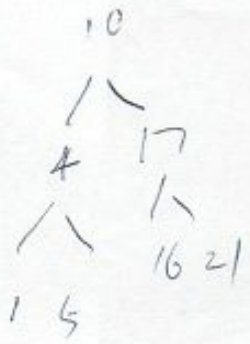
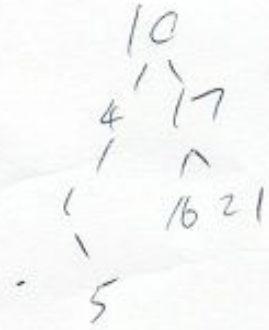
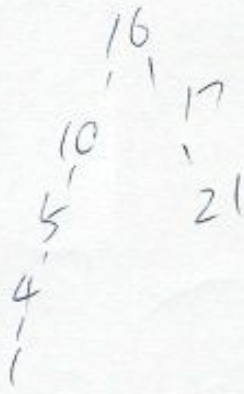
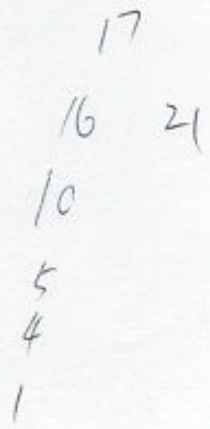
12.1-1

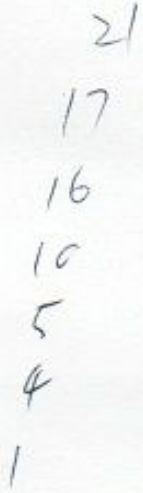
For the set of {1, 4, 5, 10, 16, 17, 21} of keys, draw binary search trees of heights 2, 3, 4, 5, and 6.











12.1-4

Give recursive algorithms that perform preorder and postorder tree walks in $\Theta(n)$ time on a tree of n nodes.

```
Preorder-walks(x)
  if x ≠ NIL
    Print key(x)
    Preorder-walks(left[x])
    Preorder-walks(right(x))
```

```
Postorder-walks(x)
  if x ≠ NIL
    Postorder-walks(left(x))
    Postorder-walks(right(x))
    Print key(x)
```

12.2-1

Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Which of the following sequences could *not* be the sequence of nodes examined?

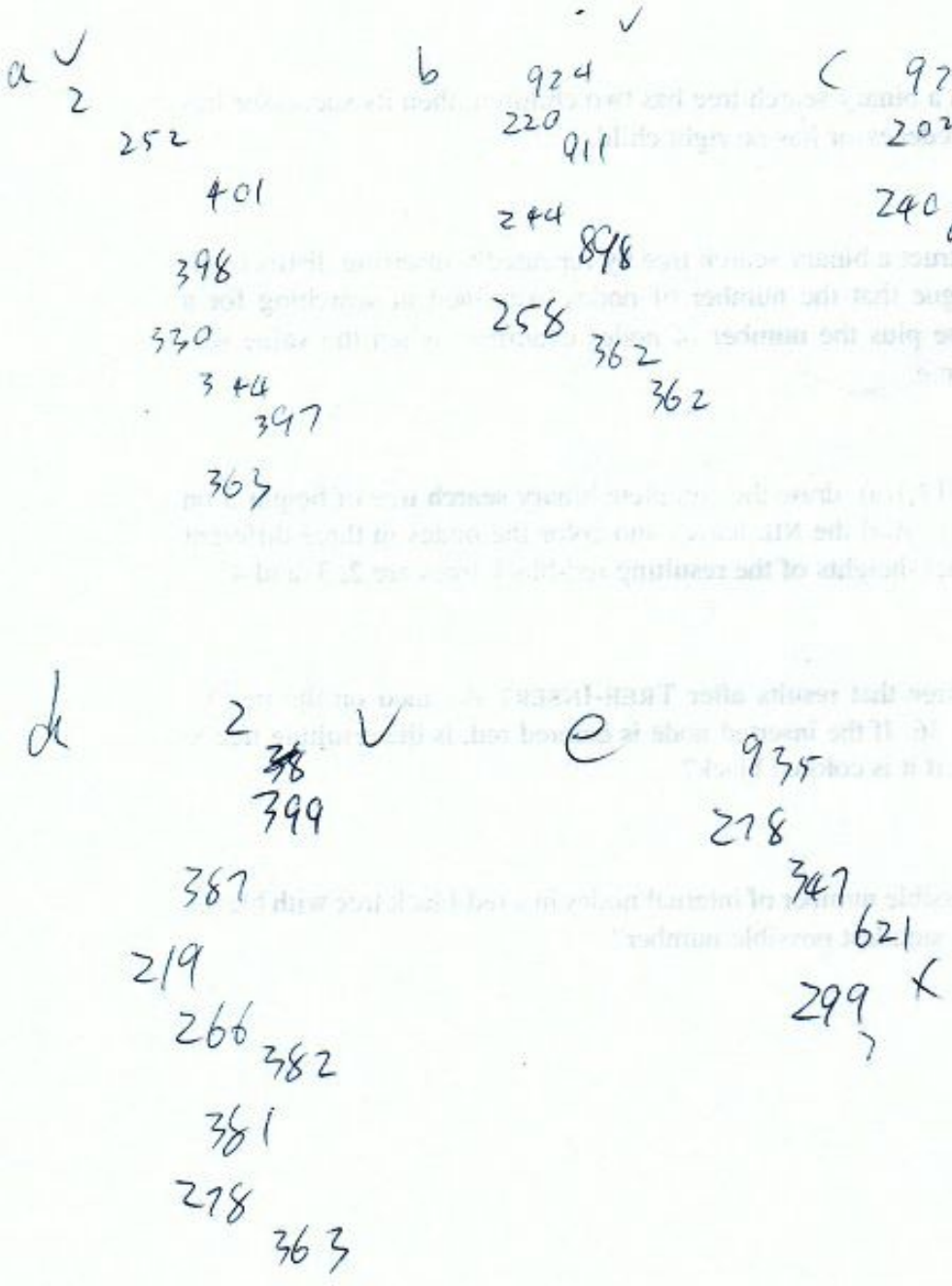
a. 2, 252, 401, 398, 330, 344, 397, 363.

