Description | Editorial | Solutions | Submissions

# 232. Implement Queue using Stacks

Solved

Easy | Topics | Companies

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (`push`, `peek`, `pop`, and `empty`).

Implement the `MyQueue` class:

- `void push(int x)` Pushes element x to the back of the queue.
- `int pop()` Removes the element from the front of the queue and returns it.
- `int peek()` Returns the element at the front of the queue.
- `boolean empty()` Returns `true` if the queue is empty, `false` otherwise.

**Notes:**

- You must use **only** standard operations of a stack, which means only `push to top`, `peek/pop from top`, `size`, and `is empty` operations are valid.
- Depending on your language, the stack may not be supported natively.

8K | 124 | 61 Online

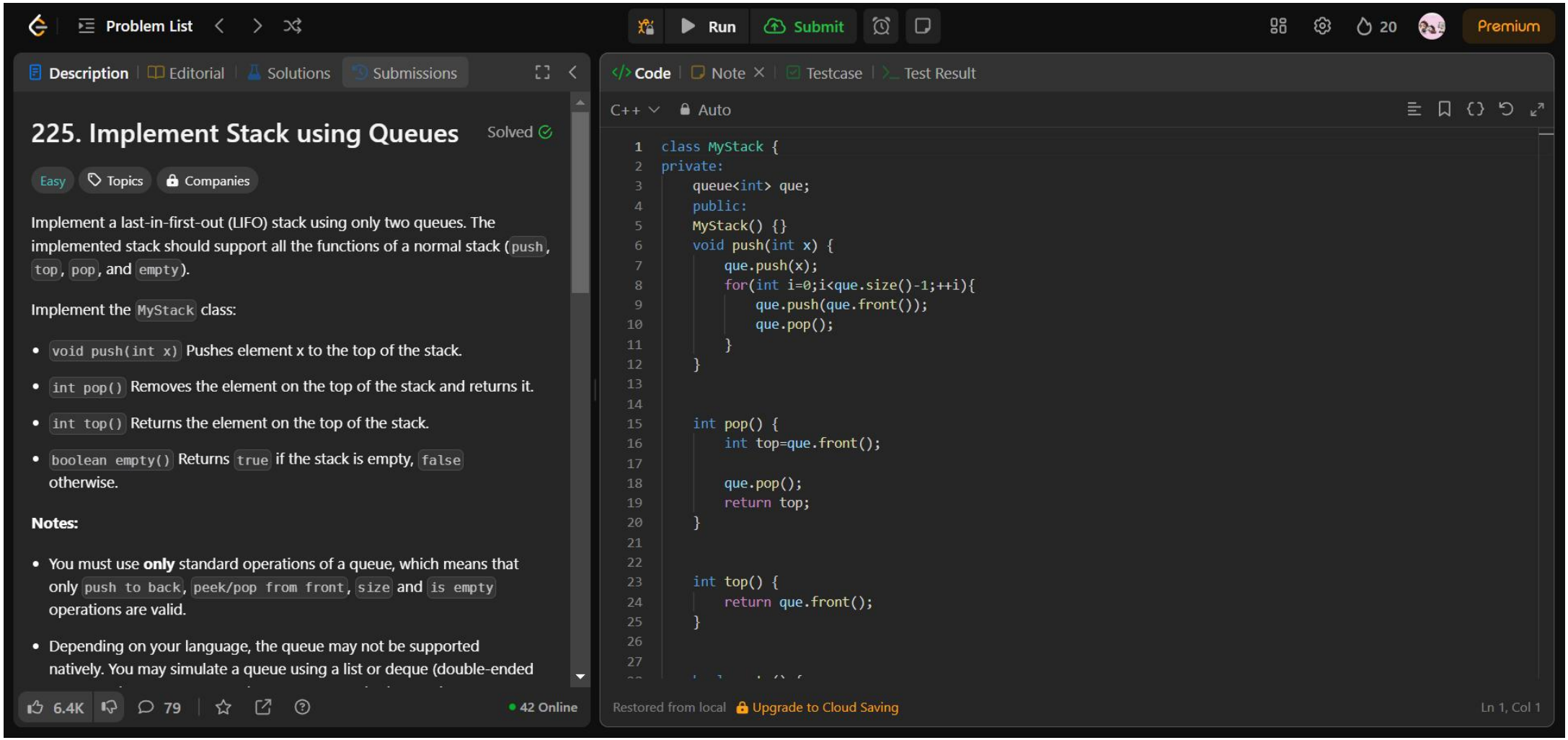
Code | Note | Testcase | Test Result

C++ | Auto

```cpp
class MyQueue {
private:
    stack<int> input;
    stack<int> output;

public:
    MyQueue() {}

    void push(int x) {
        input.push(x);
    }

    int pop() {
        peek();
        int val = output.top();
        output.pop();
        return val;
    }

    int peek() {
        if (output.empty()) {
            while (!input.empty()) {
                output.push(input.top());
                input.pop();
            }
        }
        return output.top();
```

Restored from local | Upgrade to Cloud Saving

Ln 1, C


Problem List
Run
Submit
20
Premium
Description
Editorial
Solutions
Submissions
225. Implement Stack using Queues
