Exam

Remarks: Choose 5 of the following 6 questions. In all algorithm, always explain how and why they work. ALWAYS, analyze the complexity of your algorithms. In all algorithms, always try to get the fastest possible. A correct algorithm with slow running time may not get full credit. In all data structures, try to minimize as much as possible the running time of any operation. Do not write programs. Only algorithms. I prefer explanations of the algorithms in words; Not pseudo code

1. Question 1:
   (a) Suppose you insert two elements with keys $x$ and $y$ one after the other into an AVL tree. Can the tree have different shape if you first insert $x$ and then $y$, compared to first $y$ and then $x$?
   (b) The same question as above but with delete

2. Question 2: Give an algorithm that based on inorder preorder and postorder, assigns two numbers $n_1(x), n_2(x)$ to all elements $x$ in a binary search tree. Then given a pointer to two nodes $x$ and $y$ it should determine in constant time if $x$ is a descendant of $y$ or $y$ is a descendant of $x$, or none of the above

3. Question 3: For every item here say true or false and explain in a very clear way your answer
   (a) There exists an algorithm that prints the elements of a binary heap, sorted, and runs in $O(n)$ time
   (b) There exists a deterministic hash function that is able to support Insert, Search and Delete in $O(1)$ expected time, regardless of the input
   (c) It is possible to find the $\lceil \log n \rceil$ largest element in an unsorted array in $O(n)$ time.
   (d) It is possible to find the 100 largest element on a heap in $O(1)$ time

4. Question 4:
   (a) You are given two binary trees $T_1, T_2$ both implemented with pointers. A node $x$ in the tree has $key(x)$, the key of $x$, and $left(x)$, pointer to the left subtree and $right(x)$, pointer to the right subtree. Give an algorithm to check if $T_1$ and $T_2$ are the same tree
   (b) Give an algorithm that checks what is the number of nodes in $T_1$ that have exactly one child

5. Question 5: Given a non-sorted array with $n$ elements give an algorithm that finds if there is a value that repeats $n/3$ times or more in $A$
6. **Question 6:** You are given a non-sorted array of $n$ numbers. Let $k < n$. Let $i = \lfloor n/k \rfloor$. Give an algorithm that finds the $k$-smallest, the $2k$-smallest, the $3k$-smallest numbers until the $ik$ smallest number in $A$. 