Problem Set 8, Part I

Problem 1: Checking for keys above a value

1-1)

The best case is that the key of the root is greater than v. The time efficiency is O(1). The worst case is that we traverse all the nodes of the tree and do not find any keys greater than v. In the worst case, the time efficiency is O(n) no matter whether the tree is balanced or not, because we use recursion to go down both the left and right sides of each node.

1-2)

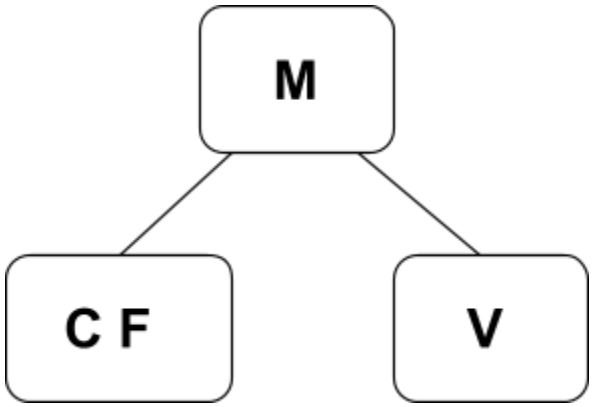
```
private static boolean anyGreaterInTree(Node root, int v) {
    if (root == null) {
        return false;
    } else {
        if (root.key > v) {
            return true;
        } else {
            return anyGreaterInTree(root.right, v);
        }
    }
}
```

1-3)

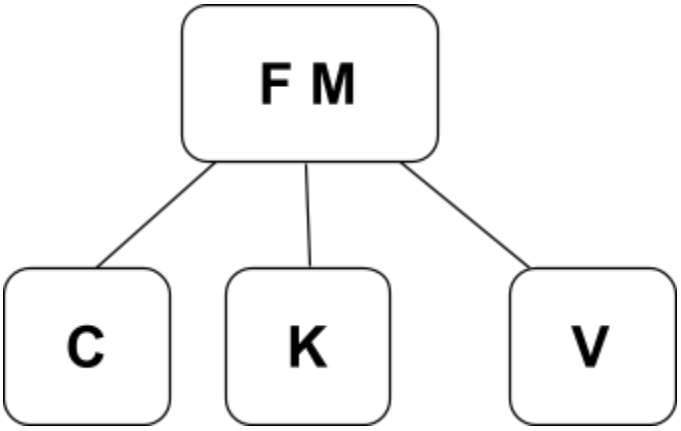
The best case is that the key of the root is greater than v. The time efficiency is O(1). The worst case is that there is no key greater than v, and it depends on the shape of the binary tree. If the tree is balanced, we have to go down the height of the tree, which is O(logn). Therefore, for a balanced binary tree, the time efficiency of the worst case is O(logn). If the tree is not balanced, the extreme case is that we have to go down the height of n-1. Therefore, for an unbalanced binary tree, the time efficiency of the worst case is O(n).

Problem 2: Balanced search trees

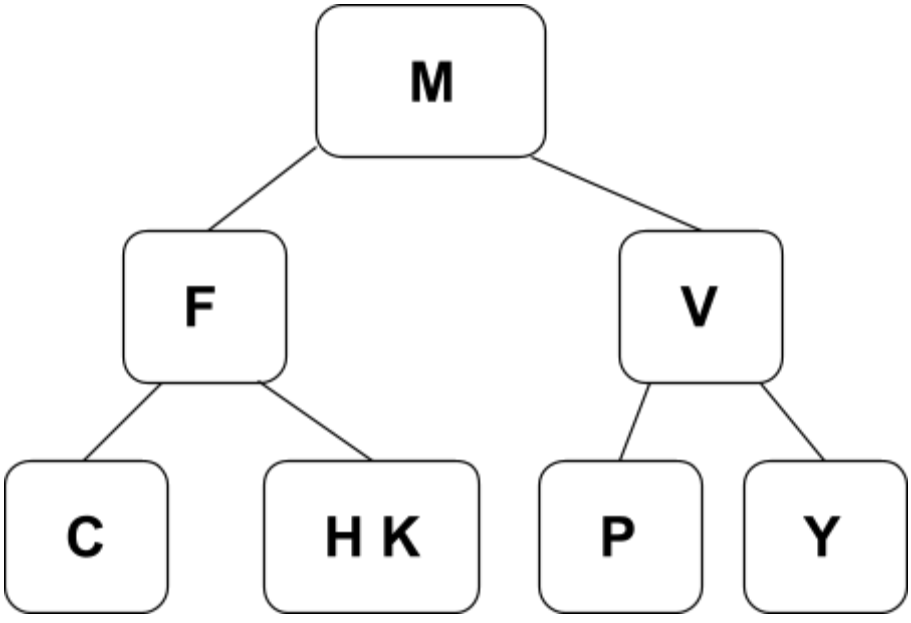
After the First Split:



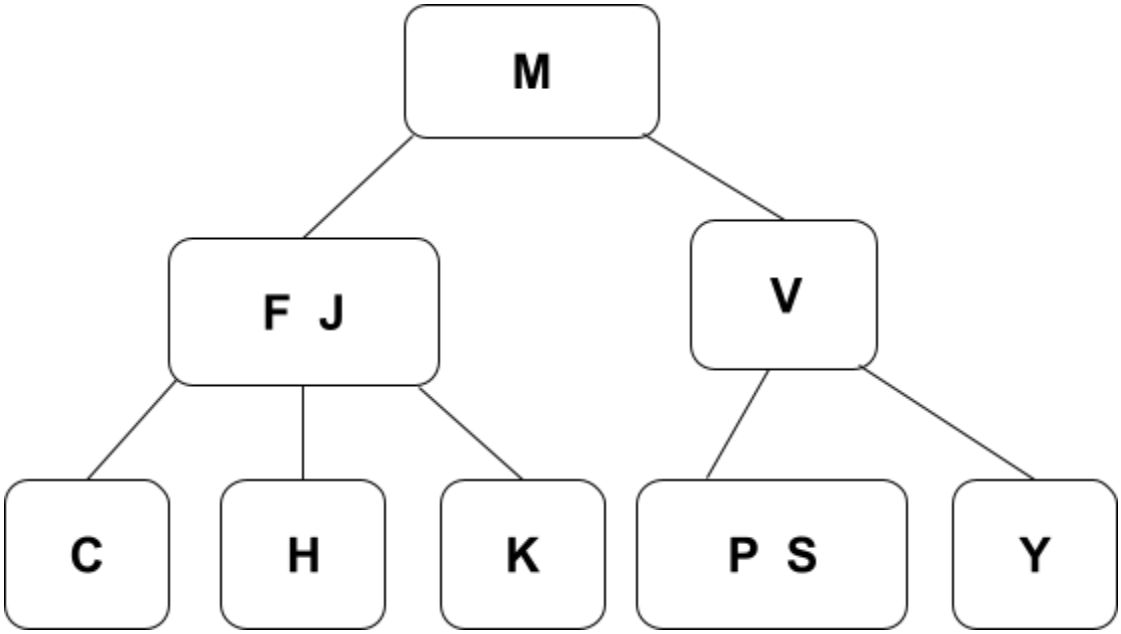
After the Second Split:



After the Third Split:



The Final Tree:



Problem 3: Hash tables

3-1) linear

0	you
1	a
2	to
3	the
4	my
5	their
6	bring
7	do

3-2) quadratic

0	
1	bring
2	to
3	the
4	
5	their
6	my
7	

3-3) double hashing

0	bring
1	do
2	to
3	the
4	
5	their
6	my
7	a

3-1) “go” will cause overflow.

3-2) “do” will cause overflow.

3-3) “you” will cause overflow.

