1. (a) Show the result of inserting 20, 35, 24, 17, 28, 40, 25, 15, 16 into an initially empty Binary Search Tree in the given order.

(b) Show the result of removing 35 from the tree created in part (a).

(c) What is the output of a post-order traversal of the tree created in part (a)

(d) Draw all minimal height binary search trees which store the following numbers:
    4, 8, 16, 32, 64.

2. Consider the BinaryTree class partially given below. Add to the BinaryTree class a method **Public boolean equals(Object T)** which would return true if "this" tree is the same as T, otherwise it would return false.

```java
public class BinaryTree<E>{
    public class TreeNode<E>{
        private <E> data;
        private TreeNode<E> left;
        private TreeNode<E> right;

        public TreeNode(E newData){
            data = newData;
            left = null;
            right = null;
        }
    } // end class TreeNode

    private TreeNode<E> root;

    public BinaryTree(){
        root = null;
    }

    // other methods
}
```
3. Consider the SinglyLinkedList class partially given below. Instance variable `head` contains the address of the first node, and `tail` contains the address of the last node. Add to the `SinglyLinkedList` class a method `public void insertFront(Object newData)` which inserts a node at the front of the list.

```java
public class SinglyLinkedList<E>{
    // other methods
}
```

```java
3. Consider the SinglyLinkedList class partially given below. Instance variable `head` contains the address of the first node, and `tail` contains the address of the last node. Add to the `SinglyLinkedList` class a method `public void insertFront(Object newData)` which inserts a node at the front of the list.

```java
public class SinglyLinkedList<E>{
    public class Node<E>{
        private E data;
        private Node<E> next;

        public Node () {
            data = null;
            next = null;
        }

        public Node( E newData, Node<E> newNext ){
            this.data = newData;
            this.next = newNext;
        }
    } // end of Node class

    private Node<E> head; // Reference to first node in the list
    private Node<E> tail; //Reference to the last node in the list

    public SinglyLinkedList( ){
        head = tail = null;
    }

    //other methods
} // end of SinglyLinkedList class
```

4. Suppose you have a stack S containing n elements and a queue Q that is initially empty. Give an algorithm to scan S to see if it contains a certain element x, with the additional constraint that your algorithm must return the elements back to S in their original order. You may not use any storage except S and Q. What is the running time of your algorithm in Big-Oh notation? Justify your answer.