   ##  " + mov.data);
        fnInOrder(mov.rightChild);
    }
    int fnCountItems(clBinTree mov)
    {
        if(mov == null)
        {
            return 0;
        }
        int count = 1;
        count = count + fnCountItems(mov.leftChild);
        count = count + fnCountItems(mov.rightChild);
        return count;
    }
}
```

SAMPLE DATA :

```
                100
        50              150
    25
15      30
```

OUTPUT

```
Enter number for the tree :50
left ..
Any more data ?y
Enter number for the tree :25
Changing level (going down) ..
left ..
Any more data ?y
Enter number for the tree :30
Changing level (going down) ..
Changing level (going down) ..
right ..
Any more data ?y
Enter number for the tree :15
Changing level (going down) ..
Changing level (going down) ..
left ..
Any more data ?y
Enter number for the tree :150
right ..
Any more data ?n
PREORDER NodeLR  ** 100  ** 50  ** 25  ** 15  ** 30  ** 150
POSTORDER LRNode  $$ 15  $$ 30  $$ 25  $$ 50  $$ 150  $$ 100
INORDER LNodeR  ## 15  ## 25  ## 30  ## 50  ## 100  ## 150
Total elements = 6
```

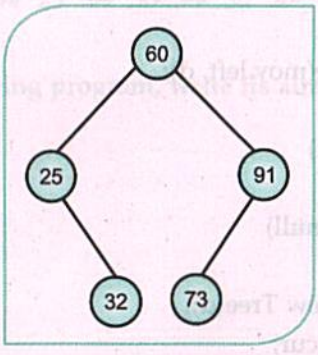
Program 2 : Given the following program, write tis aim :

```
class Tree
{
    int data;
    Tree left;
    Tree right;
    Tree()
    {
        data = 0;
        left = right = null;
    }
```

```
Tree(int dd)
{
    data = dd;
    left = null ; right = null ;
}
void main()
{
    Tree twig = new Tree(60);
    for( int k = 1; k<=5; k++)
    {
        int val = (int)(Math.random() * 100);
        fnMakeTree(twig, val);      //filling random numbers in the tree
    }
    System.out.print("\n\n Line 1 Calling fnInOrder .. \n");      fnInOrder(twig);
    System.out.print("\n\n Calling method Twig .. \n");          fnTurn(twig);
    System.out.print("\n\n Line 2 Calling fnInOrder .. \n");      fnInOrder(twig);
}
void fnMakeTree( Tree mov, int d) // binary search tree
{
    System.out.println(" data is : " + d);          // assume data is not repeated
    if(d < mov.data)
    {
        if(mov.left == null)
        {
            Tree cur = new Tree(d);
             mov.left = cur;
        }
        else
             fnMakeTree(mov.left, d);
    }
    else if (d > mov.data)
    {
        if(mov.right == null)
        {
            Tree cur = new Tree(d);
            mov.right = cur;
        }
        else
            fnMakeTree(mov.right, d);
    }
}
```

