1.

- push(8)
    - Stack: [8]
- push(2)
    - Stack: [8, 2]
- pop()
    - Removes 2
    - Stack: [8]
- push(pop() * 2)
    - pop() removes 8
    - 8 * 2 = 16
    - Stack: [16]
- push(10)
    - Stack: [16, 10]
- push(pop() / 2)
    - pop() removes 10
    - 10 / 2 = 5
    - Stack: [16, 5]

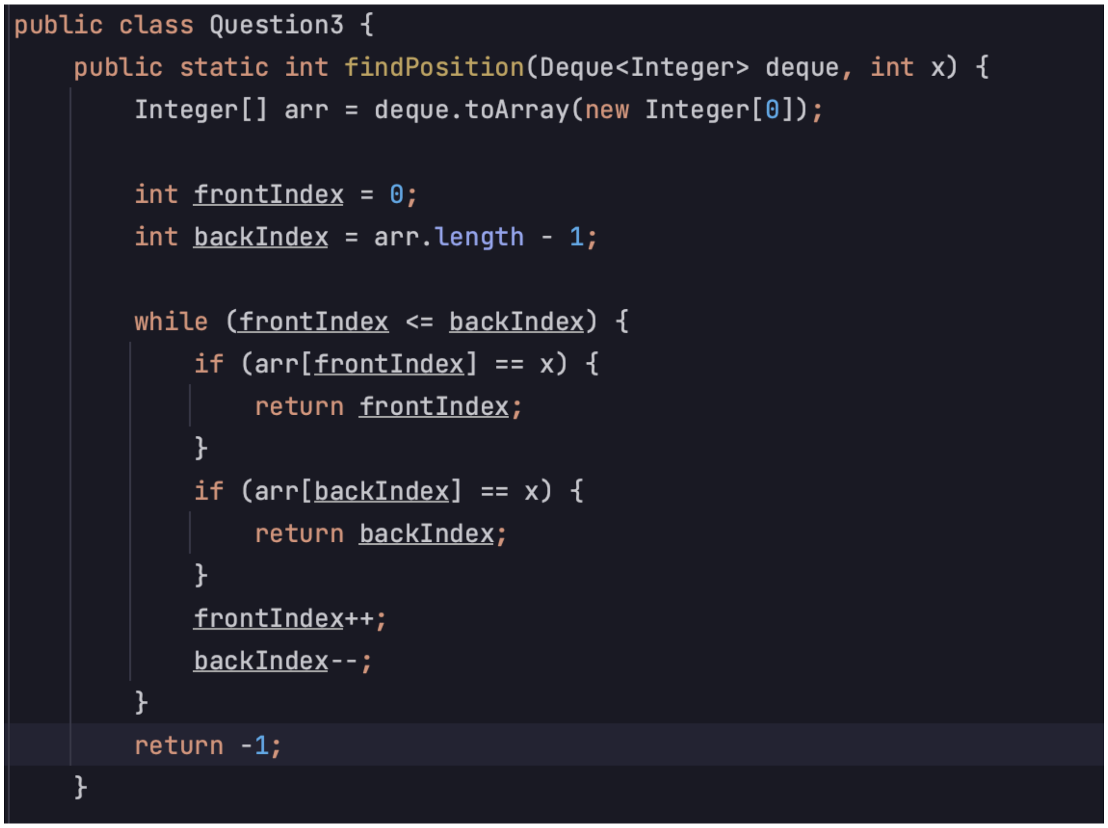
Final: [16, 5]

2.

- push(4)
    - Stack: [4]
- push(pop() + 4)
    - pop() removes 4
    - 4 + 4 = 8
    - push(8)
    - Stack: [8]
- push(8)
    - Stack: [8, 8]
- push(pop() / 2)
    - pop() removes 8
    - 8 / 2 = 4
    - push(4)
    - Stack: [8, 4]
- pop()
    - Removes 4
    - Stack: [8]
- pop()
    - Removes 8
    - Stack: [] (empty)

Final: [ ] (empty)

3.



```java
public class Question3 {
    public static int findPosition(Deque<Integer> deque, int x) {
        Integer[] arr = deque.toArray(new Integer[0]);

        int frontIndex = 0;
        int backIndex = arr.length - 1;

        while (frontIndex <= backIndex) {
            if (arr[frontIndex] == x) {
                return frontIndex;
            }
            if (arr[backIndex] == x) {
                return backIndex;
            }
            frontIndex++;
            backIndex--;
        }
        return -1;
    }
```

7.

Problem 4 Balanced Brackets

- Time complexity: O(n)
    - We iterate through the string once where n is the length of the string. The stack operations like push and pop only take a constant time; we use these operations on each character of the string, and it has linear tme complexity.
- Space complexity: O(n)
    - The space complexity is due to the stack, since in the worst case scenario all characters in the string could be opening brackets so we will store them in the stack.

Problem 5: Decode String

- Time complexity: O(n)
    - We iterate through the string exactly once where n is the length of the string. In the nested structures, we do not reprocess any character since each character is only processed once.

Problem 6 Infix to Postfix Conversion

- Time complexity: O(n)
    - Each character is processed exactly once, where n is the length of the input string. The precedence checks are implemented in constant time using HashMap.
- Space Complexity: O(n)
    - The space complexity is from the stack used to store operators and parentheses. Worst case scenario, the stack holds all operators in the expression.