COP3410
Assignment 6

Author: Jessica W.

Instructor: Dr. Taebi

Date: April 2, 2022

Chapter 6: ADT - Stacks and Queues

Problem R-6.1
What values are returned during the following series of stack operations, if executed on an initially empty stack?
push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop()

Solution:

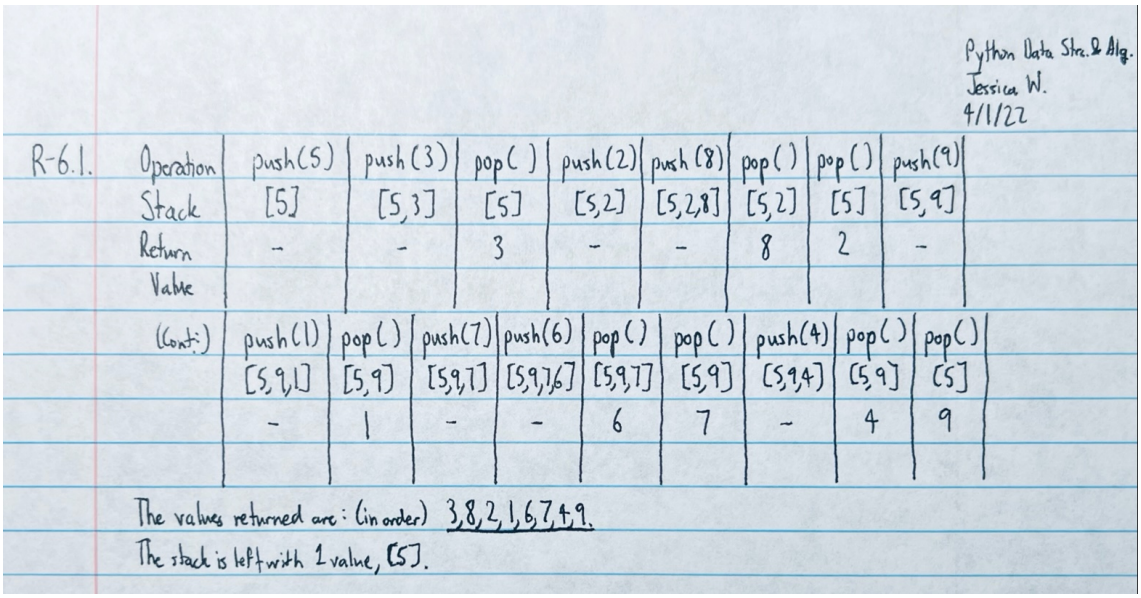

Figure 1: Stack S

Each push() operation adds one value to the stack at the back/top, while each pop() operation removes the last inserted element.
Thus, tracing the operations as in Figure 1, we find the values returned are: **3, 8, 2, 1, 6, 7, 4, 9**. ■

Problem R-6.2
Suppose an initially empty stack S has executed a total of 25 push operations, 12 top operations, and 10 pop operations, 3 of which raised Empty errors that were caught and ignored. What is the current size of S?

Solution:

The current size of S is **18**.
Note that a top operation simply returns the value of the top element, without changing the size of the stack.
Also, a push operation will always increase the size by 1.
A pop operation will decrease the size of the stack by 1, unless the stack is empty and thus raises an error.

So, in total, the stack's size has incremented 25 times and decremented $10 - 3 = 7$ times, for a final size of $0 + 25 - 7 = 18$ elements. ■

Problem R-6.3
Implement a function with signature transfer(S, T) that transfers all elements from stack S onto stack T, so that the element that starts at the top of S is the first to be inserted onto T, and the element at the bottom of S ends up at the top of T.

Solution:

```
73  def transfer(S, T):
74      # Transfer all elements of stack S onto stack T, in reverse order.
75      try:
76          while not S.is_empty():
77              T.push(S.pop())
78      except TypeError:
79          print("S and T must be stacks.")
80
```

Figure 2: Function transfer(S, T)

The function **transfer(S, T)** will attempt to push each element of S onto T, in order top to bottom, until S is empty. In this way, the top element of S is inserted onto T first, and the bottom element of S becomes the new top element of T.
(Note the function will raise a TypeError if the inputs, S or T, are not stacks and do not have methods push/pop defined.)

