1) Stack (LIFO = Last in first out)

1) push(8)

2) push(2)

Add 2 to stack

3) pop() → removing 2

4) push(pop()*2)

remove 8 and multiply it by 2 and then push the result

5) push(10)

6) pusp(pop()/2)

remove 10 and divide by 2 and then push the result

2) Queue (FIFO = First in, First out)

1) push(4)

enqueue → [4] → dequeue

2) push(pop()+4)

removes 4 and adds it by 4 and then push the result which is 8

[8]

3) push(8)

[8, 8]

4) push(pop()/2)

Remove front and divide by 2 and then push the result to back which is 4

[8, 8]

÷2

push

[8, 4]

5) pop() remove 8

[4]

6) pop() remove 4

[] empty

3) Find in deque

Initialize 2 pointers
Left: start from front
Right: start from back n-1

- If the element at left is equal to X return left. If the element is equal to right return n-1-right
- Using 2 pointers left and right to traverse the deque from both ends comparison is at most $n/2$ which is still O(n)
- If the element is found at the left pointer, its position is simply left
- If the element is found at the right pointer its pos is called n-1-right
  n-1-i
- If deque is not found or element is not found return -1

Example 10, 20, 30, 40, 50 X=30
L = 0
R = 4 First Iteration
check L=10 No == 30
check R=50 No == 30
move L to 1 R to 3

2nd Iteration
check L=20 No == 30
check R=40 No == 30
move L to 2 R to 3

3rd Iteration
check L=30 Yes == 30
return L=2

TIME O(n)
- Each element in deque at most
- Worst case it performs n/2 comparisons

SPACE O(N)
deque is converted into array for indexing which O(n) for additional space L and R constant

7) 4) To determine whether the brackets in the input text are balanced, the algorithm uses a stack. It pushes opening brackets onto the stack and pops them when a matching closing bracket is discovered during its single iteration through the string

TIME:

- Every character in the string is processed exactly once by the algorithm It carries out constant-time actions, like pushing or popping every character.
- As a result O(n), where n is the length of the input string, represents time complexity

SPACE:

- Worst case if the string has no closing brackets like ((((( )))) the stack will hold all of the opening brackets.
- The length of the input string determines the stacks minimum size, n.

O(n) space complexity

5) To decode the input string, the method makes uses two stacks, one for strings and other for counts. It process characters opening and closing brackets and numbers inn single iteration over the string

TIME

- Every character in the string gets processed only once by the algorithm
- Constant time moves such as adding to StringBuilder and pushing to or popping for every character.

SO O(n), where n is the length of the input string, represents time complexity

SPACE

- Initial findings are stored in the StringBuilder and stacks
- In worst StringBuilder and stacks can hold O(n) elements

O(n) space

6) Algorithm transforms an infix phase into postfix phase using stack. It uses operands, operators, during a single iteration of the input text.

TIME

- Every character Once using constant time O(n)

SPACE

- Stack stores operators and parenthesis and can store up to O(n) elements