Data Structures and Algorithms in Java™

Sixth Edition

Michael T. Goodrich
Department of Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

Michael H. Goldwasser
Department of Mathematics and Computer Science
Saint Louis University

Study Guide: Hints to Exercises

WILEY

Chapter 6 Stacks, Queues, and Deques

Hints

Reinforcement

R-6.1) If a stack is empty when pop is called, its size does not change.

R-6.2) It is one less than the size of S.

R-6.3) Use a paper and pencil with eraser to simulate the stack.

R-6.4) Transfer items one at a time.

R-6.5) First check if the stack is already empty.

R-6.6) Give a recursive definition.

R-6.7) If a queue is empty when dequeue is called, its size does not change.

R-6.8) Each successful dequeue operation causes that index to shift circularly to the right.

R-6.9) Use a paper and pencil with eraser to simulate the queue.

R-6.10) Use operations at the appropriate ends of the deque.

R-6.11) Use operations at the appropriate ends of the deque.

R-6.12) Use a paper and pencil to simulate the deque.

R-6.13) Use the results of removal methods as arguments to insertion methods.

R-6.14) Use the results of removal methods as arguments to insertion methods. In addition, you will need to use more of the stack for temporary storage.

R-6.15) You might start by concatenating the bodies of the dequeue and enqueue methods and then look to avoid redundancy.

Creativity

C-6.16) Pop the top integer, but remember it.

C-6.17) You will need to do three transfers.

C-6.18) After finding what's between the $<$ and $>$ characters, the tag is only the part before the first space (if any).

C-6.19) Use a stack.

C-6.20) You can still use R as temporary storage, as long as you never pop its original contents.

C-6.21) Use a stack to reduce the problem to that of enumerating all permutations of the numbers $\{1,2,\ldots,n-1\}$.

C-6.22) Think of the stacks like jugs and the dump operations like water being poured between two jugs.

C-6.23) Use the stack to store the elements yet to be used to generate subsets and use the queue to store the subsets generated so far.

C-6.24) Think of how you might use Q to process the elements of S twice.

C-6.25) Rotate elements within the queue.

C-6.26) You can try it out

C-6.27) See Section 3.6 for a discussion of cloning data structures.

C-6.28) See Section 3.6 for a discussion of cloning data structures.

C-6.29) You are welcome to modify the SinglyLinkedList class to add necessary support

C-6.30) Use separate indices for the two ends.

C-6.31) Think of using one stack for each end of the deque.

C-6.32) Use the deque like a stack.

C-6.33) Think of the queues like boxes and the integers like red and blue marbles.

C-6.34) Lazy and Crazy should only go across once.

Projects

P-6.35) You will need to use a stack.

P-6.36) Define two stacks that are used match sell orders with buy orders.

P-6.37) Start one stack at each end of the array, growing toward the center.

P-6.38) How does this functionality compare to a deque?

P-6.39) Think carefully about the orientation of the linked list.

P-6.40) The section on the Deque ADT gives advice on using a circular array implementation.