3 Search Trees, Heap, Union/Find

1. (a) Refer to the definition of the BST in p. 418 of the course book.
   
   (b) In the following solution Deletion uses in-order predecessor:

   12
   / \
  6  21
 / \ / \
5 10 18 25
 /\
7
\
9

   (c) No, the balance of the node with key 10 is 2.

2. We consider BST’s which do not allow multiple occurrences of a key.

   (a) Yes. Assume the contrary, i.e. assume that there exist two different BST
trees T1 and T2 with the same keys which have identical trace in the preorder
traversal. As T1 and T2 are different BST trees and have the same keys there
exist nodes n1 and n2 such that n1 is an ancestor of n2 in T1 and n2 is an
ancestor of n1 in T2. Thus, n1 will precede n2 in the preorder traversal of
T1 and n2 will precede n1 in the preorder traversal of T2. This contradicts
the assumption. Given the trace of the preorder traversal k1,...,kn the tree
is reconstructed by iterating the insertion operation, starting from the empty
BST: e.g. 5,4,12,7,8,15 results in

   5
   / \
  4 12
 / \
7 15
\ 
 8

   Notice that there are sequences of keys which are not traces of the pre-order
   traversal of a BST, e.g. 5,12,4

   (b) Yes. A justification is similar to that given above. The tree can be reconstructed
by insertion of the keys from right to left. For example 4,8,7,15,12,5 inserted
in the reverse order 5,12,15,7,8,4 will reconstruct the tree of example a.

   (c) No. Both trees

   5
   / \ \
4  6 5
  / \ 
4  6

12
(d) Yes. The maximal growing subsequences of a given sequence correspond to the levels of the tree. The elements of incomplete levels should be distributed according to the definition of BST.

For example 10, 5, 15, 3, 12, 20 gives the tree

```
    10
   / \
  5   15
 /   / \
3   12 20
```

Notice that there are sequences of keys which cannot be obtained as a result of the level-order traversal of a BST.

3. For every tree we compute the balance of each node. If for some node the balance is not in \([-1, 1]\) then the tree is not AVL.

The trees with AVL topologies are thus: A, B, F

4. Try to find a counterexample. We can find it for \(h = 2\):

```
  h  max n  min n
   h  h+1
0  0   1   1   2
1  1   3   3   4
2  2   7   7
```

13
5. (a) We count and indicate the balance of each node: the trees T1 and T2 are AVL trees since the balance of each node is 1, 0 or -1. The remaining ones are not AVL trees.

(b) We only show result of \textit{Insert}(41)

(c) i. First do \textit{Splay}(4). As 4 is not in the tree the algorithm brings 3 to the root by means of double rotation. Then 4 is inserted as the new root.

ii. \textbf{Advantages}: Splay tree needs no extra information in the nodes. It is very easy to implement. It works particularly well if the same key is accessed several times in row (since the accessed key is always brought to the root).

\textbf{Disadvantage}: The worst case time of a single operation may be linear on the number of nodes. The operations have \textit{amortized} logarithmic
time complexity (i.e. the execution time is on average logarithmic for sequences of operations, but it is not logarithmic for every operation, like in some other dictionaries (which ones?)).

6. Solution not provided.

7. Both arrays can represent heap: see course book p.338, assume that indexing of the array starts with 1.

8. Yes, if the index of the root is 0, then the children of a node with index $i$ have indices $3i + 1, 3i + 2, 3i + 3$.

9. The original heap is to be transformed to a new one with reversed priorities (i.e. the largest number denotes the highest priority). Transform the heap by using a heapification technique similar to that used in Heap Sort algorithm. This can be done in time $\Theta(n)$, the space needed is $\Theta(1)$.

10. Find(4) introduces no changes.

\[
\begin{align*}
\text{Find(2)}: & \quad \text{Merge(A,C):} \\
4 \quad & \quad 4 \\
2 \quad 8 \quad 3 \quad & \quad 2 \quad 8 \quad 3 \quad 1 \\
7 \quad & \quad 7 \\
\text{Find(11)}: & \\
4 \quad & \quad 4 \\
2 \quad 8 \quad 3 \quad 1 \quad 6 \quad 11 \quad & \quad 2 \quad 8 \quad 3 \quad 1 \quad 6 \quad 11 \\
7 \quad & \quad 12 \\
\end{align*}
\]