Exercise 1 Simple training exercises

1. Insert the keys 'E', 'A', 'S', 'Y', 'Q', 'U', 'E', 'S', 'T', 'I', 'O', 'N' in an initally empty BST T, and show the tree after inserting 'Q' and after inserting 'N'
2. Draw the sequence of BSTs that results when you delete the keys from the tree of the previous exercise in the order that the keys were inserted.
3. When TREE-DELETE calls TRANSPLANT, under what circumstances can the parameter v of TRANSPLANT be NIL? (CLRS 12.3-4)

Exercise 2 Fun creative exercises!

1. Give a recursive version of the TREE-INSERT procedure. (CLRS 12.3-1)
2. You can sort a given set of n numbers by first building a binary search tree containing these numbers (using TREE-INSERT repeatedly to insert the numbers one by one) and then printing the numbers by an inorder tree walk. What are the worst- case and best-case running times for this sorting algorithm? (CLRS 12.3-3)
3. Suppose that instead of each node x keeping the attribute $x.p$, pointing to x's parent, it keeps $x.succ$, pointing to x's successor. Give pseudocode for TREE-SEARCH, TREE-INSERT, and TREE-DELETE on a binary search tree T using this representation. These procedures should operate in $O(h)$ time, where h is the height of the tree T. You may assume that all keys in the binary search tree are distinct. (Hint: You might wish to implement a subroutine that returns the parent of a node) (CLRS 12.3-6)

Exercise 3 Question from the 2024 re-exam

Consider the following algorithm. The input is a *balanced* binary search tree T with n nodes and an array A with n keys, that *may or may not* be in T. The algorithm uses the procedures TREE-SEARCH and TREE-SUCCESSOR from CLRS (Chapter 12.2). Assume zero-indexing.

```
GETSUCCESSORS($T, A$)
 1  $i = 0$
 2  $B$ = a new array of size $A.length$
 3  for $k \in A$
 4      $x$ = TREE-SEARCH($T.root, k$)
 5      if $x \neq$ Nil
 6          $s$ = TREE-SUCCESSOR($x$)
 7          if $s \neq$ Nil
 8              $B[i] = s$
 9              $i = i + 1$
10  return $B$
```

1. Give the asymptotic running time of GETSUCCESSORS *in terms of n.* Argue for your answer, give as tight a bound as possible and remember to reduce your answer (ie. if you analyze the running time of each line, you should still provide an answer like $O(1)$ or $\Theta(2^n)$ for the complete algorithm).
2. The output of GETSUCCESSORS is an array B of size n, but multiple slots in the array may be empty. Can you suggest a better data structure for B, that wouldn't have unused space? Argue for your suggestion.