Opgave 1

O (n^2)

There are three loops. The outer while will run n-times. The for loop inside will run n times for each time the while loop runs. The inner-most for loop runs 10 times, and is not dependent on n. Thus we get n·n·10. 10 is a constant and is not taken into account. That is, we get $O(\ n \cdot n\ )$, or $O(\ n^2\ )$

Opgave 2

Many of you got this one wrong. Notice how the if statement in the loop is build.
Alternatively, instead of the if statement one can just write:

tm.put(arr[i], tm.getOrDefault(arr[i], 1);

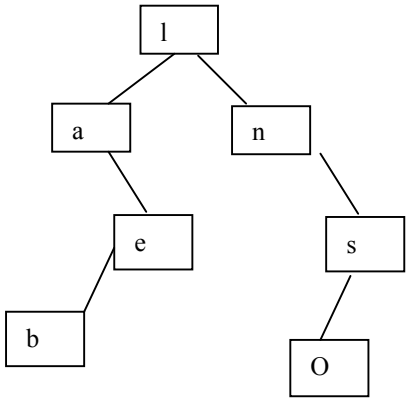
```
// Count number of occurrences of each number in the array
    public void frequency (int[] arr)
    {
        //-- Use a TreeMap to store a counter value for each number(key)
        // A sorted set of the numbers (keys) is then directly available
        // key = the integer value in array
        // value = "counter"
        TreeMap<Integer, Integer> tm = new TreeMap();
        for (int i = 0; i < arr.length; i++)
        {
            if (tm.containsKey(arr[i]))
            {
                tm.put(arr[i], tm.get(arr[i]) + 1);
            } else
            {
                tm.put(arr[i], 1);
            }
        }
        //-- Traverse using a set of keys (the numbers) in ascending order
        Set navSet = tm.navigableKeySet();
        System.out.println("Elements : Number of occurrences");

        Iterator it = navSet.iterator();
        while (it.hasNext())
        {
            Integer element = (int) it.next();
            System.out.println(element + " : " + tm.get(element));
        }
    }
```

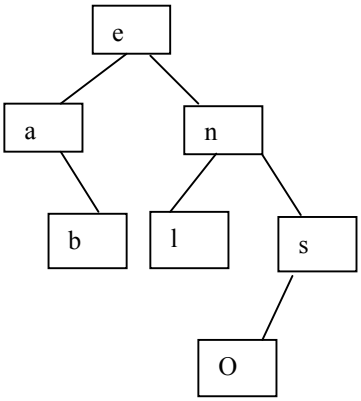
Opgave 3

HashMap indsætter og søger i konstant tid O(1), så vælg den.

Opgave 4

b) 2 alternativer – afhænger af om største i venstre eller mindste i højre vælges som ny rod





Opgave 5

a)

```
 public int sum()
    {
        return sumRec(root);
    }

    private int sumRec(BinaryTreeNode node)
    {
        int sum = 0;
        if (node == null)         // base case
            return sum;

        sum = (int)node.element;
        sum += sumRec(node.left);
        sum += sumRec (node.right);
        return sum;
    }
```

Opgave 6 (Ekstra):

```
public String preorderTraversal()
    {
        String s = new String();
        s = preorderTraverse(root, s);
        return s;
    }

    private String preorderTraverse(BinaryTreeNode n, String s)
```

```
    {
        if (n != null)
        {
            s += n.element + " ";
            s = preorderTraverse(n.left, s);
            s = preorderTraverse(n.right, s);
        }
        return s;
    }
```