Homework Assignment #3
Due: September 27, 2024 at 5:00 p.m.

1. It is straightforward to extend the data structure for the BLOCK ADT of Assignment 1 to add support for two more operations:

- DELETE(i, x) removes and returns the element in the ith position of the sequence, and adds x as the new last element of the sequence. This operation changes the sequence from $x_0, x_1, \ldots, x_{K-1}$ to $x_0, x_1, \ldots, x_{i-1}, x_{i+1}, x_{i+2}, \ldots, x_{K-1}, x$.
- SHIFTLEFT(x) is equivalent to DELETE$(0, x)$.

So that DELETE runs in $O(K)$ time and SHIFTLEFT runs in $O(1)$ time. For this question, you can assume that you have this enhanced BLOCK data structure.

Now, we wish to implement an enhanced SEQUENCE, which stores a (variable-length) sequence $x_0, x_1, \ldots, x_{n-1}$ and supports the following operations.

- INDEX(i) returns x_i.
- INSERT(i, x) inserts element x in the ith position of the sequence, so the sequence changes from $x_0, x_1 \ldots, x_{n-1}$ to $x_0, x_1, \ldots, x_{i-1}, x, x_i, x_{i+1}, \ldots, x_{n-1}$. (If $i = n$, then x is appended to the end of the sequence.)
- DELETE(i) deletes the element in the ith position of the sequence, so the sequnce changes from $x_0, x_1 \ldots, x_{n-1}$ to $x_0, x_1, \ldots, x_{i-1}, x_{i+1}, x_{i+2}, \ldots, x_{n-1}$.

Our goal is to have every individual INDEX operation take constant time, while each update take $O(\sqrt{n})$ steps in an amortized sense. More precisely, assuming the SEQUENCE is initially empty, the total time for every sequence of m INSERT and DELETE operations should be $O(m\sqrt{n})$, where n is the maximum number of elements in the SEQUENCE at any time.

Moreover, your data structure should use $O(n)$ words of memory. (Assume a word is big enough to store a pointer or an element x_i.)

[3] **(a)** Give a high-level overview of how your data structure works.

[1] **(b)** Draw a picture of your data structure after 16 elements have been inserted into it.

[5] **(c)** Give a detailed implementation, describing the layout of information in memory and pseudocode for the three operations. Whenever you use a BLOCK, specify the K value used for it.

[3] **(d)** Briefly explain why your data structure meets the required time and space bounds.

Remark: It is possible to implement this data structure so that each INSERT and DELETE takes $O(\sqrt{n})$ time. But you only need to provide an implementation that takes *amortized* $O(\sqrt{n})$ time per INSERT or DELETE.