

232. Implement Queue using Stacks

```
class MyStack {  
private:  
    std::queue<int> q1, q2;  
public:  
    MyStack() {}  
    void push(int x) {  
        q2.push(x);  
        while (!q1.empty()) {  
            q2.push(q1.front());  
            q1.pop();  
        }  
        std::swap(q1, q2);  
    }  
    int pop() {  
        int topElement = q1.front();  
        q1.pop();  
        return topElement;  
    }  
    int top() {  
        return q1.front();  
    }  
    bool empty() {  
        return q1.empty();  
    }  
};
```

The screenshot displays the LeetCode interface for problem 232. The left panel shows the submission status: "Accepted" with 18/18 testcases passed, submitted by user "ashishkumar0315" on Mar 12, 2025. Performance metrics show 0 ms runtime (100.00% beats) and 9.50 MB memory (53.12% beats). A bar chart shows the runtime distribution. The right panel shows the C++ code and the test result for "Case 1". The input sequence is ["MyStack", "push", "push", "top", "pop", "empty"] and the output sequence is [[], [1], [2], [], [], []].

```
class MyStack {  
private:  
    std::queue<int> q1, q2;  
public:  
    MyStack() {}  
    void push(int x) {  
        q2.push(x);  
        while (!q1.empty()) {  
            q2.push(q1.front());  
            q1.pop();  
        }  
        std::swap(q1, q2);  
    }  
    int pop() {  
        int topElement = q1.front();  
        q1.pop();  
        return topElement;  
    }  
    int top() {  
        return q1.front();  
    }  
    bool empty() {  
        return q1.empty();  
    }  
};
```

Testcase 1: Runtime: 0 ms

Input: ["MyStack", "push", "push", "top", "pop", "empty"]

Output: [[], [1], [2], [], [], []]

225. Implement Stack using Queues:

```
class MyStack {
private:
    std::queue<int> q1, q2;

public:
    MyStack() {}

    void push(int x) {
        q2.push(x);
        while (!q1.empty()) {
            q2.push(q1.front());
            q1.pop();
        }
        std::swap(q1, q2);
    }

    int pop() {
        int topElement = q1.front();
        q1.pop();
        return topElement;
    }

    int top() {
        return q1.front();
    }

    bool empty() {
        return q1.empty();
    }
};
```

The screenshot displays a code editor with the C++ implementation of a stack using two queues. The code is as follows:

```
3  std::queue<int> q1, q2;
4
5  public:
6      MyStack() {}
7
8      void push(int x) {
9          q2.push(x);
10         while (!q1.empty()) {
11             q2.push(q1.front());
12             q1.pop();
13         }
14         std::swap(q1, q2);
15     }
```

Below the code, the performance metrics are shown:

- Runtime:** 0 ms | Beats 100.00%
- Memory:** 9.50 MB | Beats 53.12%

A bar chart indicates that 1.17% of solutions used 2 ms of runtime.

The test result section shows the following input and output:

Input: ["MyStack", "push", "push", "top", "pop", "empty"]

Output: [[], [1], [2], [], [], []]

