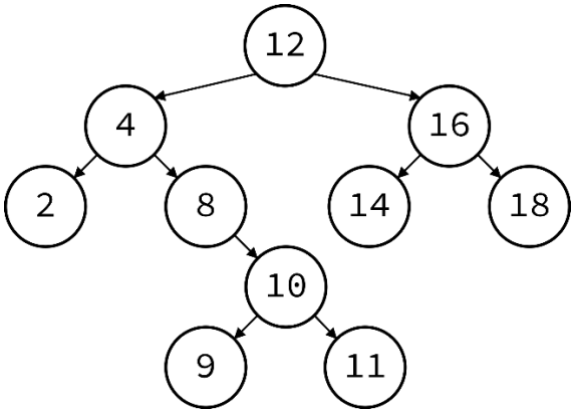
Concept Application & Algorithmic Part

Question 1: (20 points)

Suppose we have the following Binary Search Tree.



Algorithm Write up. Write an algorithm to determine the leaf nodes of the given tree reclusively.

Output: The leaf nodes of the given tree are: 2 9 11 14 18

```
private void printLeafNodes(BSTnode node) {
    // base case.
   if (node != null) {
      //check if a node is a leaf node.
      if (node.getLeft() == null && node.getRight() == null) {
      // print leaf nodes.
     System.out.printf("%d ", node.getData());
     }else {
         //check another node
         printLeafNodes(node.getLeft());
         printLeafNodes(node.getRight());
     }
   }
}
```

```java
    private void printLeafNodes(BSTnode node) {
        // base case
        if (node != null) {
            if (node.getLeft() == null && node.getRight() == null) {
                System.out.printf("%d ", node.getData());
            } else {
                printLeafNodes(node.getLeft());
                printLeafNodes(node.getRight());
            }
        }
```

Output - TreesPackage (run)

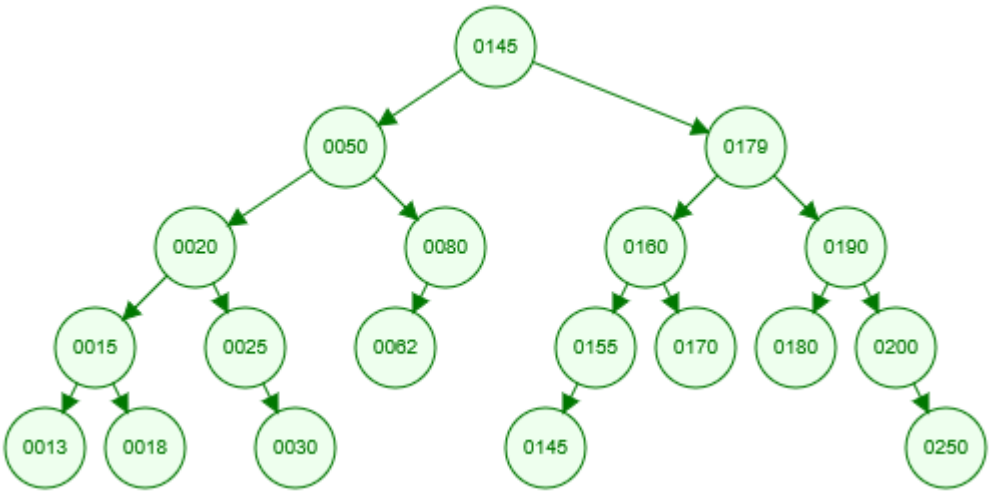
```
> Please enter your choice: 6
>    Inorder Traversal of nodes:
>    2, 4, 8, 9, 10, 11, 12, 14, 16, 18,

|-------------------------------------------------|
|------      Binary Search Tree Menu      ------|
|-------------------------------------------------|
|   1. Insert an item into the tree               |
|   2. Delete an item from the tree               |
|   3. Search for an item in the tree             |
|   4. Find the parent of some node               |
|   5. Print the sum of all data values           |
|   6. Print an inorder traversal of the tree     |
|   7. Print postorder traversal of the tree      |
|   8. Print All leaf nodes                       |
|   9. Count nodes which are divisible by 7       |
|   10. Modify all nodes by adding some value     |
|   11. Quit                                      |
|-------------------------------------------------|

> Please enter your choice: 8
> All leaf nodes:
>    2 9 11 14 18
```