```
    void fnInOrder(Tree leaf)
    {
        if( leaf == null)
        {
            return;
        }
        fnInOrder(leaf.left);
        System.out.print("  ##  " + leaf.data);
        fnInOrder(leaf.right);
    }
    void fnTurn(Tree branch)
    {
        if( branch == null)
            return ;
        System.out.print(" ** " + branch.data);
        fnTurn(branch.left);
        fnTurn(branch.right);
        System.out.print(" ## " + branch.data);
        Tree tmp  = branch.left ;
        branch.left = branch.right;
        branch.right = tmp;
    }
}
```

Assume the Tree to be



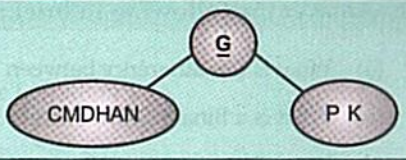
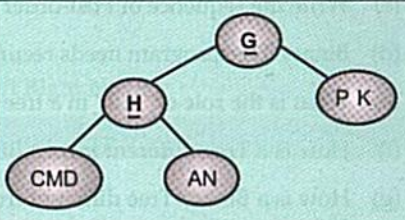
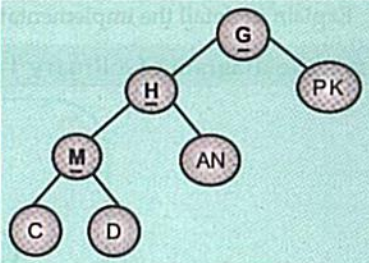
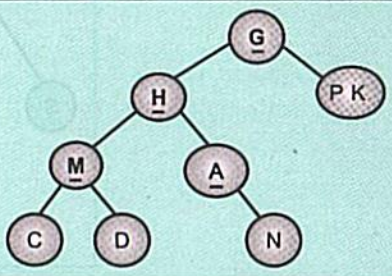
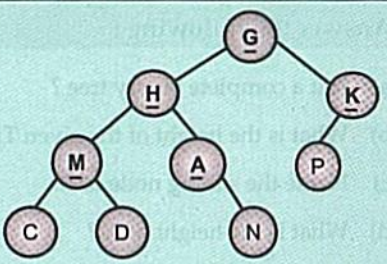
Q. 2. Given the following traversals, draw the corresponding binary tree

In-order : C M D H A N G P K

Pre-order : G H M C D N A K P

[Hint : Use the traversal sequence alternately,

In-order – Left Node Right and Pre-order – Node Left Right]

Step Num	Traversals InOrder – L Node R Traversal PreOrder – Node L R	Tree
1	CMDHAN**G** PK **G**HMCDNAKP	
2	CMD**H**ANGPK G**H**MCDNAKP	
3	C**M**DHANGPK GH**M**CDNAKP	
4	CMDH**A**NGPK GHMCDN**A**KP	
5	CMDHANGP**K** GHMCDNA**K**P	

Exercise

1. State True or False :

(a) The total number of nodes present in a tree is called its height.

(b) The Out-degree of every leaf node in a tree is 0.

(c) The In-degree of Root node is 1.

(d) In a Complete Binary Tree, each node can have 0, 1 or 2 child nodes.

(e) Height of Root is same as the Depth of the Tree.

2. Answer the following in brief :

(a) What is the difference between a Binary Tree and a Complete Binary Tree ?

(b) What is a Binary Search Tree?

(c) Write the sequence of Post-order traversal.

(d) Binary Tree program needs recursive functions. Justify.

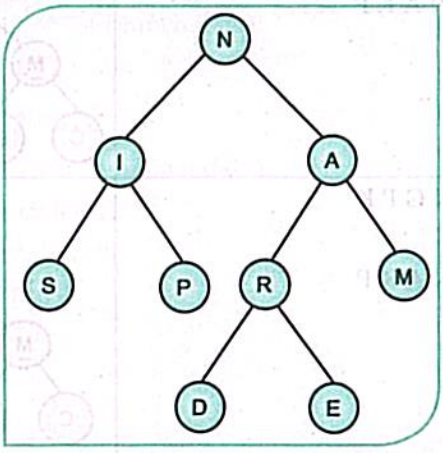
(e) What is the role of 'null' in a tree program ?

(f) How is a Tree different from a Binary Tree? Explain with a neat example.

(g) How is a Binary Tree different from a Binary Search Tree? Explain with a neat example.

(h) Explain in detail the implementation of Binary Search Tree and perform its operations.

3. Given the diagram of a Binary Tree : [for more exercise refer to Chapter 6(B)]



Answer the following :

(a) Is it a complete binary tree ?

(b) What is the height of the given Tree?

(c) Name the sibling node of R.

(d) What is the height of A ?

(e) What is the path of A to E ?

(f) What is the depth of R ?

(g) What is the size of the tree ?

(h) Draw the right sub-tree of N.

(i) What is the in-degree and out-degree of P ?

(j) Write the Pre-order , Post-order & In-order traversal of the given tree.

4. Draw the Binary Tree, using the traversals :

(a) In-order : P S T Q L W M R N

Pre-order : W S P Q T L R M N

Hint : Use the traversal sequence alternately,

In-order – Left Node Right and Pre-order – Node Left Right]

(a) In-order : H Y D R O G E N

Post-order : H D Y G O N E R

Hint : Use the traversal sequence alternately,

In-order – Left Node Right and Post-order – Left Right Node]

5. Draw the Binary Search Tree for the input list :

(a) 60, 25,75,15,50,66,33,44.

(b) G A R D E N S

❑❑