1. [5, 16] ← top to bottom
2. [] empty queue
3.
    - Create two index variables being left = n-1 where n is the number of elements in the original deque and right = 1
    - At each iteration check if popFront() equals target or popRear() equals target which removes and returns the element at the front and the back of the deque respectively
    - If popFront() equals target, return right. If popRear() equals target return left
    - If neither conditions are satisfied, increment right by one and decrement left by one and iterate again

7. #4 time complexity - O(N) where N is the length of the given string. Iterates through each char in the string one time, space complexity - O(1) the auxiliary space created does not depend on the length of the given string
#5 time complexity - O(Nk) where N is the length of the given string and k is the integer value for the amount of times the characters are repeated. Iterates through each character some k times
Space complexity - O(N) because a new string builder is created each iteration
#6 time complexity - O(N^2) where N is the length of the given string. Iterate through each character in the string and at each string it can append all elements in the intermediate stack into the result stack. the intermediate stack can have length up to N.
Space complexity - O(N) the amount of elements in the made linkedlists depend linearly on the length of the given string