Question 2: (20 points)

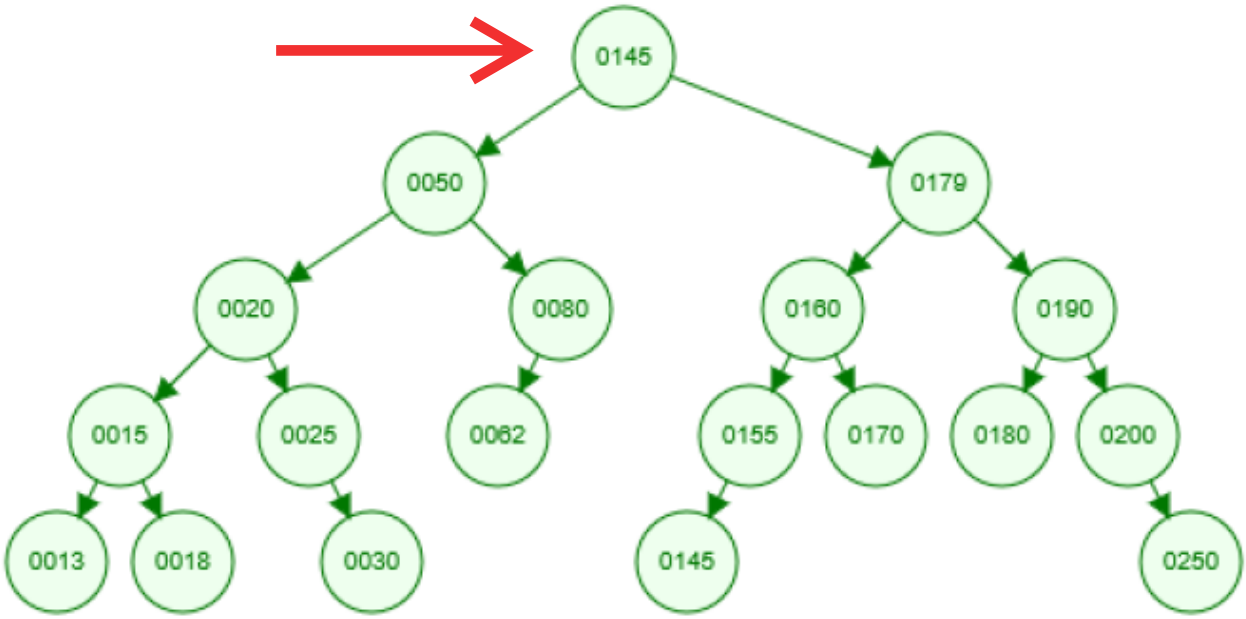
Suppose we have the following Binary Search Tree



