In Order: 60 65 70 90 100 120 150 190

```
if(root==search){
return true;
}
if(root==null){
   return false;
 }

if(root.data<search){
b=bst(root.left,search);
}

if(root.data>search){
b=bst(root.right,search);
}
```

bst(root,100); //true

In Order: 60 65 70 90 100 120 150 190

```
if(root==search){
return true;
}
if(root==null){
   return false;
 }

if(root.data<search){
b=bst(root.left,search);
}

if(root.data>search){
b=bst(root.right,search);
}
```

bst(root,100); //true

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```java
package dsaj6;

/**
 *
 * @author Admin
 */
public class Tree {

    public int data;
    Tree left;
    Tree right;

    public Tree(int data) {
        this.data = data;
        left = null;
        right = null;
    }

    public Tree() {

    }

    public static void inOrder(Tree root) {
        if (root == null) {
            return;
        }
        // LEFT ROOT RIGHT

        inOrder(root.left);//for Left Sub Tree
        System.out.print("===>" + root.data);//ROOT
        inOrder(root.right);//Right Sub Tree

    }

    public static Tree insert(Tree root, int data) {
        if (root == null) {
            return new Tree(data);
        }
        if (data < root.data) {
            root.left = insert(root.left, data);
        } else {
            root.right = insert(root.right, data);
        }

        return root;
    }

    public boolean bst(Tree root,int search){
        boolean b=false;
        try{
        if(root.data==search){
            return true;
        }
        if(root==null){
            return false;
        }
        if(search<root.data){
```

```java
            b=bst(root.left, search);
        }if(search>root.data){
            b=bst(root.right,search);
        }
        }catch(Exception ne){
            System.out.println("Element Not Found");
            return false;
        }
        return b;
    }
    public static void main(String[] args) {
        Tree root = new Tree(100);
        insert(root, 70);
        insert(root, 60);
        insert(root, 65);
        insert(root, 90);
        insert(root, 150);
        insert(root, 120);
        insert(root, 190);

        System.out.println("\nBinary Search : "+root.bst(root, 191));
    }
}
```

```java
package dsaj6;

/**
 *
 * @author Admin
 */
public class Tree {

    public int data;
    Tree left;
    Tree right;

    public Tree(int data) {
        this.data = data;
        left = null;
        right = null;
    }

    public Tree() {

    }

    public static void inOrder(Tree root) {
        if (root == null) {
            return;
        }
        // LEFT ROOT RIGHT
```

```java
        inOrder(root.left);//for Left Sub Tree
        System.out.print("===>" + root.data);//ROOT
        inOrder(root.right);//Right Sub Tree

    }

    public static Tree insert(Tree root, int data) {
        if (root == null) {
            return new Tree(data);
        }
        if (data < root.data) {
            root.left = insert(root.left, data);
        } else {
            root.right = insert(root.right, data);
        }

        return root;
    }

    public boolean bst(Tree root,int search){
        boolean b=false;
        try{
        if(root.data==search){
            return true;
        }
        if(root==null){
            return false;
        }
        if(search<root.data){
            b=bst(root.left, search);
        }if(search>root.data){
            b=bst(root.right,search);
        }
        }catch(Exception ne){
            System.out.println("Element Not Found");
            return false;
        }
        return b;
    }
    public static void main(String[] args) {
        Tree root = new Tree(100);
        insert(root, 70);
        insert(root, 60);
        insert(root, 65);
        insert(root, 90);
        insert(root, 150);
        insert(root, 120);
        insert(root, 190);

        System.out.println("\nBinary Search : "+root.bst(root, 191));
    }
}
```

Collection(I)

drawback of array
1. fixed in size after declaring the array we
can not change it
2. It can store same type of data

Advantage of Collection
1. This is flexible and growable in nature
2. It can store same type data or different
types data except TreeSet

Collection(I)

Enumeration(I)
Iterator(I)
ListIterator(I)
Comparable(I)
Comparator(I)

List(I)   Set(I)   Queue(I)

Vector(C) 1.0 **(Indexed)**

Stack(C)

ArrayList(C) **(Indexed)**   HashSet(C)   SortedSet(I)   PriorityQueue(C)

LinkedList(C)   LinkedHashSet(C)   NavigableSet(I)   Dequeue(I)

TreeSet(C)   ArrayDeque(C)

Object get(Index);
add(Object)

First diagram — Collection Framework

Collection(I)

Enumeration(I)
Iterator(I)
ListIterator(I)
Comparable(I)
Comparator(I)

List(I)   **Set(I)**   **Queue(I)**

Vector(C) 1.0 (Indexed)

Stack(C)

ArrayList(C) (Indexed)

LinkedList(C)

HashSet(C)   SortedSet(I)   PriorityQueue(C)

LinkedHashSet(C)   NavigableSet(I)   Dequeue(I)

TreeSet(C)   ArrayDeque(C)

Object get(Index);   It can be represent data inform of individual element
add(Object)   Collection Framework

Second diagram

java.util package

put(key,value);   **Map(I)**
Object getKey(key)

HashMap(C)   SortedMap(I)   Dictionary(AC)

LinkedHashMap(C)   NavigableMap(I)   HashTable(C)

WeakHashMap(C)   TreeMap(C)   Properties(C)

IdentityHashMap(C)

Map(Key value Pair): If we want to represent data inform of key value pair then we should go for Map

Note: Map is not a child Interface of Collection