|  |  |
| --- | --- |
| Student Name | Asadullah |
| Roll Number | 21SW036 |
| Section # | 03 |
| Lab # | 12 |

**Task#01**

Question statement

Create a Binary Search Tree for the following data.

6,8,22,3,7,5,12,10,9,20,35,40,42

# Q1.Java

**Code:**

package Lab\_12;  
  
class BST{  
 Node root;  
  
 static class Node{  
 int val;  
 Node left, right;  
 public Node(int val){  
 this.val =val;  
 }  
 } // end of class Node  
  
 public void insert(int val){  
 root = insert(root, val);  
 }  
 public Node insert(Node root, int val){  
 if (root==null){  
 root = new Node(val);  
 return root;  
 }  
 if (val<root.val){  
 root.left = insert(root.left, val);  
 } else {  
 root.right = insert(root.right, val);  
 }  
 return root;  
 } // end of insert  
  
 public void inOrder(){  
 inOrder(root);  
 }  
 public void inOrder(Node root){  
 if (root==null){ // base condition for recursive call  
 return;  
 }  
 inOrder(root.left);  
 System.*out*.print(root.val + " ");  
 inOrder(root.right);  
 }  
} // end of class BST  
public class Q1 {  
  
 public static void main(String[] args) {  
  
 BST bst = new BST();  
  
 bst.insert(6);  
 bst.insert(8);  
 bst.insert(22);  
 bst.insert(3);  
 bst.insert(7);  
 bst.insert(5);  
 bst.insert(12);  
 bst.insert(10);  
 bst.insert(9);  
 bst.insert(20);  
 bst.insert(35);  
 bst.insert(40);  
 bst.insert(42);  
  
 bst.inOrder();  
  
 }  
}

**Output:**

****

**Task#02**

Question statement:

Create a Binary Search Tree for the following data.

A B C D E F G H I J K L M N O P

# Q2.Java

**Code:**

package Lab\_12;  
  
public class Q2 {  
  
 static class BST{  
 Node root;  
  
 static class Node{  
 String val;  
 Node left, right;  
 public Node(String val){  
 this.val =val;  
 }  
 } // end of class Node  
  
 public void insert(String val){  
 root = insert(root, val);  
 }  
 public Node insert(Node root, String val){  
 if (root==null){ // base condition to stop recursive call  
 root = new Node(val);  
 return root;  
 }  
 if (val.compareTo(root.val) < 0){  
 root.left = insert(root.left, val);  
 } else {  
 root.right = insert(root.right, val);  
 }  
 return root;  
 } // end of insert  
  
 public void inOrder(){  
 inOrder(root);  
 }  
 public void inOrder(Node root){  
 if (root==null){ // base condition for recursive call  
 return;  
 }  
 inOrder(root.left);  
 System.*out*.print(root.val + " ");  
 inOrder(root.right);  
 }  
} // end of class BST  
   
 public static void main(String[] args) {  
  
 BST bst = new BST();  
  
 bst.insert("A");  
 bst.insert("B");  
 bst.insert("C");  
 bst.insert("D");  
 bst.insert("E");  
 bst.insert("F");  
 bst.insert("G");  
 bst.insert("H");  
 bst.insert("I");  
 bst.insert("J");  
 bst.insert("K");  
 bst.insert("L");  
 bst.insert("M");  
 bst.insert("N");  
 bst.insert("O");  
 bst.insert("P");  
  
 bst.inOrder();  
  
 }  
}

**Output:**

****