# Assignment – 6

**Name: KARANDEEP SINGH** **UID: 22BCS16869**

**Subject: Advance Programming Lab – II** **Section: 22BCS-IoT_626 [B]**

**Subject Code: 22CSP-351** **Date of Submission: 17th Mar,2025**

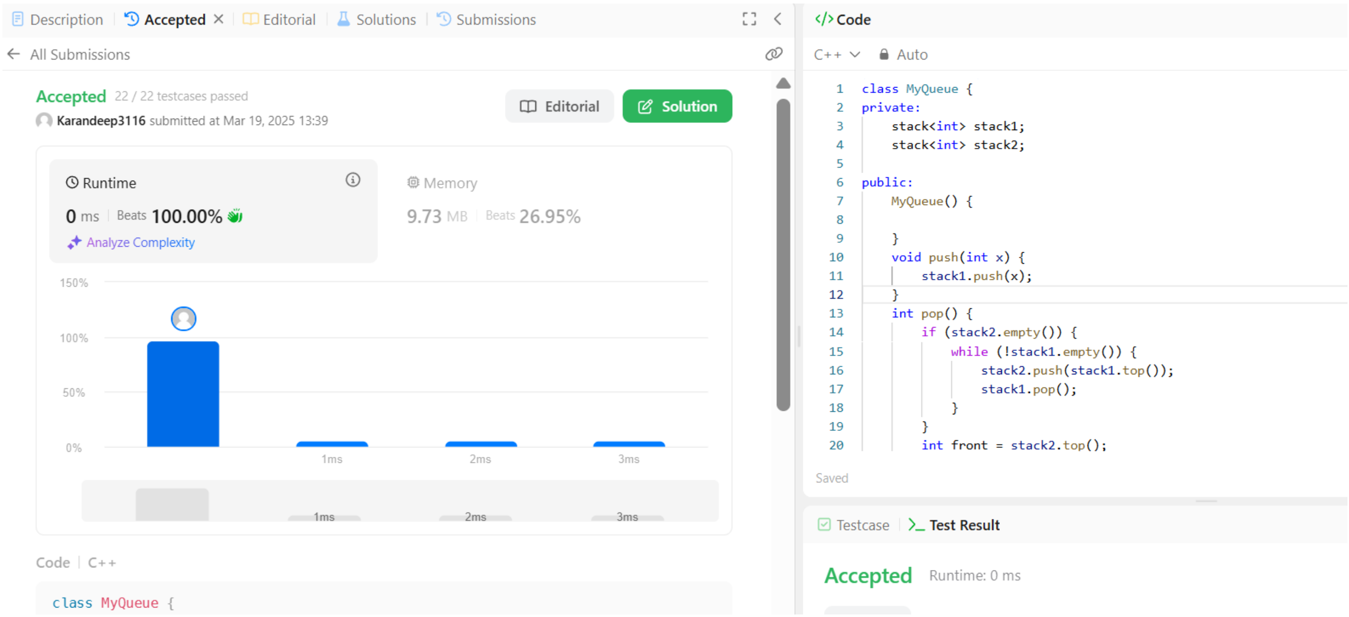
1. Implement Queue using Stack

CODE:

```
class MyQueue {
private:
    stack<int> stack1;
    stack<int> stack2;

public:
    MyQueue() {

    }
    void push(int x) {
        stack1.push(x);
    }
    int pop() {
        if (stack2.empty()) {
            while (!stack1.empty()) {
                stack2.push(stack1.top());
                stack1.pop();
            }
        }
        int front = stack2.top();
        stack2.pop();
        return front;
    }
    int peek() {
        if (stack2.empty()) {
            while (!stack1.empty()) {
                stack2.push(stack1.top());
                stack1.pop();
            }
        }
        return stack2.top();
    }
    bool empty() {
        return stack1.empty() && stack2.empty();
    }
};
```



