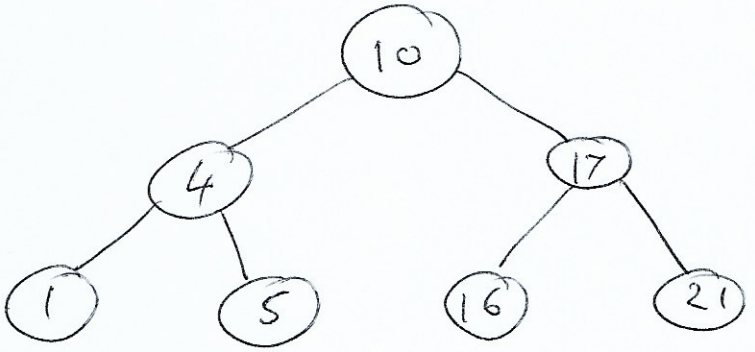
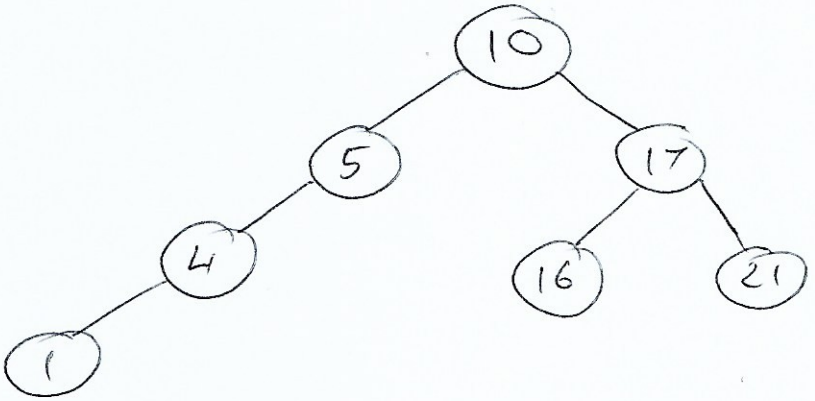
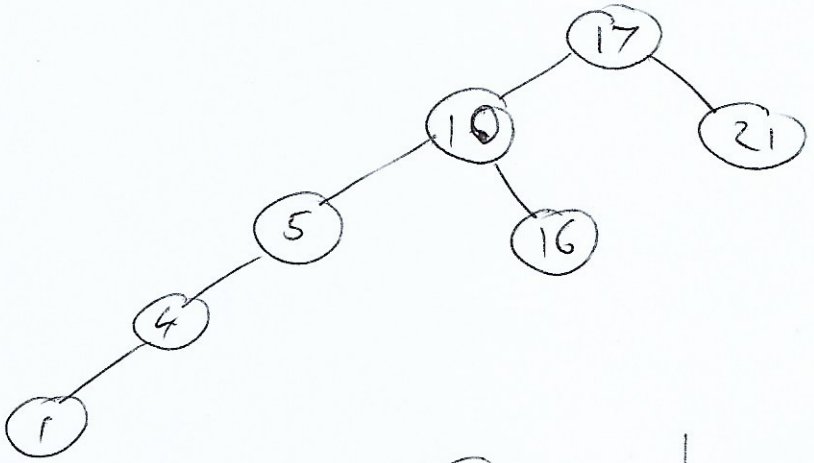
12.1 - 1



