Sarah Sindeband
COP 3410
Assignment 6
R6.1, R6.2, R6.3, R6.7, R6.8, R6.9
Use pen and paper to draw out the stack and que for every problem and their changes.

R6.1 What values are returned during the following series of stack operations, if executed upon an initially empty stack?

push(5) |5|

push(3) |5|3|

pop() |5|~~3~~| → |5|

push(2) |5|2|

push(8) |5|2|8|

pop() |5|2|~~8~~| → |5|2|

pop() |5|~~2~~| → |5|

push(9) |5|9|

push(1) |5|9|1|

pop() |5|9|~~1~~| → |5|9|

push(7) |5|9|7|

push(6) |5|9|7|6|

pop() |5|9|7|~~6~~| → |5|9|7|

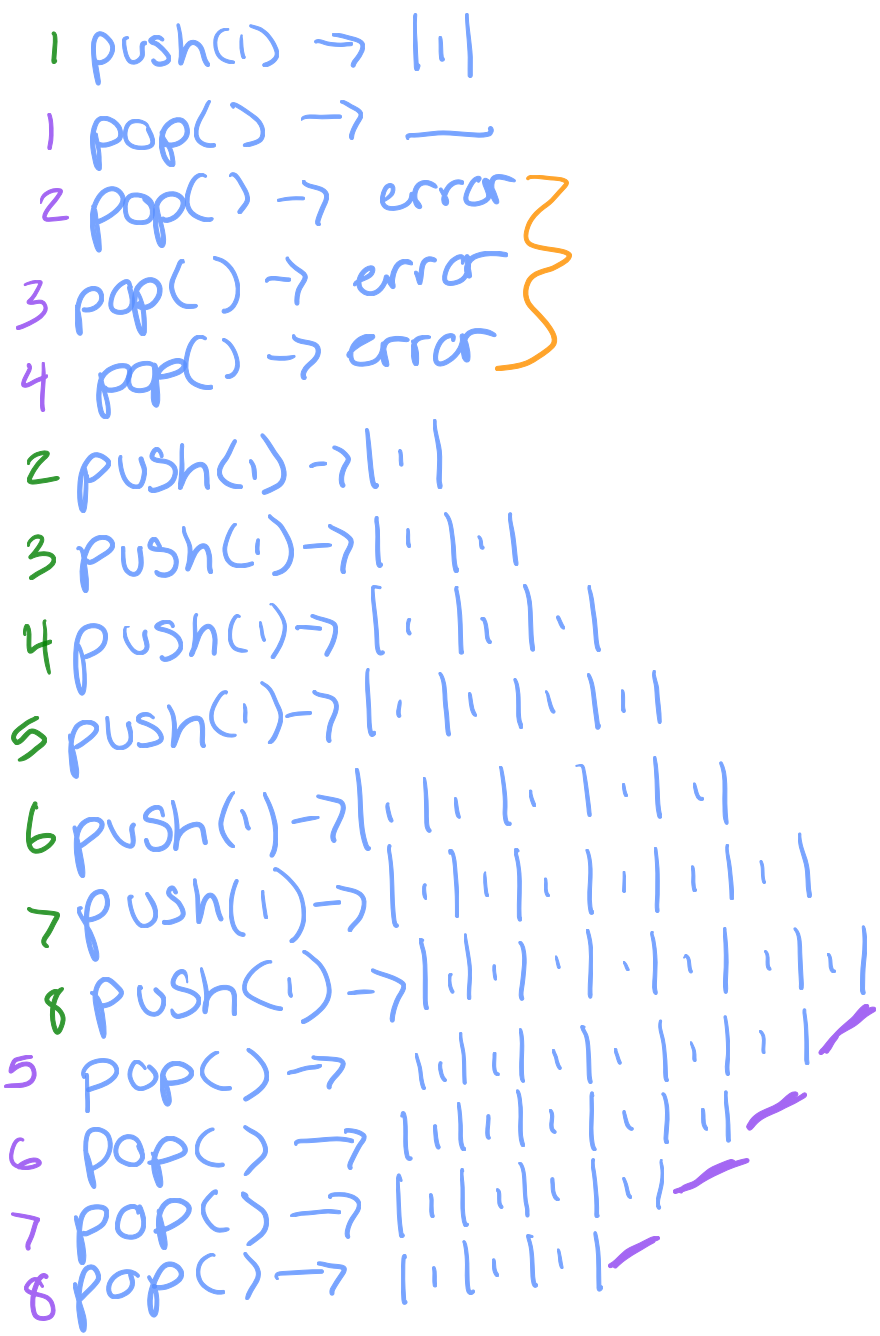
pop() |5|9|~~7~~| → |5|9|

push(4) |5|9|4|

pop() |5|9|~~4~~| → |5|9|

pop() |5|~~9~~| → |5|

R6.2 Suppose an initially empty stack S has been 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?