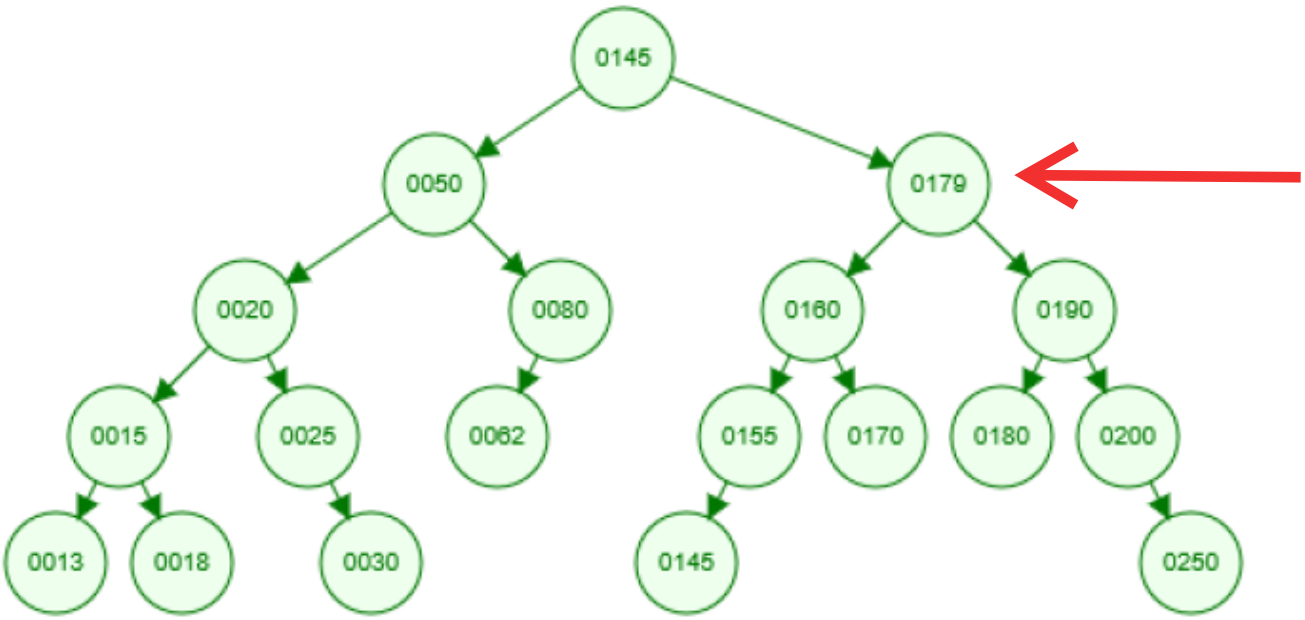
Show the complete tracing of **recurSearch**() method (given in the code of BST) to find the node 0155. You are required to draw recursive trees or boxes to show the detailed working of this method.

Show the output and the detailed working of this method.

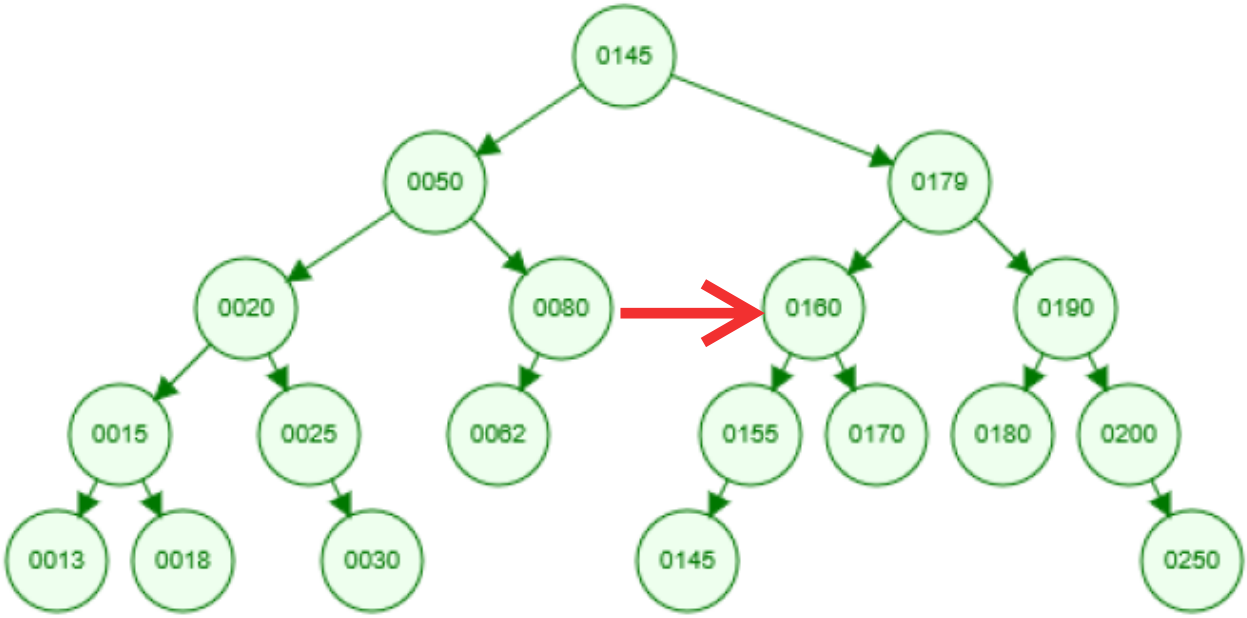
- Strart from root (0145)
- Check if data of root is equal to data I need (0155)
- It is not equal
- Then check if data is less than data of root pointer will go to left node else go to right node
- It is greater than pointer will go to right node



