Problem 1.1  [Constant-Time Concatenate].  Design a data structure $D$ that maintains an ordered set of $n$ keys while supporting the following operations:

(a) $\text{insert}(D, k)$ in $O(\log n)$ time: add key $k$ to $D$’s set.

(b) $\text{delete}(D, k)$ in $O(\log n)$ time: remove key $k$ from $D$’s set.

(c) $\text{predecessor}(D, k)$ in $O(\log n)$ time: find the largest key $< k$ that is in $D$’s set.

(d) $\text{successor}(D, k)$ in $O(\log n)$ time: symmetrically

(e) $\text{concatenate}(D_1, D_2)$ in $O(1)$ time: given two data structures $D_1$ and $D_2$, where all of the keys in $D_1$ are less than all of the keys in $D_2$, destructively combine them into a single data structure holding their combined set of keys.

Each time bound can be amortized. (You should already be comfortable with amortization from a prerequisite class. If not, we recommend that you talk with the course staff for advice.) The variable $n$ represents the number of keys currently in the data structure.

Hint: Start from B-trees.