Accepted 22 / 22 testcases passed
Karandeep3116 submitted at Mar 19, 2025 13:39

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
9.73 MB | Beats 26.95%

Code | C++

```
class MyQueue {
```

C++ Auto

```
class MyQueue {
private:
    stack<int> stack1;
    stack<int> stack2;

public:
    MyQueue() {

    }
    void push(int x) {
        stack1.push(x);
    }
    int pop() {
        if (stack2.empty()) {
            while (!stack1.empty()) {
                stack2.push(stack1.top());
                stack1.pop();
            }
        }
        int front = stack2.top();
```

Saved

Accepted Runtime: 0 ms

2. <u>Implement Stack using Queue</u>

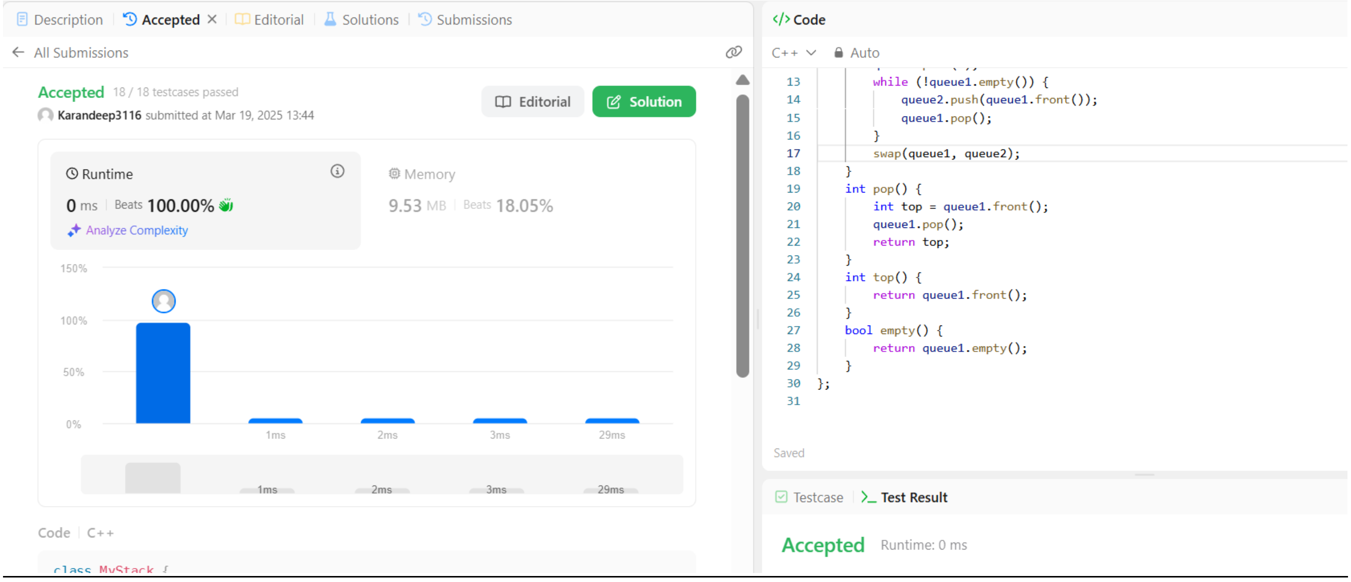
<u>CODE:</u>

```
class MyStack {
private:
    queue<int> queue1;
    queue<int> queue2;

public:
    MyStack() {

    }
    void push(int x) {

        queue2.push(x);
        while (!queue1.empty()) {
            queue2.push(queue1.front());
            queue1.pop();
        }
        swap(queue1, queue2);
    }
    int pop() {
        int top = queue1.front();
        queue1.pop();
        return top;
    }
    int top() {
        return queue1.front();
    }
    bool empty() {
        return queue1.empty();
    }
};
```