Solved
Easy
Topics
Companies
Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).
Implement the MyStack class:
void push(int x) Pushes element x to the top of the stack.
int pop() Removes the element on the top of the stack and returns it.
int top() Returns the element on the top of the stack.
boolean empty() Returns true if the stack is empty, false otherwise.
Notes:
You must use only standard operations of a queue, which means that only push to back, peek/pop from front, size and is empty operations are valid.
Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended
6.4K
79
42 Online
Code
Note
Testcase
Test Result
C++
Auto
class MyStack {
private:
queue<int> que;
public:
MyStack() {}
void push(int x) {
que.push(x);
for(int i=0;i<que.size()-1;++i){
que.push(que.front());
que.pop();
}
}
int pop() {
int top=que.front();
que.pop();
return top;
}
int top() {
return que.front();
}
Restored from local
Upgrade to Cloud Saving
Ln 1, Col 1

Problem List | Run | Submit | 20

Description | Accepted | Editorial | Solutions

# 155. Min Stack

Solved

Medium | Topics | Companies | Hint

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the `MinStack` class:

- `MinStack()` initializes the stack object.
- `void push(int val)` pushes the element `val` onto the stack.
- `void pop()` removes the element on the top of the stack.
- `int top()` gets the top element of the stack.
- `int getMin()` retrieves the minimum element in the stack.

You must implement a solution with `O(1)` time complexity for each function.

