Aufgabenblatt: Binäre Bäume

(1.) (a.) Implementieren Sie die Methoden ohne Methodenrumpf der Klasse CNode!
(b.) Erstellen Sie eine Klasse CBinaryTree mit den notwendigen minimalen Funktionen, dass Elemente zugefügt werden können und der Baum auf drei verschiedenen(welche sind das???) Arten durchlaufen wird und die Elemente ausgegeben werden!

```
/* Class CBinaryTree */
 class CBinaryTree
 {
    private CNode root;

    /* Konstructor */
    public CBinaryTree()
    {
       root = null;
    }
    public boolean isEmpty()
    {
       ……..
    }
    /* Functions to insert data */
    public void insert(int data)
    {
       root = insert(root, data);
    }
    /* Function to insert data recursively */
    private CNode insert(CNode node, int data)
    {
       if (node == null)
          node = new CNode(data);
       else
       {
          if (node.getRight() == null)
             node.right = insert(node.right, data);
          else
             node.left = insert(node.left, data);
       }
       return node;
    }
    /* Function to count number of nodes */
    public int countNodes()
    {
       return countNodes(root);
    }
    /* Function to count number of nodes recursively */
    private int countNodes(CNode r)
    {
       if (r == null)
       ……..
       else
       {
          int l = 1;
          ……..
          // count nodes on the left!
          ……..
          // count nodes on the right!
          ……..
          return l;
       }
    }
```

(c.) Erstellen Sie ein Testprogramm und testen Sie ihren Baum!
(d.) Verwenden Sie anschließend eine Klasse der Wahl um die Daten im Baum zu speichern!

```
    /* Function to search for an element */
    public boolean search(int val)
    {
        return search(root, val);
    }
    /* Function to search for an element recursively */
    private boolean search(CNode r, int val)
    {
        if (r.getData() == val)
            …….
        if (r.getLeft() != null)
            // search left & return ???!
            …….
        if (r.getRight() != null)
            // search right & return ???!
            …….
        return false;
    }

    /* Function for inorder traversal */
    public void inorder()
    {
        inorder(root);
    }
    private void inorder(CNode r)
    {
        if (r != null)
        {
            // go left!
            …….
            System.out.print(r.getData() +" ");
            // go right!
            …….
        }
    }
    /* Function for preorder traversal */
    public void preorder()
    {
            // do it analog inorder!!!
            …….
    }
    private void preorder(CNode r)
    {
            // do it analog inorder!!!
            …….
    }

    /* Function for postorder traversal */
    public void postorder()
    {
            // do it analog inorder!!!
            …….
    }
    private void postorder(CNode r)
    {
            // do it analog inorder!!!
            …….
    }
}
```