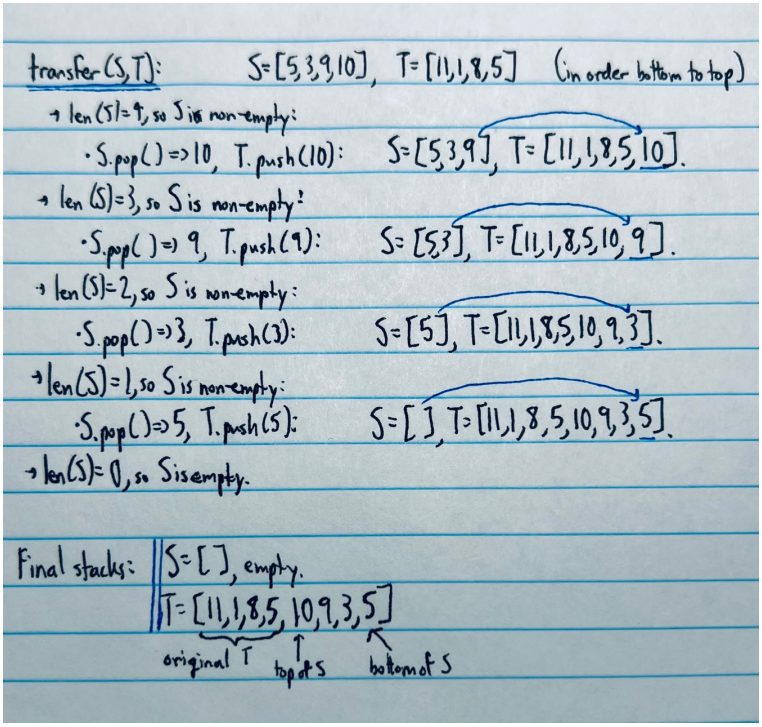

Figure 3: Example of transfer(S, T)

■

Problem R-6.7
What values are returned during the following sequence of queue operations, if executed on an initially empty queue? enqueue(5), enqueue(3), dequeue(), enqueue(2), enqueue(8), dequeue(), dequeue(), enqueue(9), enqueue(1), dequeue(), enqueue(7), enqueue(6), dequeue(), dequeue(), enqueue(4), dequeue(), dequeue().

Solution:

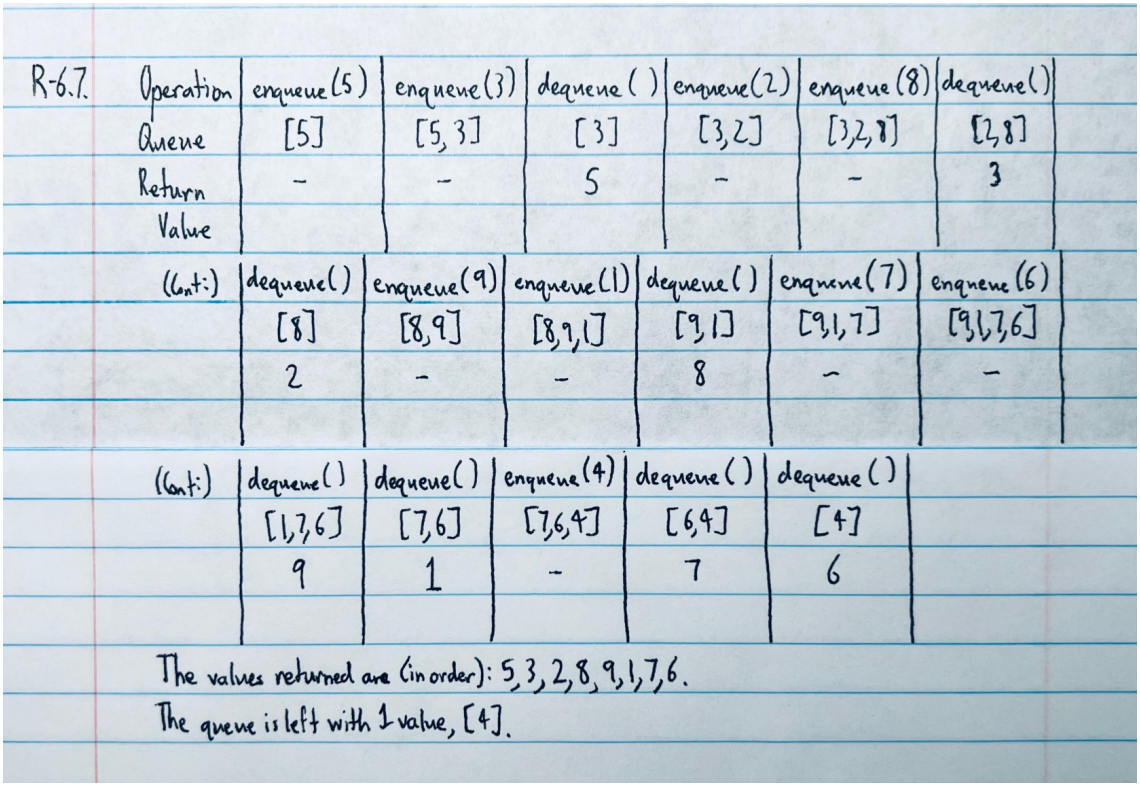

Figure 4: Queue Q

Each push() operation adds one value to the stack at the back, while each pop() operation removes the frontmost element.
Thus, tracing the operations as in the figure, we remove elements in the order they were inserted and find the values returned are: **5, 3, 2, 8, 9, 1, 7, 6**. ■

Problem R-6.8
Suppose an initially empty queue Q has executed a total of 32 enqueue operations, 10 first operations, and 15 dequeue operations, 5 of which raised Empty errors that were caught and ignored. What is the current size of Q?

Solution:

The current size of Q is $\boxed{\mathbf{22}}$.
Note that a first operation simply returns the value of the first element, without changing the size of the queue.
Also, an enqueue operation will always increase the size by 1.
A dequeue operation will decrease the size of the queue by 1 unless the queue is empty and thus raises an error.

So, in total, the queue's size has incremented 32 times and decremented $15 - 5 = 10$ times, for a final size of $0 + 32 - 10 = 22$ elements.

Problem R-6.9
Had the queue of the previous problem been an instance of ArrayQueue that used an initial array of capacity 30, and had its size never been greater than 30, what would be the final value of the front instance variable?

Solution:

The final value of the front instance variable would be $\boxed{\mathbf{10}}$, assuming the initial value is index 0.
Note that an array-based queue "cycles" the front index, with each successful dequeue operation incrementing *front* by 1 modulo 30.
*A dequeue operation that returns an Error will not change front.
Neither an enqueue operation nor a first operation will change the value of the front instance variable, so long as the size of the ArrayQueue stays within 30.

Thus, as we have 10 successful dequeue operations, the value of front will be $0 + 10 = 10$.

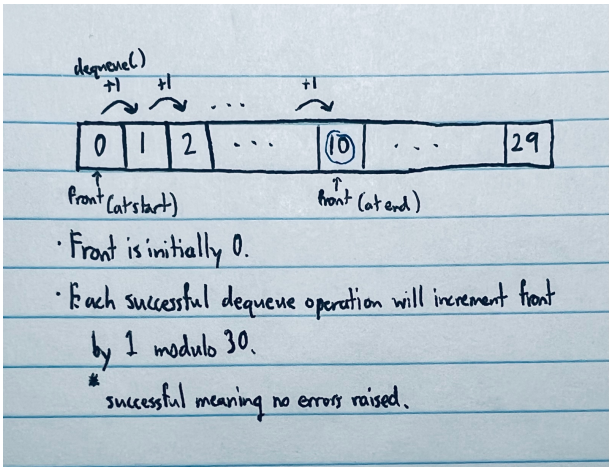

Figure 5: Visualization with an ArrayQueue