**Example 1:**

**Input**

```
["MinStack","push","push","push","getMin","pop","top",
"getMin"]
```

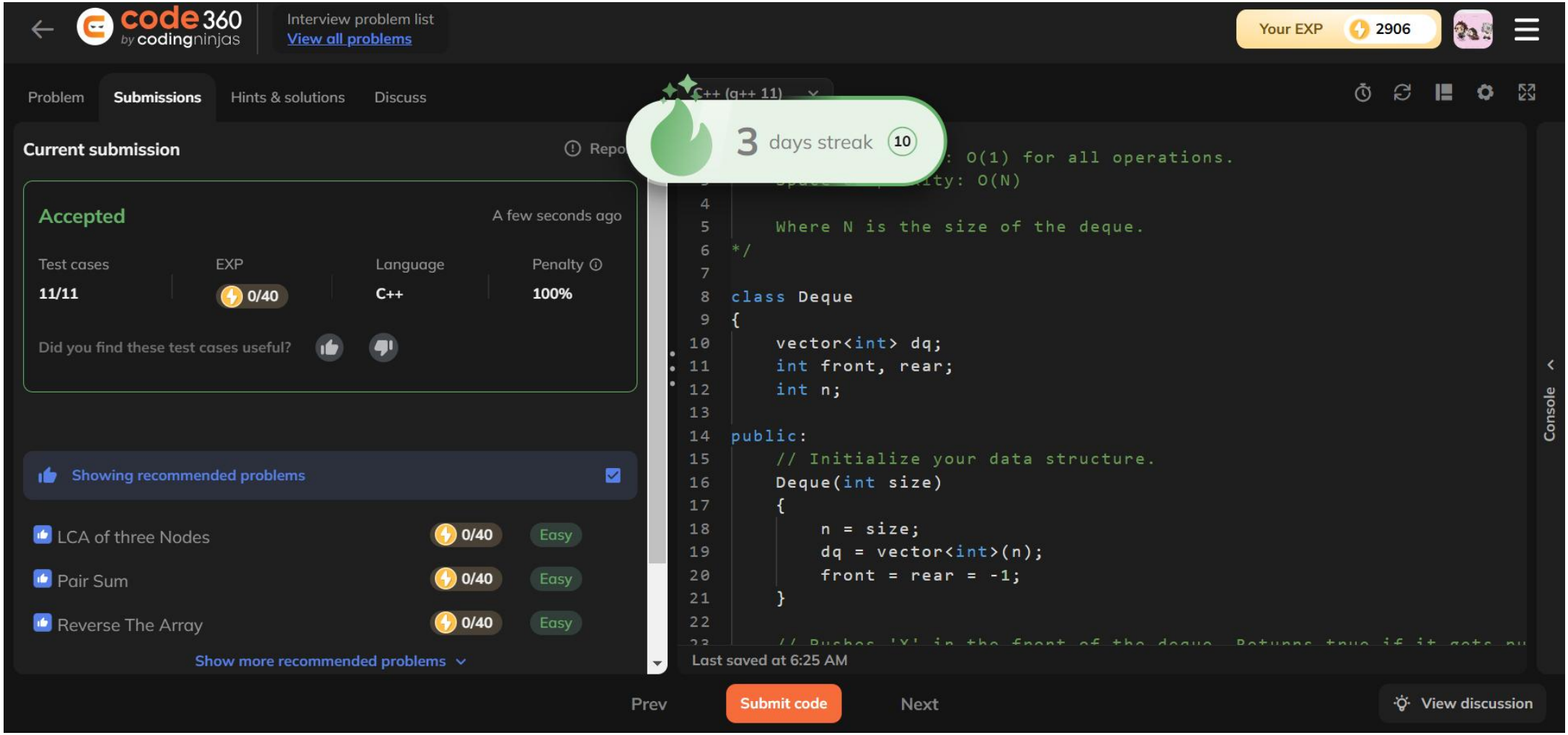
14.9K | 180 | 136 Online

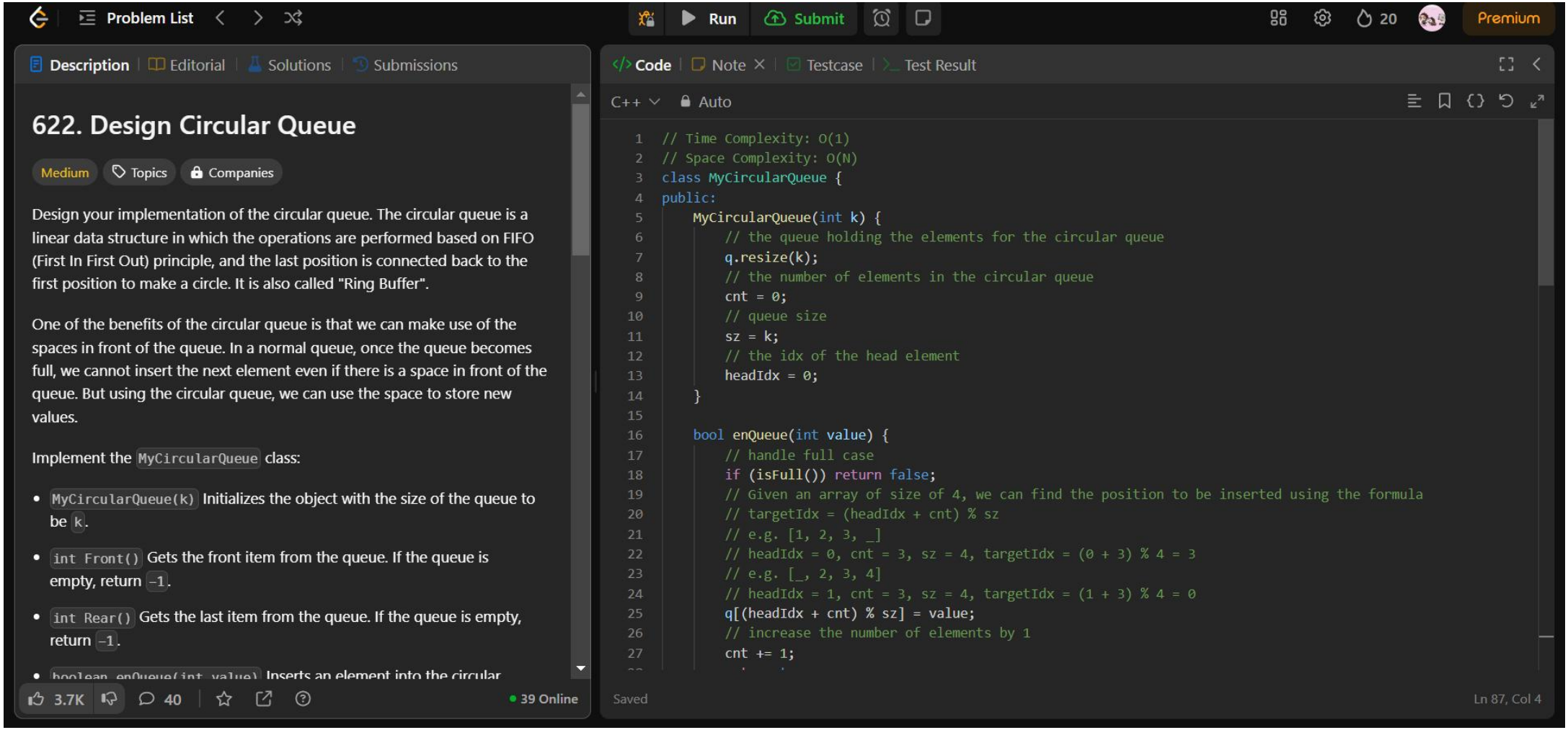
Code | Note | Testcase | Test Result

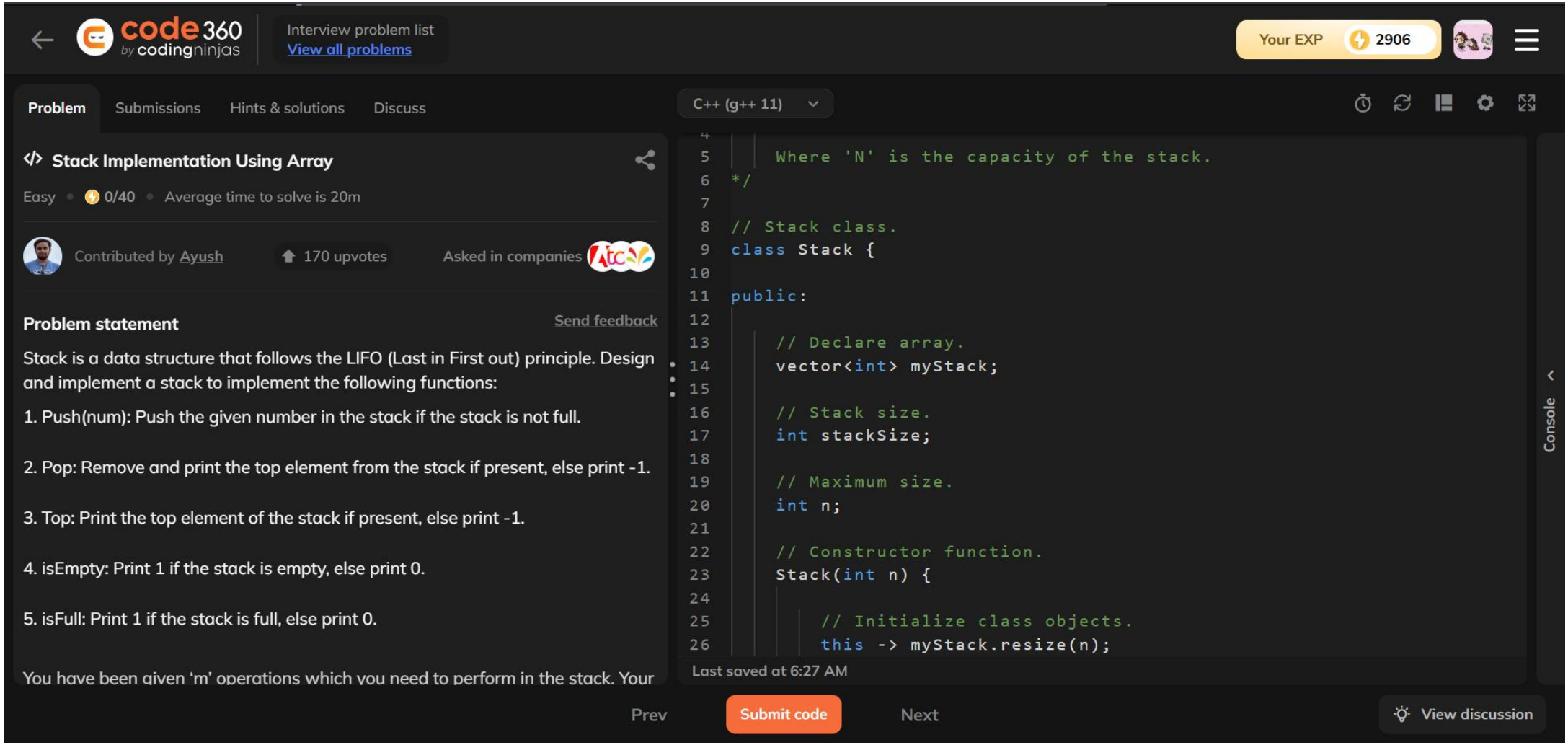
C++ | Auto

