1.

Red and black trees must satisfy the condition that the path from a node to any of its descendents NULL nodes must have the same number of black nodes.

```
Function getBlackHeight( root node)
        If root node is NULL
                Return 0;
        Endif
        Int leftHeight is getBlackHeight(root node left child)
        Int rightHeight is getBlackHeight(root node right child)
        Int counter is 1 if rootnode is black, 0 if red
        If leftHeight or rightHeight are -1, or if they are not equal
                Return -1
        Else
                Return rightHeight + counter
        Endif
End Function
```

This function will return -1 if it is not a red and black tree, or will return the black Height if it is a red and black tree

2.
    i. Not a red and black tree as the paths from root to null nodes are not the same for the whole tree.there are 2 black nodes from root to 41's NULL leaf, and 1 black node from root to 0's NULL leaf
    ii. Valid tree
    iii. Valid tree
    iv. Valid tree
3. The height of a red balck tree with n nodes is 2 * log2(n + 1), so a tree of 14 nodes would have max height of 2 * log2(15) = ~7.8 so the height is 7

4.

```
Create new array inputArray[]
n = size of array
Counter starts at 0
Create new OS tree
For each element in array
        Count how many larger elements precede it in array
        Add this number to counter
endFor
```

5.

```
Func listIntervalOverlap(T,x,i)
        If i overlaps with x
                Print x
        If left of x != NULL and max(left of x) is greater than or equal to low(i)
                listIntervalOverlap(T,left of x,i)
        If right of x != NULL and max(right of x) is greater than or equal to low(i) and low(x) is
        less or equal to high(i)
                listIntervalOverlap(T,right of x,i)
        Endif
        Return x

End func
```