Binary Search Tree (BST) Insertion and Searching

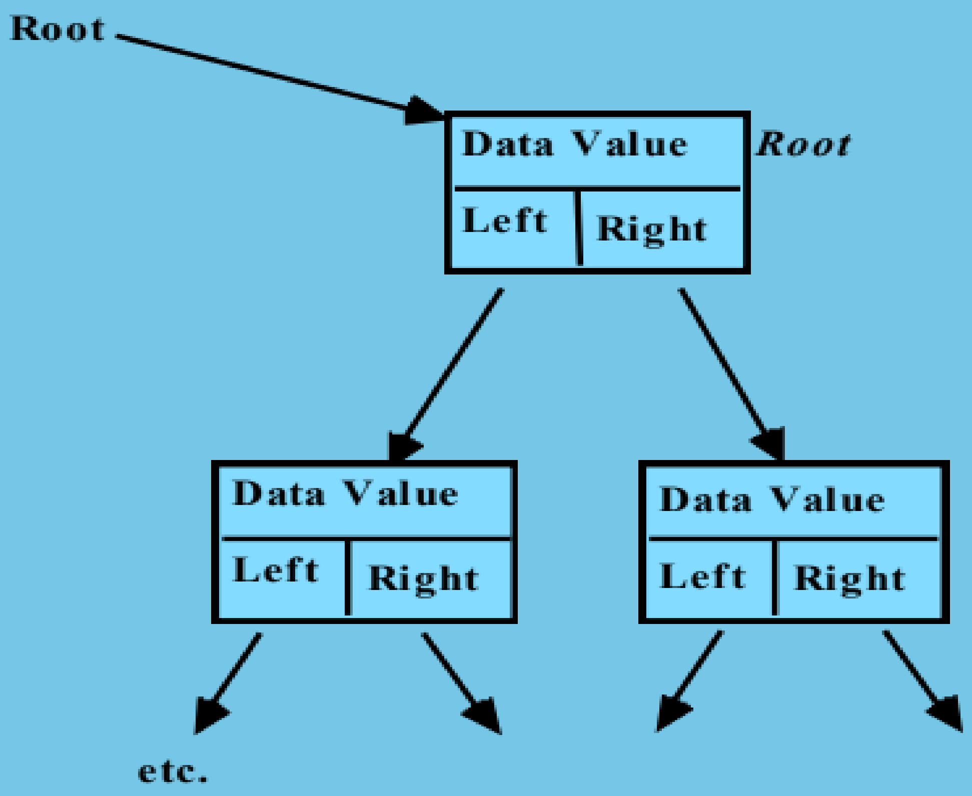
Recursive Function

- A recursive function is a function that <u>calls itself during its execution.</u>
- This enables the function to repeat itself several times, outputting the result and the end of each iteration.
- The recursion continues until some condition is met to prevent it.
- Below is an example of a recursive function.

Example: Sum of Natural Numbers Using Recursion

```
#include <stdio.h>

int sum (int n ) ;

int main()
{
    int number, result;

    printf ("Enter a positive integer: ");
    cin>>number;

    result = sum(number);

Cout<< "sum = “<< result ;
    return 0;
}

int sum (int  num)
{
    if (num !=0)
        return  num +  sum ( num-1 );     // sum() function calls itself
    else
        return num;

}
```

Implementing Binary Trees



Insertion in Binary Search Tree

- While doing insertion in BST the new ley or data always inserted at leaf.
- We start searching a key from root till we hit the leaf node.
- When a leaf node is founded, the new node is added as a child of the leaf node.
- The algorithm depends on the property of BST that if each left subtree has values below root and each right subtree has values grater than root.