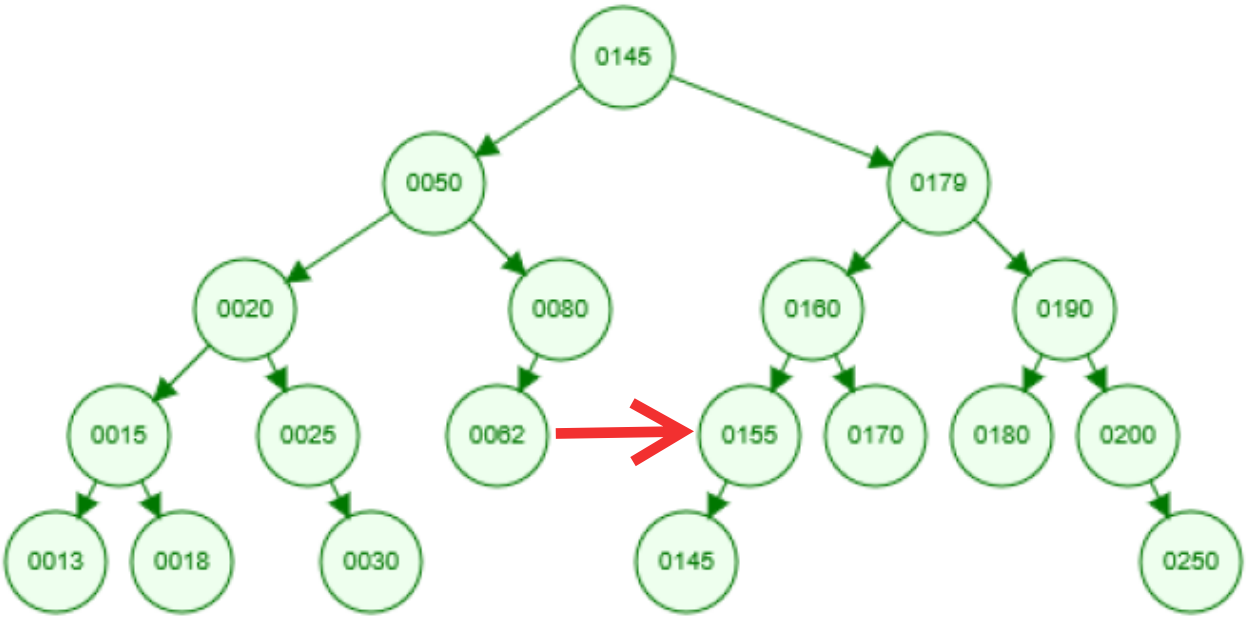
- Root now is (0179)
- Check if data of root is equal to data I need (0155)
- It is not equal
- Then check if data is less than data of root pointer will go to left node
  else go to right node
- It is less than pointer will go to left node



- Root now is (0160)
- Check if data of root is equal to data I need (0155)
- It is not equal
- Then check if data is less than data of root pointer will go to left node
  else go to right node
- It is less than pointer will go to left node



- Root now is (0155)
- Check if data of root is equal to data we need (0155)
- It is equal we found a node
- Then return true




```
public boolean recurSearch(int data) {
    return recurSearch(root, data);
}

private boolean recurSearch(BSTnode p, int data) {
    if (p == null) {
        return false;
    } else {
        // if the data we are searching for is found at p (at the current root)
        if (data == p.getData()) {
            return true;
        } else if (data < p.getData()) {
            return recurSearch(p.getLeft(), data);
        } else {
            return recurSearch(p.getRight(), data);
        }
    }
}
```