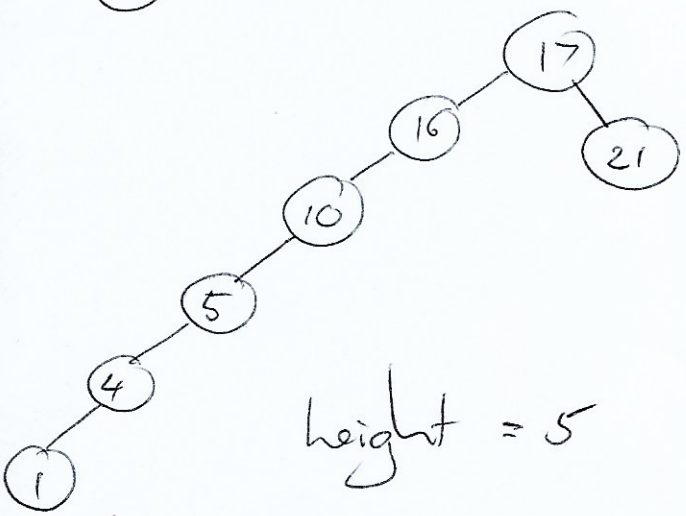
height = 2



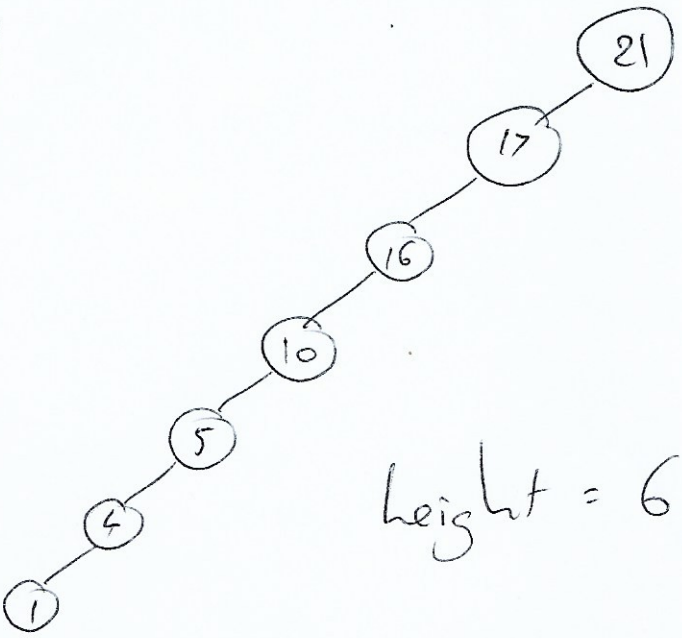
height = 3



height = 4



height = 5



height = 6

12.1 - 5

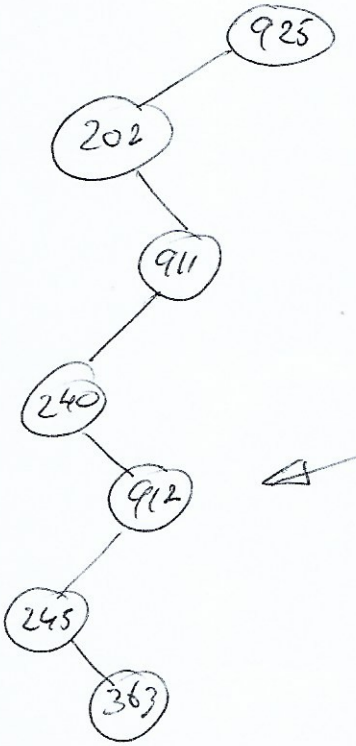
Given a list of keys, suppose we build a binary search tree using a comparison-based algorithm.
If we then run InOrderTreeWalk on the tree, we will get a sorted list of keys.
This is therefore a comparison sort algorithm.

Furthermore, we know that comparison sort algorithms run in $\Omega(n \log n)$, so this algorithm must also run in $\Omega(n \log n)$.
Since the InOrderTreeWalk is $\Theta(n)$, it must be that the algorithm for constructing the binary search tree is $\Omega(n \log n)$

12.2 - 1

a., b., are valid

c. is not :



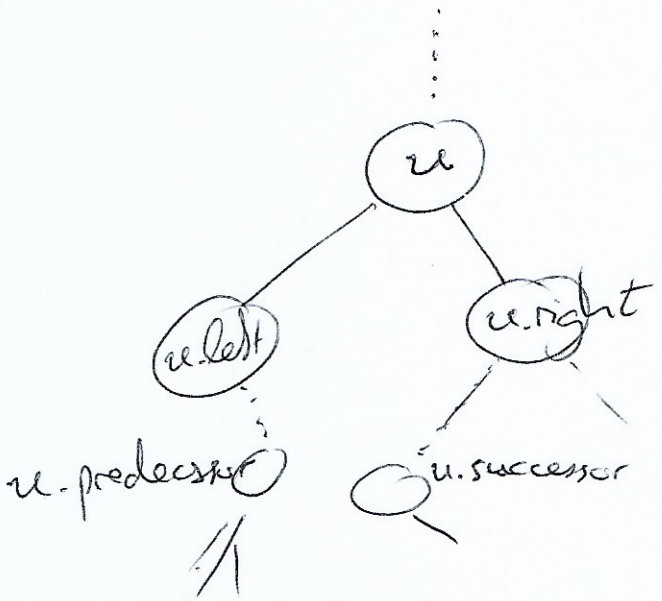
Here 912 is in the left subtree of 911, which violates the binary search tree property.

d. is valid

e. is not valid.

12.2-5

Say x has two childre:



x's successor is the minimum in the subtree of x.right. A minimum in a subtree has no left child.

x's predecessor is the maximum in the subtree of x.left. A maximum in a subtree has no right child.

12.3-2

The path followed when searching for a node is the same path followed when the node was inserted.
This is because all nodes inserted afterwards will not change the existing tree but be added as new leaves.
Thus, the number of nodes examined is one more than when inserting (one more because we examine also the node we seek.).