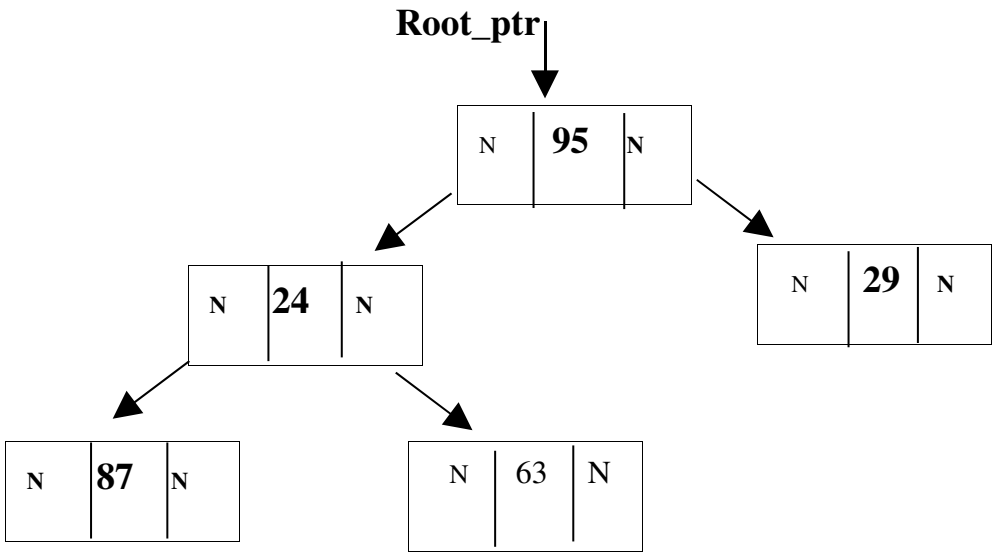
TREES

Binary Trees:

- Has a special node called Root
  Each node has either:
  -No child ➔ Leaf node
  -One child (either a left or a right child)
  -Two children
  Each node has one and only one parent except the root node.

Example: **N stands for NULL.**

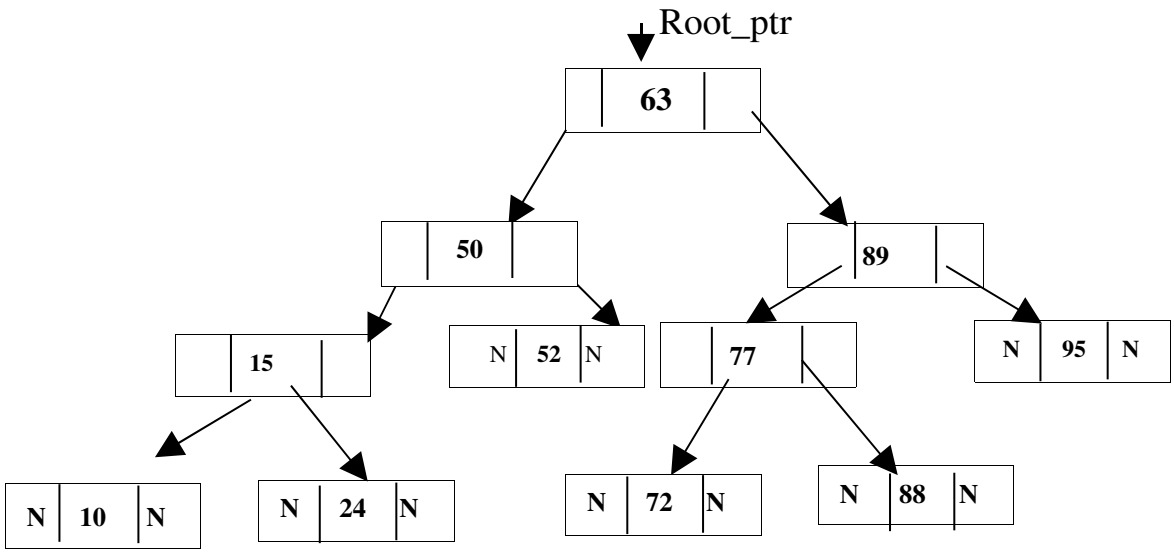


Binary Search Trees:

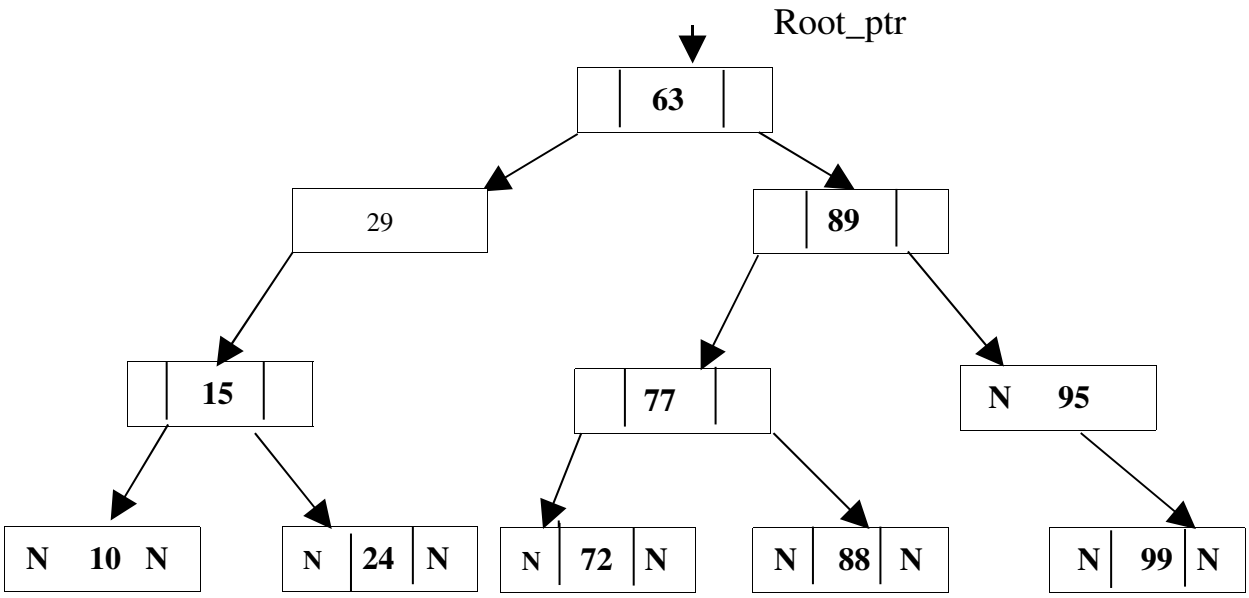
A Binary Search Tree (BST) is a binary tree with the following properties:

Every node contains a special value called key that defines the order of the nodes.

- Key values are unique: No key appear more than once in the tree.
  For any given node, the key must be greater than all the keys in subtree rooted at its left child and smaller than all the keys rooted at its right child.



Linked List representation of BST



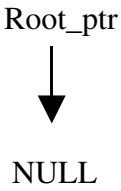
Array Representation of a Binary Search Tree

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
63	**29**	**89**	**15**		**77**	**95**	**10**	**24**			**72**	**88**		**99**	

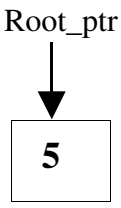
Left child of Array[j] is located at :Array[2*j]
Right child of Array[j] is located at Array[2*j+1]
Parent of Array[j] is located at: Array[j\2]

Trees: Insertion In A Binary Search Tree

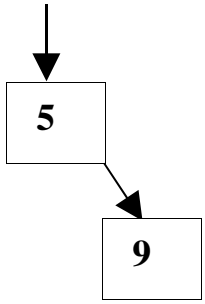
Example:



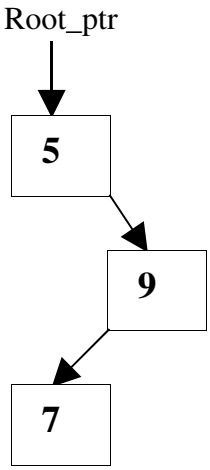
a) Insert 5:



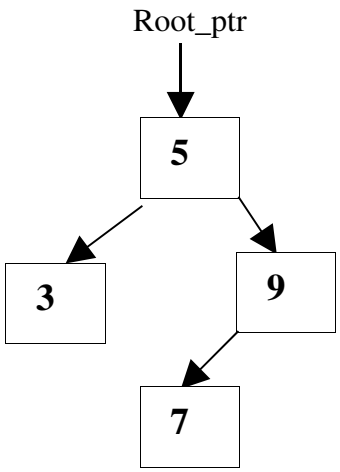
b) Insert 9: Root_ptr



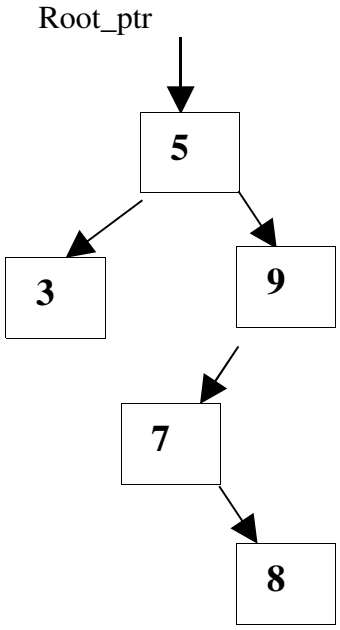
c) Insert 7:



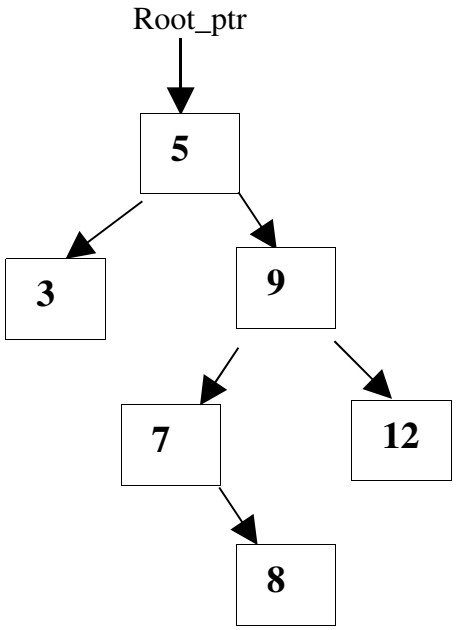
d) Insert 3

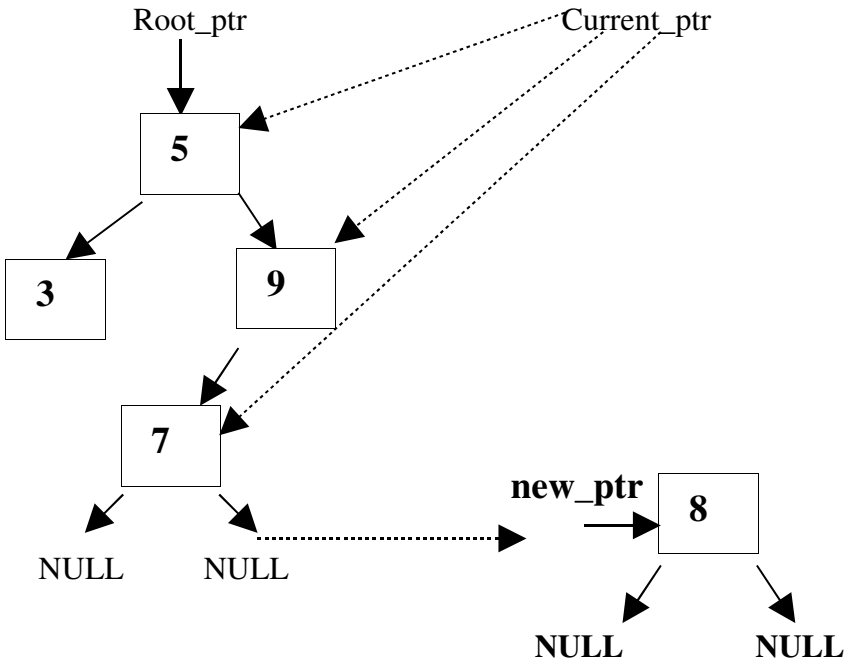


e) Insert 8



f) Insert 12:





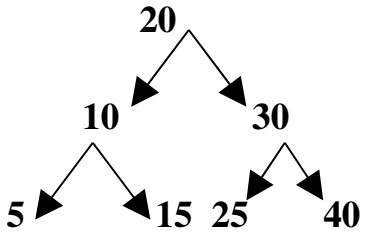
Iterative process:

Current_ptr = Root_ptr; new_ptr.left_ptr = NULL; new_ptr.right_ptr = NULL;

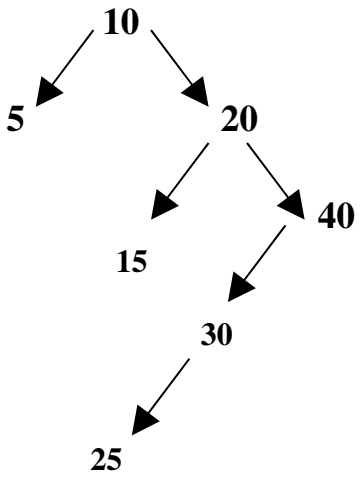
```
While ( Current_ptr != NULL){ /* Start of while */
        if (new_ptr.data < Current_ptr.data ){
                /*  Move to left */
                if(Current_ptr.left_ptr  != NULL)
                        Current_ptr = Current_ptr.left_ptr;
                else{
                        /* Reaching a leaf node */
                        Current_ptr.left_ptr = new_ptr;
                        break;
                    }
        } /* End of if */
        else if (new_ptr.data > Current_ptr.data){
                /* Move to right */
                if(Current_ptr.right_ptr != NULL)
                        Current_ptr = Current_ptr.right_ptr;
                else{
                        /* Reaching a leaf node */
                        Current_ptr.right_ptr = new_ptr;
                         break;
                    }
        }
}/* End of while */
```

TREES:SHAPE OF BST

Input: 20 10 30 5 15 25 40



Input: 10 5 20 15 40 30 25



Input: 5 10 15 20 25 30 40