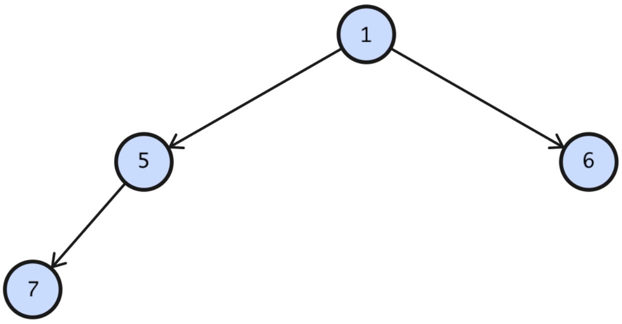
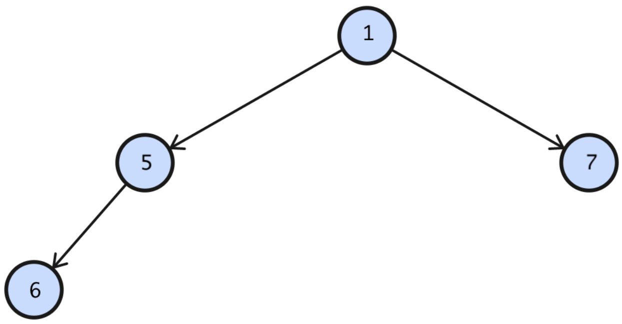
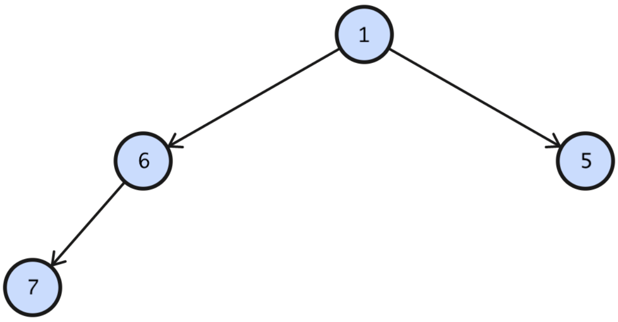
Title: Heaps and AVL Trees
Author: EFE ACER
ID: 21602217
Section: 3
Assignment: 3
Description: The answers of question 1 and 2 are given in this pdf.

Question 1:

(a) There are 3 such heaps:





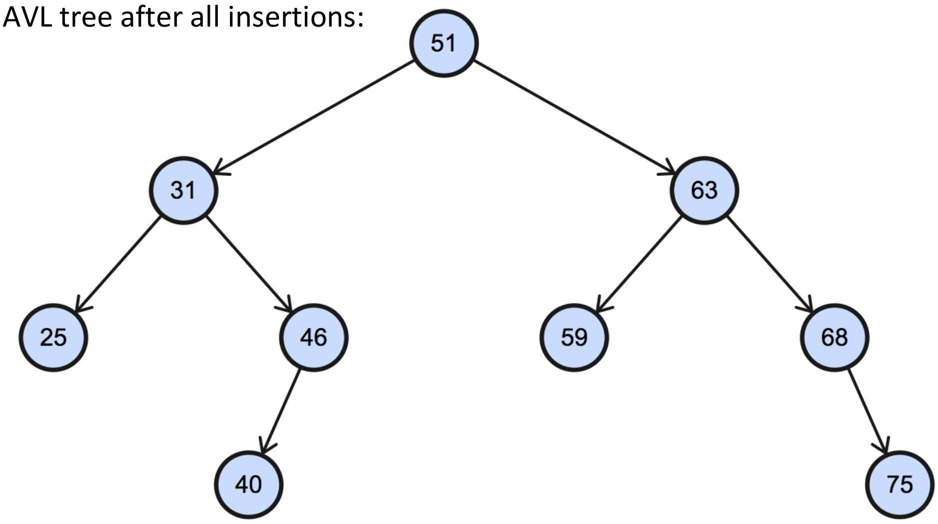


(b) Let these 5 distinct elements be a_1, a_2, a_3, a_4 and a_5 satisfying $a_5 > a_4 > a_3 > a_2 > a_1$. Then these 5 elements can be arranged in a max-heap as follows:

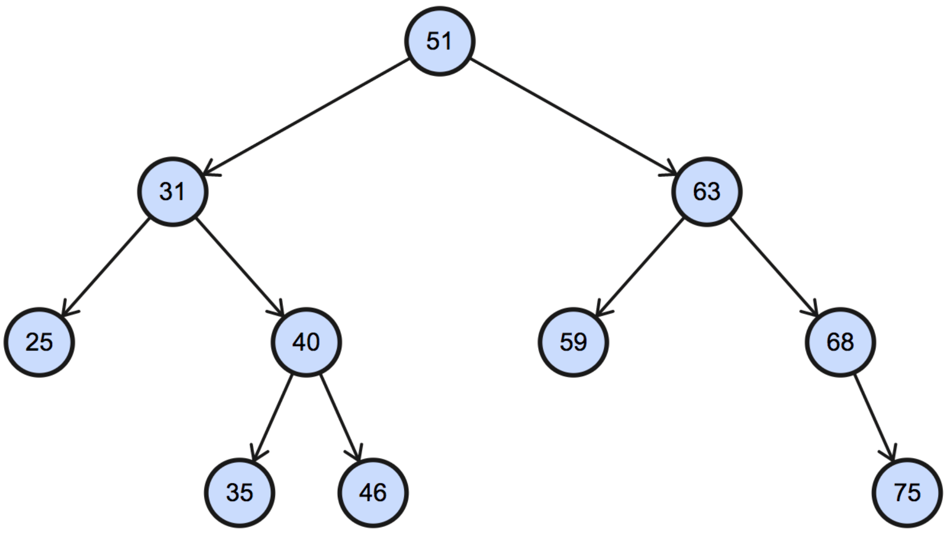
```
     a5            a5            a5            a5
    /  \          /  \          /  \          /  \
  a4    a3      a3    a4      a4    a1      a4    a2
 /  \          /  \          /  \          /  \
a2   a1       a2   a1       a2   a3       a1   a3

     a5            a5            a5            a5
    /  \          /  \          /  \          /  \
  a4    a3      a3    a4      a4    a1      a4    a2
 /  \          /  \          /  \          /  \
a1   a2       a1   a2       a3   a2       a3   a1
```

Hence, there are 8 valid max-heaps.

(c) i. AVL tree after all insertions:

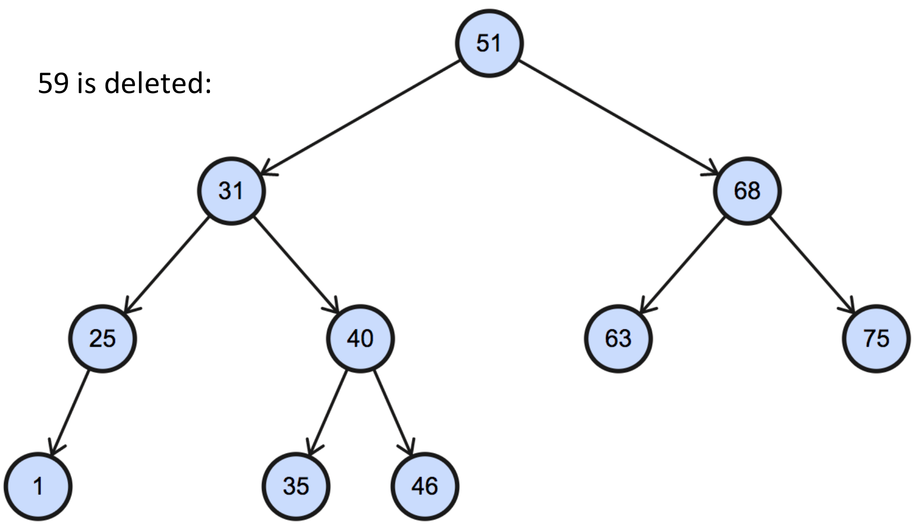


ii. For a single right rotation, an item should be added as the outmost and leftmost node in the subtree of a particular node such that the node becomes imbalanced. Such an item would be 35, and the tree after the insertion is the following:



iii. Single left rotation will occur in the "Case 3b" of the lecture slides, deleting 59 demonstrates the case. The final tree after is the following:

59 is deleted:



(d) The statement is true:

Consider the set {1, 2, 3, 4, 5}. When we insert the items in this set in increasing order to an initially empty AVL tree, we obtain:

```
     2
   /   \
  1     4
       / \
      3   5
```

Whereas, insertion in decreasing order gives the tree:

```
     4
    / \
   2   5
  / \
 1   3
```

As it can be seen from the given example, insertion order affects the resulting tree structure. The reason is that the insertion order determines the place of insertion for the next item, which indeed changes the type and place of rotations.

Question 2:

The results of the experiment performed by the program is given in the following table (Table 1):

Tree Size	Random	Ascending	Descending
1000	707	990	990
2000	1334	1989	1989
3000	2047	2988	2988
4000	2749	3988	3988
5000	3497	4987	4987
6000	4197	5987	5987
7000	4776	6987	6987
8000	5520	7987	7987
9000	6330	8986	8986
10000	7072	9986	9986

Table 1

- The complexity for insertion to an AVL Tree is O(logN) in the average case and O(N) in the worst case, since the number of rebalancing operations (rotations) vary according to the place of insertion that is affected by the insertion order. Since we have a finite range for the random number generation, it becomes more probable for the item to be inserted to be a part of a sorted subsequence. Hence the average number of rotations tends to be closer to O(N). Table 1 is in agreement with this reasoning.

- Different patterns of insertion do affect the number of rotations in the AVL Tree. The aim of any tree is to remain a non-linear, logarithmic structure. However, when a sequence of numbers that are ascending or descending is inserted, the tree becomes a linear structure, just like a linked list. An AVL Tree in particular performs rotations to avoid this linear structure. Thus, the number of rotations increases as the insertion pattern becomes sorted. Table 1 also agrees with this reasoning.