9 pop() → | 1 | 1 |

10 pop() → | 1 |

9 push(0) → | 1 | 0 |

1 top() → 0

10 push(2) → | 1 | 0 | 2 |

2 top() → 2

11 push(3) → | 1 | 0 | 2 | 3 |

3 top() → 3

12 push(4) → | 1 | 0 | 2 | 3 | 4 |

4 top() → 4

13 push(5) → | 1 | 0 | 2 | 3 | 4 | 5 |

5 top() → 5

14 push(6) → | 1 | 0 | 2 | 3 | 4 | 5 | 6 |

6 top() → 6

15 push(7) → | 1 | 0 | 2 | 3 | 4 | 5 | 6 | 7 |

7 top() → 7

16 push(6) → | 1 | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 6 |

8 top → 6

17 push(5) → | 1 | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 6 | 5 |

9 top → 5

18 push(4) -> |1|0|2|3|4|5|6|7|6|5|4|

10 top() -> 4

19 push(3) -> |1|0|2|3|4|5|6|7|6|5|4|3|

11 top() -> 3

20 push(2) -> |1|0|2|3|4|5|6|7|6|5|4|3|2|

12 top() -> 2

21 push(1) -> |1|0|2|3|4|5|6|7|6|5|4|3|2|1|

22 push(0) -> |1|0|2|3|4|5|6|7|6|5|4|3|2|1|0|

23 push(1) -> |1|0|2|3|4|5|6|7|6|5|4|3|2|1|0|1|

24 push(2) -> |1|0|2|3|4|5|6|7|6|5|4|3|2|1|0|1|2|

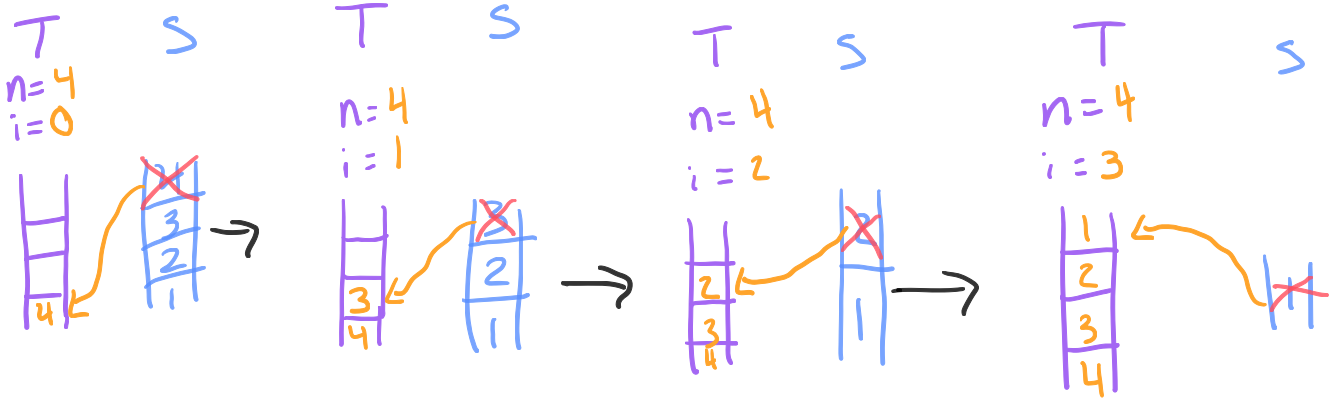
25 push(3) -> |1|0|2|3|4|5|6|7|6|5|4|3|2|1|0|1|2|3|

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

len(s) -> 18

R6.3 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.

```
class PracticeStack:
    def __init__(self):
        self._stack = [ ]

    def __len__ (self):
        return len(self._stack)

    def is_empty(self):
        return len(self._stack) == 0

    def push(self, e):
        self._stack.append(e)

    def top(self):
        return self._stack[-1]

    def pop(self):
        return self._stack.pop()

    def __str__(self):
        return str(self._stack)

def transfer(S, T):
    n = len(S)
    i = 0
    while i < n:
        T.push(S.top())
        S.pop()
        i+=1
    return T

if __name__ == "__main__":
    S = PracticeStack()
    T = PracticeStack()
    S.push(1)
    S.push(2)
    S.push(3)
    S.push(4)
    print(S)
    transfer(S, T)
    print(T)
```



R6.7 What values are returned during the following sequence of queue operations, if executed initially on an empty queue?

enqueue(5) [5]

enqueue(3) [5,3]

dequeue() [3]

enqueue(2) [3,2]

enqueue(8) [3,2,8]

dequeue() [2,8]

dequeue() [8]

enqueue(9) [8,9]

enqueue(1) [8,9,1]

dequeue() [9,1]

enqueue(7) [9,1,7]

enqueue(6) [9,1,7,6]

dequeue() [1,7,6]

dequeue() [7,6]

enqueue(4) [7,6,4]

dequeue() [6,4]

dequeue() [4]

R6.8 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?

