FULL STACK DEVELOPMENT – WORKSHEET - 6

Ques 1. Write a java program that inserts a node into its proper sorted position in a sorted linked list.

Ans 1:

```
class LL
{
  Node head;
  class Node
  {
    int data;
    Node next;
      Node (int z)
    {
      data = z;
      next = null;
    }
  }
  private Node tail;
  private int size;
   public int size ()
  {
    return this.size;
  }
 public boolean isEmpty ()
  {
```

```
    return this.size () == 0;
  }
   public Node insert (int data)
   {
    Node newNode = new Node (data);
   newNode.next = head;
    head = newNode;
    return head;
   }
   public void display ()
   {
    Node node = head;
    while (node != null)
      {
          System.out.print (node.data + " ");
          node = node.next;
      }
    System.out.println ("\n");
   }
   public void sortedInsert (int data)
   {
    this.sortedInsert (this.head, data);
   }
   private void sortedInsert (Node node, int data)
   {
    Node newNode = new Node (data);
```

```
    if (node == null || data <= node.data)
     {
          newNode.next = this.head;
          this.head = newNode;
     }
    else
     {
          while (node.next != null && node.next.data < data)
           {
             node = node.next;  }
          newNode.next = node.next;
          node.next = newNode;
     }
  }
}
public class Main
{
  public static void main (String[]args) throws Exception
  {
    LL ll = new LL ();
      ll.insert (82);
      ll.insert (69);
      ll.insert (37);
      ll.insert (29);
      System.out.println ("before Insertion");
      ll.display ();
```

```
    ll.sortedInsert (4);
    System.out.println ("after Insertion");
    ll.sortedInsert (6);
    ll.display ();
  }
}
```

Ques 2. Write a java program to compute the height of the binary tree.

Ans2:

```
class Tree {
   int data;
   Tree left, right;
   Tree(int item) {
      data = item;
      left = right = null;
   }}
public class BinaryTree {
   Tree root;
   int findHeight(Tree Tree) {
      if (Tree == null)
         return 0;
      else {
         int leftHeight = findHeight(Tree.left);
         int rightHeight = findHeight(Tree.right);
         if (leftHeight > rightHeight)
            return (leftHeight + 1);
         else
```

```
            return (rightHeight + 1);
        }
    }
    public static void main(String[] args) {
      BinaryTree tree = new BinaryTree();
        tree.root = new Tree(1);
        tree.root.left = new Tree(2);
        tree.root.right = new Tree(3);
        tree.root.left.left = new Tree(4);
        tree.root.left.right = new Tree(5);
        System.out.println("The height of binary tree is: " +
            tree.findHeight(tree.root));
    }
}
```

Ques 3. Write a java program to determine whether a given binary tree is a BST or not.

Ans3:

```
class BinaryTree {
    static class Node {
        int value;
        Node left;
Node right;
        Node(int value) {
            this.value = value;
        }
    }
   public static boolean isValidBST(Node root) {
```

```
        return isValidBST(root, null, null);
    }
    public static boolean isValidBST(Node root, Integer max, Integer min) {
        if (root == null) return true;
        if (max != null && root.value >= max) return false;
        if (min != null && root.value <= min ) return false;
        return isValidBST(root.left, root.value, min) &&
            isValidBST(root.right, max, root.value);
    }
    public static void main( String args[] ) {
        Node root = new Node(18);
        root.left = new Node(3);
        root.right = new Node(5);
        root.left.left = new Node(6);
        root.left.left.left = new Node(2);
        root.left.right = new Node(15);
        root.right.left = new Node(11);
        root.right.left.right = new Node(9);
        root.right.right = new Node(10);
        if (BinaryTree.isValidBST(root))
            System.out.println("A valid BST");
        else
            System.out.println("Not a valid BST");
    }
}
```

Ques 5. Write a java program to Print left view of a binary tree using queue.

Ans5:

```
class nodes
{

int val;


nodes lt, rt;


public nodes(int j)
{
val = j;
rt = null;
lt = null;
}
}
public class LeftViewExample1
{
nodes r = null;

public void displayLeftView()
```

```
{
if (r == null)
{
return;
}
Queue<nodes> que = new LinkedList<>();
que.add(r);
while (!que.isEmpty())
{

int siz = que.size();

for (int j = 1; j <= siz; j++)
{
nodes tmp = que.poll();



if (j == 1)
{
System.out.print(tmp.val + " ");
}

if (tmp.lt != null)
{
que.add(tmp.lt);
}
```

```
if (tmp.rt != null)
{
que.add(tmp.rt);
}
}
}
}

public static void main(String argvs[])
{

LeftViewExample1 lv = new LeftViewExample1();

lv.r = new nodes(20);

lv.r.lt = new nodes(22);
lv.r.rt = new nodes(8);
lv.r.lt.lt = new nodes(25);
lv.r.lt.rt = new nodes(3);
lv.r.rt.rt = new nodes(5);
lv.r.lt.rt.rt = new nodes(10);
lv.r.lt.rt.lt = new nodes(14);
lv.r.lt.rt.lt.rt = new nodes(7);
System.out.println(" left view of the Binary Tree: ");
```

```
lv.displayLeftView();
}
}
```