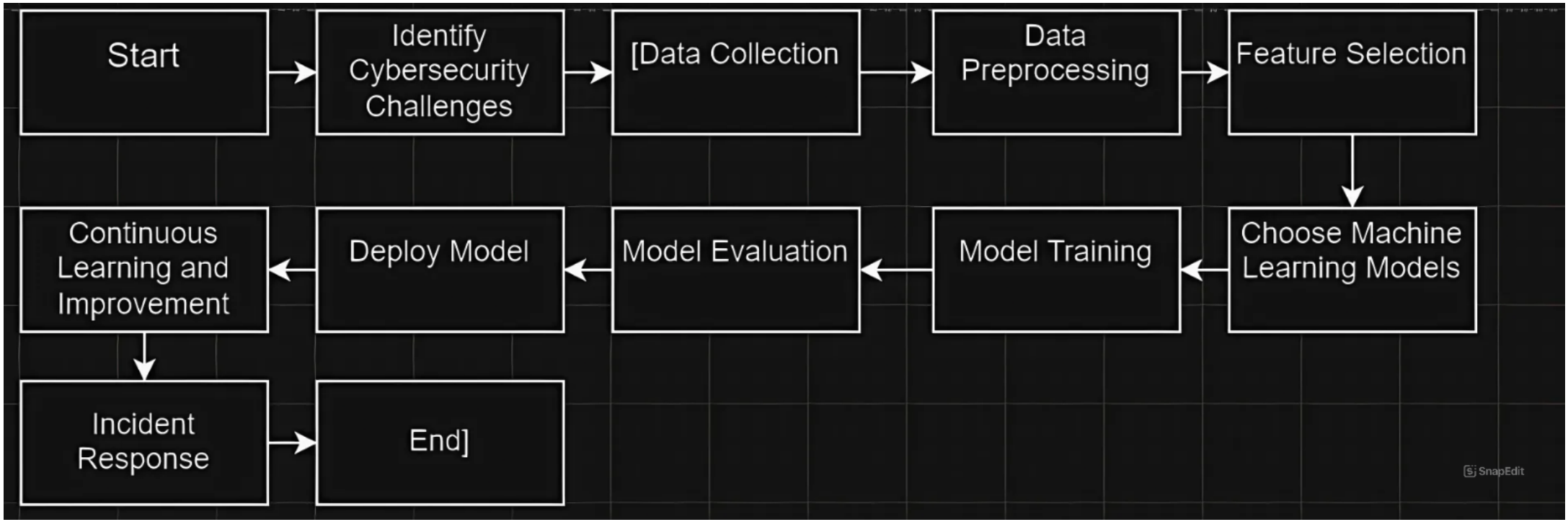
```cpp
class MinStack {
public:
    long long minValue;
    stack<long long> minStack;
    MinStack() {
        minValue = 0;
    }

    void push(int val) {
        long long nval = val;
        if(minStack.empty())
            minValue = val;
        else if(val < minValue)
        {
            nval = 2LL*val - minValue;
            minValue = val;
        }
        minStack.push(nval);
    }

    void pop() {
        if(minStack.top() < minValue)
            minValue = 2LL*minValue - minStack.top();
        minStack.pop();
    }

    int top() {
```

