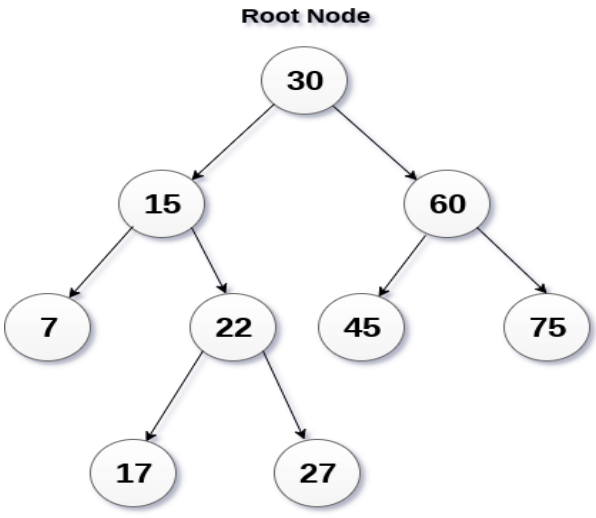
# PROGRAM 4: IMPLEMENTATION OF A BINARY SEARCH TREE

**Aim:** To write a program in C++ for implementing Binary Search Tree.

**Description:**

1. Binary Search tree can be defined as a class of binary trees, in which the nodes are arranged in a specific order. This is also called ordered binary tree.
2. In a binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root.
3. Similarly, value of all the nodes in the right sub-tree is greater than or equal to the value of the root.
4. This rule will be recursively applied to all the left and right sub-trees of the root.



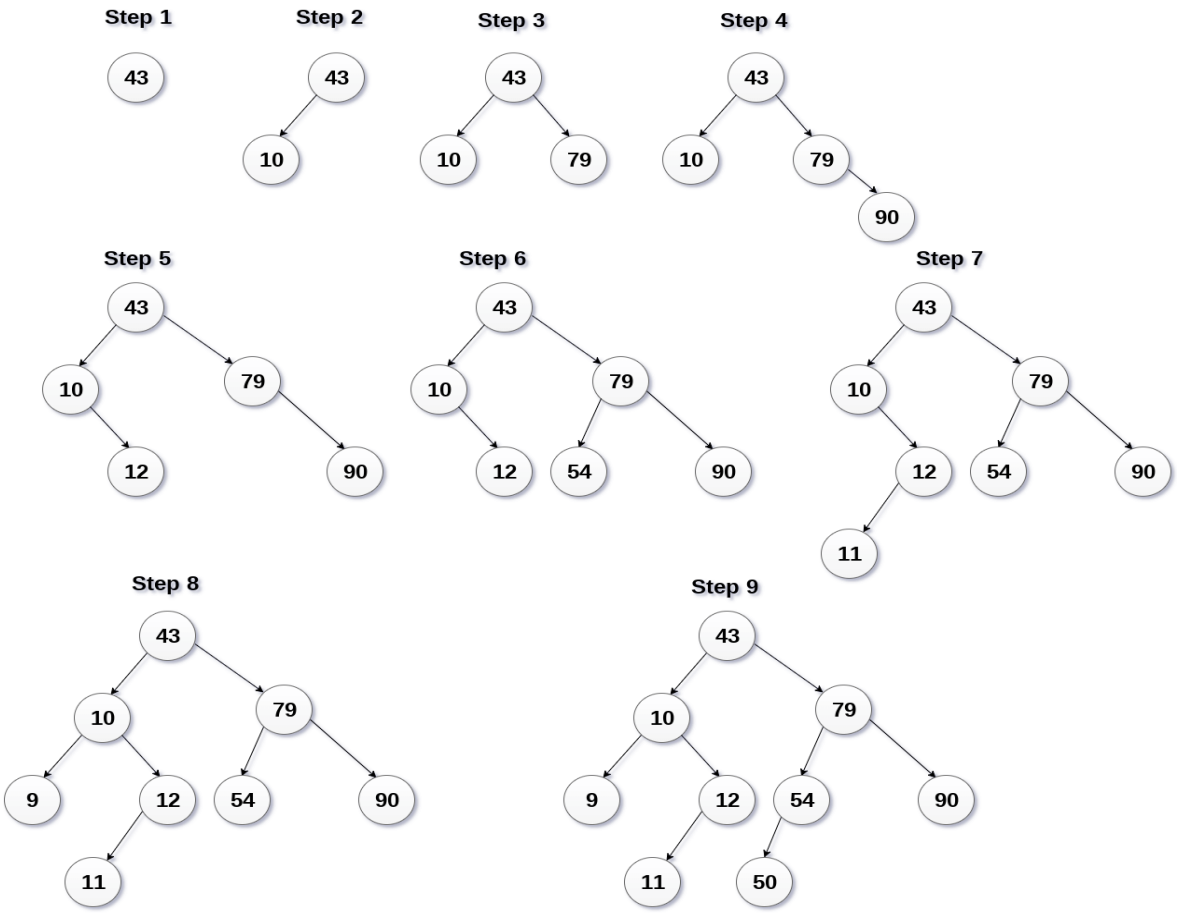

Binary Search Tree

***Creation of Binary Search Tree:***

**43, 10, 79, 90, 12, 54, 11, 9, 50**

1. Insert 43 into the tree as the root of the tree.
2. Read the next element, if it is lesser than the root node element, insert it as the root of the left sub-tree.
3. Otherwise, insert it as the root of the right of the right sub-tree.

The process of creating BST by using the given elements, is shown in the image below.




**Binary search Tree Creation**

**Algorithm:**

Step 1: Start

Step 2:If node == NULL

return createNode(data)

if (data < node->data)

node->left = insert(node->left, data);

else if (data > node->data)

node->right = insert(node->right, data);

return node;

Step 3: Stop

**Program:**

```
// Binary Search Tree operations in C++
#include <iostream>
using namespace std;

struct node {
  int key;
  struct node *left, *right;
};

// Create a node
struct node *newNode(int item) {
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->key = item;
  temp->left = temp->right = NULL;
```

```
  return temp;
}

// Inorder Traversal
void inorder(struct node *root) {
  if (root != NULL) {
    // Traverse left
    inorder(root->left);
    // Traverse root
    cout << root->key << " -> ";
    // Traverse right
    inorder(root->right);
  }
}

// Insert a node
struct node *insert(struct node *node, int key) {
  // Return a new node if the tree is empty
  if (node == NULL) return newNode(key);
  // Traverse to the right place and insert the node
  if (key < node->key)
    node->left = insert(node->left, key);
  else
    node->right = insert(node->right, key);
  return node;
}
```

```
// Driver code
int main() {
  struct node *root = NULL;
  root = insert(root, 8);
  root = insert(root, 3);
  root = insert(root, 1);
  root = insert(root, 6);
  root = insert(root, 7);
  root = insert(root, 10);
  root = insert(root, 14);
  root = insert(root, 4);
  cout << "Inorder traversal: ";
  inorder(root);
}
```

**Output:**

**Inorder traversal: 1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 10 -> 14 ->**

**Result:** Thus a program is written and implemented successfully in C++ for Binary Search Tree.