3-4) probe sequence: 3 6 1 4 7

3-5) table after the insertion:

0	list
1	try
2	
3	our
4	
5	
6	linked
7	

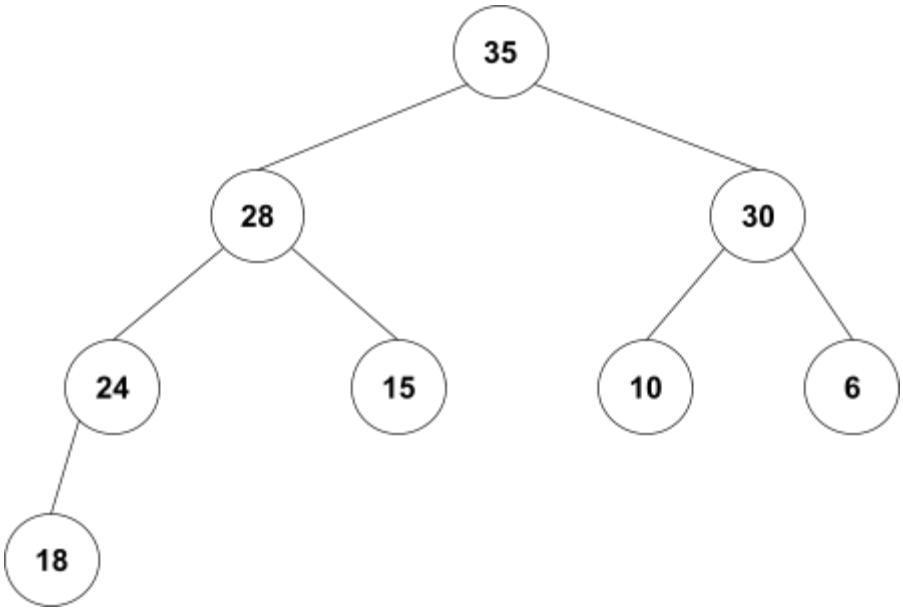
Problem 4: Complete trees and arrays

4-1) the position of the left child: 2*40 + 1 = 81
the position of the right child: 2*40 + 2 = 82
the position of the parent: (40-1)/2 = 19

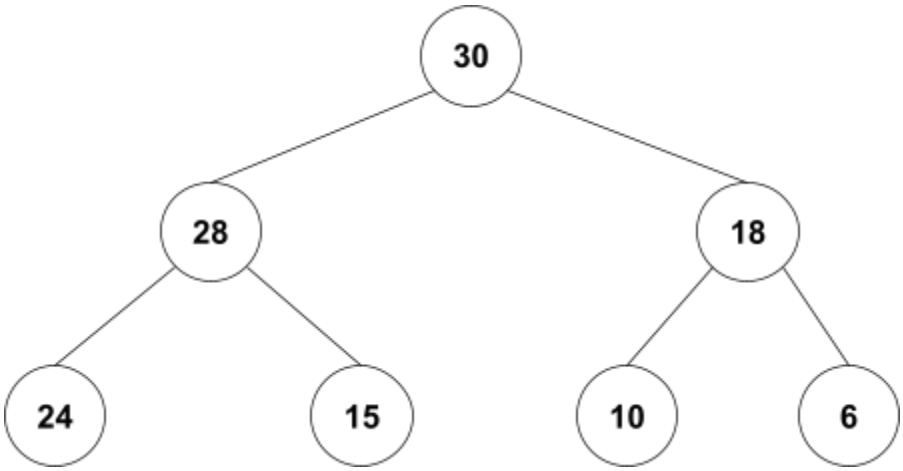
4-2)
1+2+4+8+16+32 = 63
1+2+4+8+16+32+64 = 127
As a complete binary tree, levels 0 through level 5 are fully occupied, and the height of the tree is 6.

4-3)
The rightmost leaf node in the bottom level is the last one in the array, so its index is 112. Because 112 is an even number, it's the right child of its parent.

Problem 5: Heaps
5-1)
after one removal



after a second removal
(copy your revised diagram from part 1 here, and edit it to show the result of the second removal)



5-2)