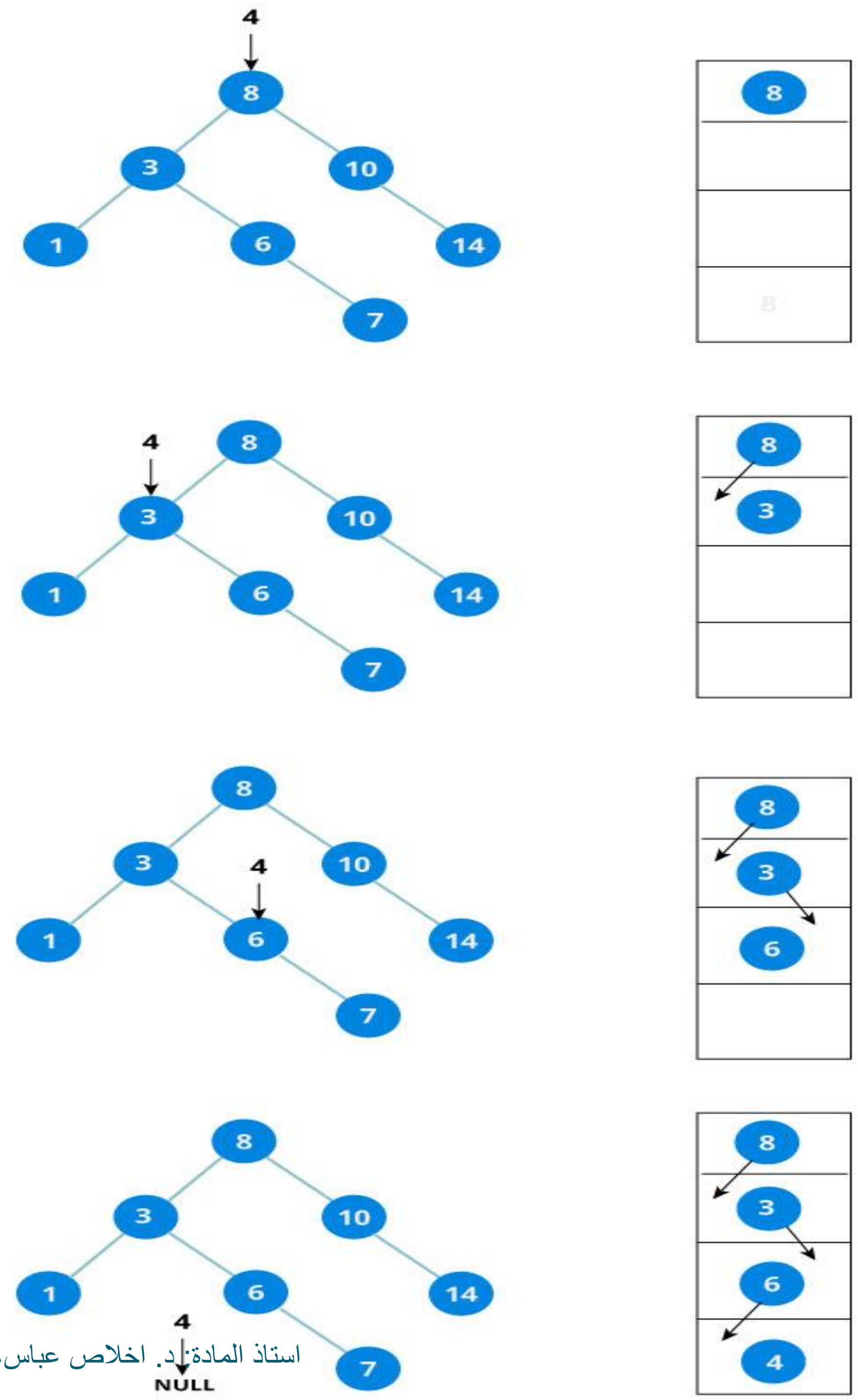
Insert Node in a Binary Search Tree

```
Node* Insert(Node *root, int data)
{
  if (root == NULL)
     return  new_node(data);                    //create new node and return

if (data < root->data)
       root->left = Insert(root->left, data); // Insert() function calls itself

else  if (data > root->data)
       root->right = Insert(root->right, data); // Insert() function calls itself

return root ;
}
```

Example:

Insert a number 4

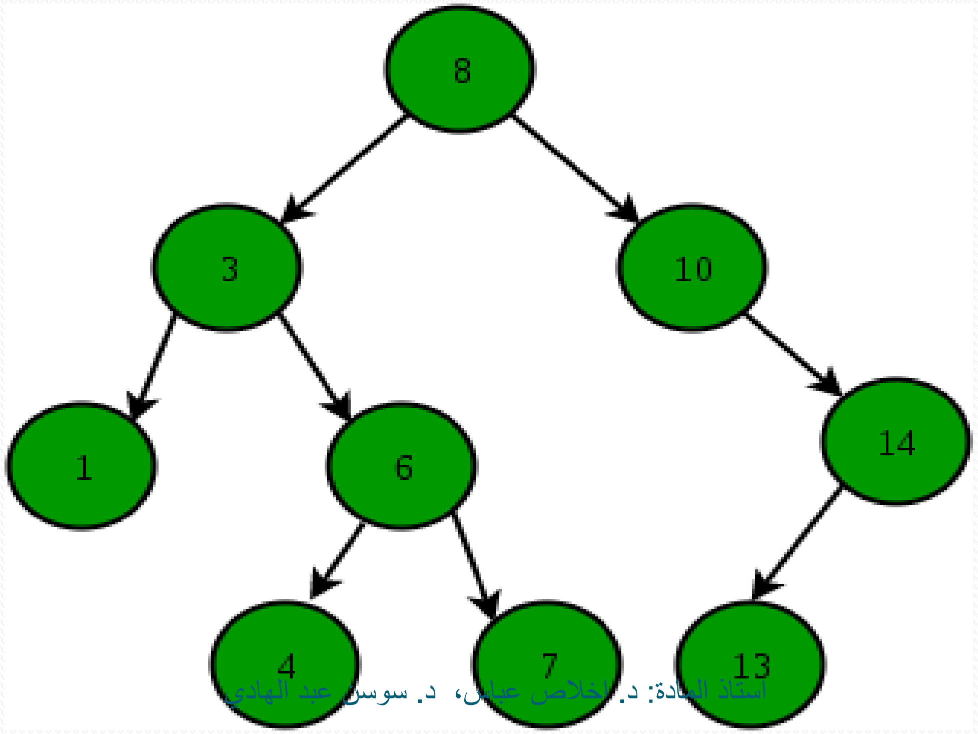
to an Existing BST.



Searching in binary search tree

illustration to search 6 in below tree:

1. Start from root.
2. If the key present at root we return root.
3. Compare the inserting element with root, if less than root, then recurse for left, else recurse for right.
4. If key to search is found anywhere, return true, else return false.



Search function for given data in a given BST

```
struct node* search(struct node* root, int data)
{
  if (root == NULL || root->data == data) // if root is null or data is present at root
     return root;


    if (root->data < data)                    // if data is greater than root's key
     return search(root->right, data);

    else
     return search(root->left, data);          // if data is smaller than root's data
}
```