Saved


code360 by codingninjas
Interview problem list
View all problems
Your EXP 2906
Problem
Submissions
Hints & solutions
Discuss
C++ (g++ 11)
3 days streak 10
Current submission
Accepted
A few seconds ago
Test cases
11/11
EXP
0/40
Language
C++
Penalty
100%
Did you find these test cases useful?
Showing recommended problems
LCA of three Nodes 0/40 Easy
Pair Sum 0/40 Easy
Reverse The Array 0/40 Easy
Show more recommended problems
: O(1) for all operations.
ity: O(N)
Where N is the size of the deque.
*/
class Deque
{
vector<int> dq;
int front, rear;
int n;
public:
// Initialize your data structure.
Deque(int size)
{
n = size;
dq = vector<int>(n);
front = rear = -1;
}
Console
Last saved at 6:25 AM
Prev
Submit code
Next
View discussion


Problem List
Run
Submit
20
Premium
Description
Editorial
Solutions
Submissions
622. Design Circular Queue
Medium
Topics
Companies
Design your implementation of the circular queue. The circular queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle, and the last position is connected back to the first position to make a circle. It is also called "Ring Buffer".
One of the benefits of the circular queue is that we can make use of the spaces in front of the queue. In a normal queue, once the queue becomes full, we cannot insert the next element even if there is a space in front of the queue. But using the circular queue, we can use the space to store new values.
Implement the MyCircularQueue class:
MyCircularQueue(k) Initializes the object with the size of the queue to be k.
int Front() Gets the front item from the queue. If the queue is empty, return -1.
int Rear() Gets the last item from the queue. If the queue is empty, return -1.
3.7K
40
39 Online
Code
Note
Testcase
Test Result
C++
Auto
// Time Complexity: O(1)
// Space Complexity: O(N)
class MyCircularQueue {
public:
MyCircularQueue(int k) {
// the queue holding the elements for the circular queue
q.resize(k);
// the number of elements in the circular queue
cnt = 0;
// queue size
sz = k;
// the idx of the head element
headIdx = 0;
}
bool enQueue(int value) {
// handle full case
if (isFull()) return false;
// Given an array of size of 4, we can find the position to be inserted using the formula
// targetIdx = (headIdx + cnt) % sz
// e.g. [1, 2, 3, _]
// headIdx = 0, cnt = 3, sz = 4, targetIdx = (0 + 3) % 4 = 3
// e.g. [_, 2, 3, 4]
// headIdx = 1, cnt = 3, sz = 4, targetIdx = (1 + 3) % 4 = 0
q[(headIdx + cnt) % sz] = value;
// increase the number of elements by 1
cnt += 1;
Saved
Ln 87, Col 4


code 360 by codingninjas
Interview problem list
View all problems
Your EXP 2906
Problem
Submissions
Hints & solutions
Discuss
C++ (g++ 11)
Stack Implementation Using Array
Easy • 0/40 • Average time to solve is 20m
Contributed by Ayush
170 upvotes
Asked in companies
Problem statement
Send feedback
Stack is a data structure that follows the LIFO (Last in First out) principle. Design and implement a stack to implement the following functions:
1. Push(num): Push the given number in the stack if the stack is not full.
2. Pop: Remove and print the top element from the stack if present, else print -1.
3. Top: Print the top element of the stack if present, else print -1.
4. isEmpty: Print 1 if the stack is empty, else print 0.
5. isFull: Print 1 if the stack is full, else print 0.
You have been given 'm' operations which you need to perform in the stack. Your
Where 'N' is the capacity of the stack.
*/
// Stack class.
class Stack {
public:
// Declare array.
vector<int> myStack;
// Stack size.
int stackSize;
// Maximum size.
int n;
// Constructor function.
Stack(int n) {
// Initialize class objects.
this -> myStack.resize(n);
Last saved at 6:27 AM
Console
Prev
Submit code
Next
View discussion


Start
Identify Cybersecurity Challenges
[Data Collection
Data Preprocessing
Feature Selection
Choose Machine Learning Models
Model Training
Model Evaluation
Deploy Model
Continuous Learning and Improvement
Incident Response
End]
SnapEdit