Assuming this is based on the ArrayQueue from class/textbook:

Q.enqueue(1) [1, None, None, None, None, None, None, None, None, None]
Q.enqueue(1) [1,1,None, None, None, None, None, None, None, None]
Q.enqueue(1) [1,1,1, None, None, None, None, None, None, None]
Q.enqueue(1) [1,1,1,1, None, None, None, None, None, None]
Q.enqueue(1) [1,1,1,1,1,None, None, None, None, None]

Q.dequeue() [None, 1, 1, 1, 1, None, None, None, None, None]
Q.dequeue() [None, None, 1,1,1, None, None, None, None, None]
Q.dequeue() [None, None, None, 1,1, None, None, None, None, None]
Q.dequeue() [None, None, None, None, 1, None, None, None, None, None]
Q.dequeue() [None, None, None, None, None, None, None, None, None, None]

Q.dequeue() Empty error
Q.dequeue() Empty error
Q.dequeue() Empty error
Q.dequeue() Empty error
Q.dequeue() Empty error

Q.enqueue(1) [None, None, None, None, None, 1, None, None, None, None]
Q.enqueue(1) [None, None, None, None, None, 1, 1, None, None, None]
Q.enqueue(1) [None, None, None, None, None, 1, 1, 1, None, None]
Q.enqueue(1) [None, None, None, None, None, 1, 1, 1, 1, None]
Q.enqueue(1) [None, None, None, None, None, 1, 1, 1, 1, 1]

Q.enqueue(1) [1, None, None, None, None, 1, 1, 1, 1, 1]
Q.enqueue(1) [1, 1, None, None, None, 1, 1, 1, 1, 1]
Q.enqueue(1) [1, 1, 1, None, None, 1, 1, 1, 1, 1]
Q.enqueue(1) [1, 1, 1, 1, None, 1, 1, 1, 1, 1]
Q.enqueue(1) [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Q.first() 1
Q.first() 1
Q.first() 1
Q.first() 1
Q.first() 1

Continued:

Q.first() 1
Q.first() 1
Q.first() 1
Q.first() 1
Q.first() 1

Q.dequeue() [1, 1, 1, 1, 1, None, 1, 1, 1, 1]
Q.dequeue() [1, 1, 1, 1, 1, None, None, 1, 1, 1]
Q.dequeue() [1, 1, 1, 1, 1, None, None, None, 1, 1]
Q.dequeue() [1, 1, 1, 1, 1, None, None, None, None, 1]
Q.dequeue() [1, 1, 1, 1, 1, None, None, None, None, None]

Q.enqueue(2) [1, 1, 1, 1, 1, 2, None, None, None, None]
Q.enqueue(3) [1, 1, 1, 1, 1, 2, 3, None, None, None]
Q.enqueue(4) [1, 1, 1, 1, 1, 2, 3, 4, None, None]
Q.enqueue(5) [1, 1, 1, 1, 1, 2, 3, 4, 5, None]
Q.enqueue(6) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6]

Q.enqueue(1) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, None, None, None, None, None, None, None, None, None]
Q.enqueue(1) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, None, None, None, None, None, None, None, None]
Q.enqueue(1) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, None, None, None, None, None, None, None]
Q.enqueue(1) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, None, None, None, None, None, None]
Q.enqueue(1) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, None, None, None, None, None]

Q.enqueue(2) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, None, None, None, None]
Q.enqueue(2) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, 2, None, None, None]
Q.enqueue(2) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, 2, 2, None, None]
Q.enqueue(2) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, 2, 2, 2, None]
Q.enqueue(2) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

Q.enqueue(1) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
Q.enqueue(2) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 2, None]

len(Q) 22

R6.9 Had the queue of the previous problem been an instance of ArrayQueue that used an initial array capacity of 30, had its size never been greater than 30, what would be the final value of the _front instance variable?

print(Q) [1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

print(Q.front) 0

print(Q.first()) 1