b. 924, 220, 911, 244, 898, 258, 362, 363.

c. 925, 202, 911, 240, 912, 245, 363.

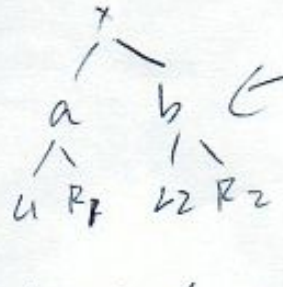
d. 2, 399, 387, 219, 266, 382, 381, 278, 363.

e. 935, 278, 347, 621, 299, 392, 358, 363.



12.2-5

Show that if a node in a binary search tree has two children, then its successor has no left child and its predecessor has no right child.



: in the subtree L_2. if L_2 is empty, b is successor. If x has a, a < x. so meaning is can't have left child

If a node has left child, predecessor should be the rightest node in left subtree. so predecessor don't has right child

12.3-2

Suppose that we construct a binary search tree by repeatedly inserting distinct values into the tree. Argue that the number of nodes examined in searching for a value in the tree is one plus the number of nodes examined when the value was first inserted into the tree.

search

Set To ~~insert~~ a value, need examine n nodes

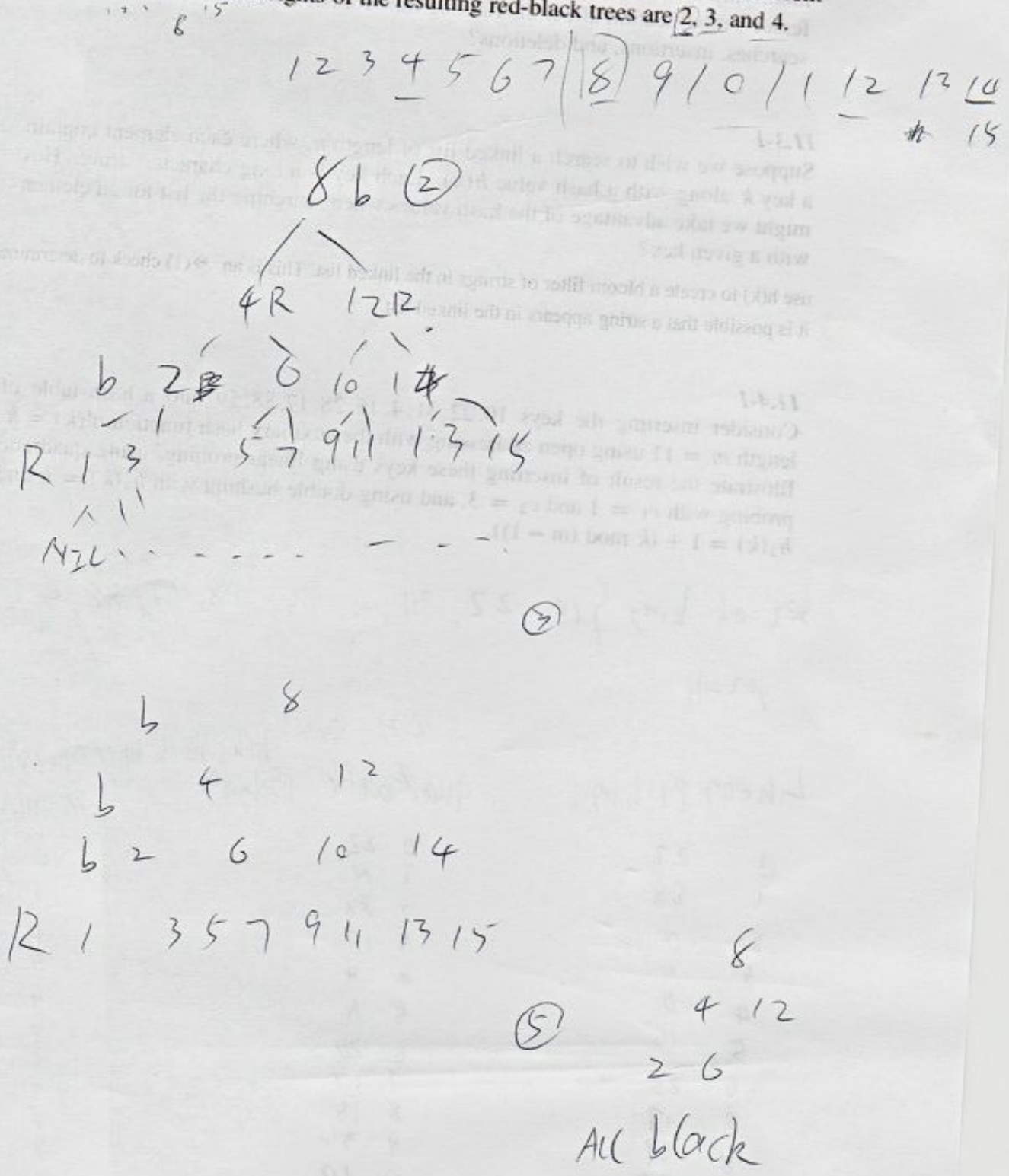
So insert a value, ~~#~~ need to examine n-1 nodes.

for 1st inserted into tree

$n-1+1=n$

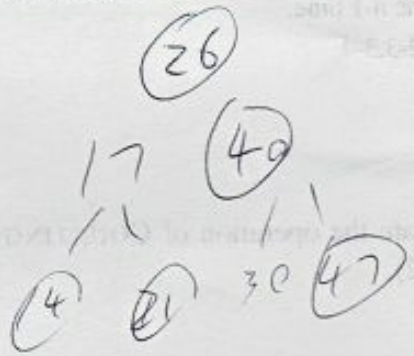
13.1-1

In the style of Figure 13.1(a), draw the complete binary search tree of height 3 on the keys $\{1, 2, \ldots, 15\}$. Add the NIL leaves and color the nodes in three different ways such that the black-heights of the resulting red-black trees are 2, 3, and 4.



13.1-2

Draw the red-black tree that results after TREE-INSERT is called on the tree in Figure 13.1 with key 36. If the inserted node is colored red, is the resulting tree a red-black tree? What if it is colored black?

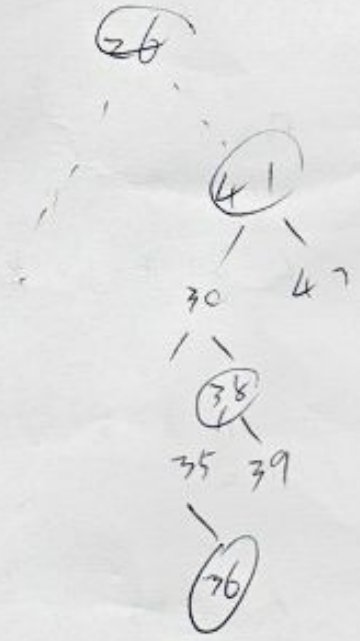


① Red

Not a red-black tree

36 is red, red can't have red child

② black



Not red-black tree

For each node, all simple paths from the node to descendant leaves contain the same number of black nodes