Binary Search Trees (BST)

Objective

The objective of this lab is to understand the implementation of binary Search Tree using linked list.

Task

```
class node<T> {
  T data;    node<T> left;       node<T> right;

  node(T d){
  data=d;
  }
}

public class BST<T extends Comparable<T>> {
    node<T> root;

  public void insert(T key){  //  required non-recursive implementation
    {    /* Algorithm steps
          1.   If root is null then Insert key as root node into the tree
          2.   Otherwise iteratively check until null
          3.   if key is less than the data store in a node, if so then move to the left sub-tree.
          4.   If not, then move to the right sub-tree.
          5.   Insert new node with key where null is found
    */   }

   public void LNR(node n) //  required recursive implementation
     {    /* Algorithm steps
             Base case: if n==null return
             else  Recursive case LNR(n.left);   Print  n.data;    Recursive case LNR(n.right).
       */     }

  public node find(T key)   //  required non-recursive implementation
    {  /* Algorithm steps
          1.   Compare the input key with the data stored in a root node of the tree.
          2.   If the key is matched, then return the address of the node.
          3.   Otherwise iteratively check if key is less than the data store in a node, if so then move to the left
               sub-tree. If not, then move to the right sub-tree.
          4.   Repeat step 3 iteratively until match found.
          5.   If key is not found, then return NULL.
    */   }

  public node Minimum()   //  required non-recursive implementation
    {     // in binary search tree minimum can be find by getting left most node of a tree     }

 public node Maximum()   //  required non-recursive implementation
    {     // in binary search tree minimum can be find by getting right most node of a tree   }

}
```