Theoretical exercise I

Remarks: In all algorithms, always explain how and why they work. ALWAYS, analyze the running time 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.

Always give arguments as for why what you did is correct. You can not just answer ”yes” or ”no”.

NEVER write a program. Only write algorithm in the pseudocode we learned

1. Question 1: You are given a sorted array of integers. The numbers in the array are pairwise distinct, namely no value appears more than once. Your goal is to find if there is an index $1 \leq k \leq n$ so that $A[k] = k$. (Assume that the first index is 1).


3. Question 3: Given an odd size array of pairwise distinct elements, the median in the number that is smaller than $(n - 1)/2$ of the elements and larger than $(n - 1)/2$ elements. Say that we have an algorithm that finds the median in time $O(n)$. Give an algorithm to find the $k$ smallest number in time $O(n)$.

4. Question 4: An algorithm is called stable if elements that are of the same value do not change their order after the sorting.

   (a) Is Merge-sort stable?

   (b) Find a way to make any sorting algorithm stable (you can add additional data to every number).

5. Question 5: Say that we want to maintain both a Stack and a Queue of the same elements. A new element that arrives implies a Push to the stack and a ENQUEUE to the Queue. Any Pop operation should remove the element from the QUEUE as well (even though its not the highest propriety element). Dequeue removes the head of the queue, but also removes this element from the stack (even though its not the last to being inserted). Using two copies of every element and using additional pointers between element, show that we can implement both a stack and a queue so that every operation requires $O(1)$ time.