Problem 1. (20 = 10 + 5 + 5 points). A sorted binary search tree allows to efficiently find, delete and insert items (an item corresponds to a node in the tree, but the node can store more information than just the item.)

Explain (in pseudocode or in English) how to extend this data structure (by putting some additional information into the nodes) so that for any \( k \), one also can efficiently find the \( k \)th smallest item stored in the tree (10 points). Efficient means the time to find the \( k \)th smallest item should be linear in the depth of this item in the tree. Explain how inserting (5 points) and deleting an item (5 points) works for this data structure.

Problem 2. (20 = 10 + 10 points). A stack is a data structure that allows for two operations: push\((a)\) puts an item \(a\) on top of the stack, and pop returns the item on top of the stack and removes it from the stack. E.g. starting with the empty stack, after the sequence of operations

\[
\text{push}(2), \text{push}(4), \text{push}(3), \text{pop}, \text{pop}
\]

we are left with a stack that contains the single item 2, the first pop returns 3, the second pop returns 4.

A queue is a data structure that allows for two operations: enqueue\((a)\) adds an item to the end of the queue, and dequeue returns the item at the front of the queue and removes it from the queue. E.g. starting with the empty queue, after the sequence of operations

\[
\text{enqueue}(2), \text{enqueue}(4), \text{enqueue}(3), \text{dequeue}, \text{dequeue}
\]

we are left with a queue that contains the single item 3, the first dequeue returns 2, the second dequeue returns 4.

Explain (in pseudocode or in English) how to implement a queue using two stacks such that an enqueue and dequeue operation takes constant amortized time (10 points for the construction, 10 points for the analysis.)