Description | Accepted × | Editorial | Solutions | Submissions

← All Submissions

Accepted 18 / 18 testcases passed

Karandeep3116 submitted at Mar 19, 2025 13:44

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

9.53 MB | Beats 18.05%

Code | C++

```
class MyStack {
```

Code

C++ Auto

```
        while (!queue1.empty()) {
            queue2.push(queue1.front());
            queue1.pop();
        }
        swap(queue1, queue2);
    }
    int pop() {
        int top = queue1.front();
        queue1.pop();
        return top;
    }
    int top() {
        return queue1.front();
    }
    bool empty() {
        return queue1.empty();
    }
};
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

### 3. <u>Implement Min Stack using Two Stacks</u>

<u>CODE:</u>

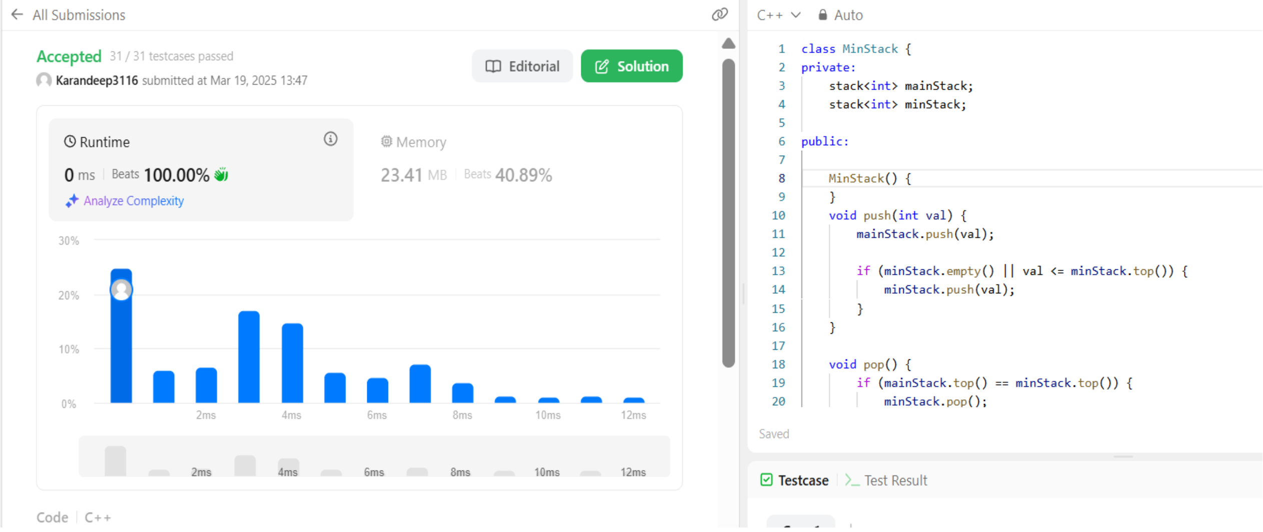
```
class MinStack {
private:
    stack<int> mainStack;
    stack<int> minStack;

public:

    MinStack() {
    }
    void push(int val) {
        mainStack.push(val);

        if (minStack.empty() || val <= minStack.top()) {
            minStack.push(val);
        }
    }

    void pop() {
        if (mainStack.top() == minStack.top()) {
            minStack.pop();
        }
        mainStack.pop();
    }
    int top() {
        return mainStack.top();
    }
    int getMin() {
        return minStack.top();
    }
};
```



Accepted 31 / 31 testcases passed
Karandeep3116 submitted at Mar 19, 2025 13:47

0 ms | Beats 100.00%

23.41 MB | Beats 40.89%

```
class MinStack {
private:
    stack<int> mainStack;
    stack<int> minStack;

public:

    MinStack() {
    }
    void push(int val) {
        mainStack.push(val);

        if (minStack.empty() || val <= minStack.top()) {
            minStack.push(val);
        }
    }

    void pop() {
        if (mainStack.top() == minStack.top()) {
            minStack.pop();
```

## 4. Implement Circular Queue using Queue

CODE:

```
class MyCircularQueue {
private:
    vector<int> queue;
    int front, rear;
    int capacity;
    int size;

public:

    MyCircularQueue(int k) {
        queue = vector<int>(k, 0);
        front = rear = -1;
        capacity = k;
        size = 0;
```

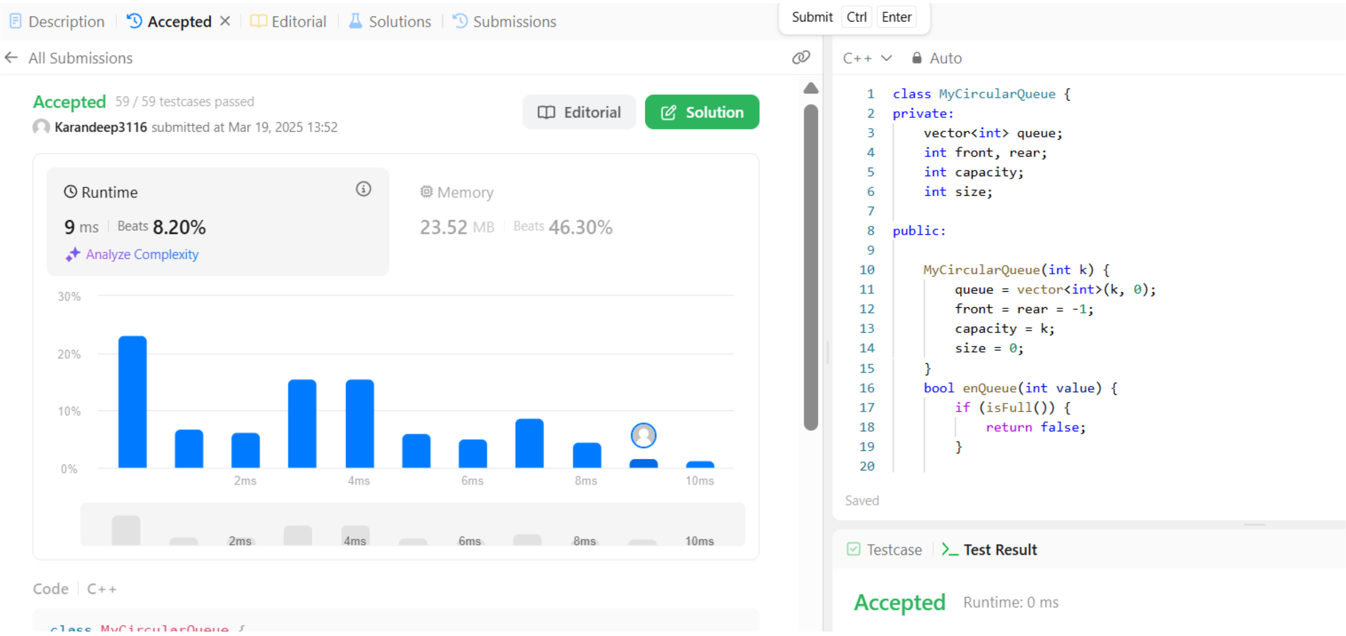
```
}
bool enQueue(int value) {
    if (isFull()) {
        return false;
    }

    if (isEmpty()) {
        front = 0;
    }

    rear = (rear + 1) % capacity;
    queue[rear] = value;
    size++;
    return true;
}
bool deQueue() {
    if (isEmpty()) {
        return false;
    }

    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % capacity;
    }
    size--;
    return true;
}
int Front() {
    if (isEmpty()) {
        return -1;
    }
    return queue[front];
}
int Rear() {
    if (isEmpty()) {
        return -1;
    }
    return queue[rear];
}
bool isEmpty() {
    return size == 0;
}
bool isFull() {
    return size == capacity;
```

```
        }
    };
```



Accepted 59 / 59 testcases passed
Karandeep3116 submitted at Mar 19, 2025 13:52

Runtime
9 ms | Beats 8.20%

Memory
23.52 MB | Beats 46.30%

```
class MyCircularQueue {
private:
    vector<int> queue;
    int front, rear;
    int capacity;
    int size;

public:

    MyCircularQueue(int k) {
        queue = vector<int>(k, 0);
        front = rear = -1;
        capacity = k;
        size = 0;
    }
    bool enQueue(int value) {
        if (isFull()) {
            return false;
        }
```

Accepted Runtime: 0 ms

5. <u>Implement BST Level Order Traversal using Queue (BFS)</u>

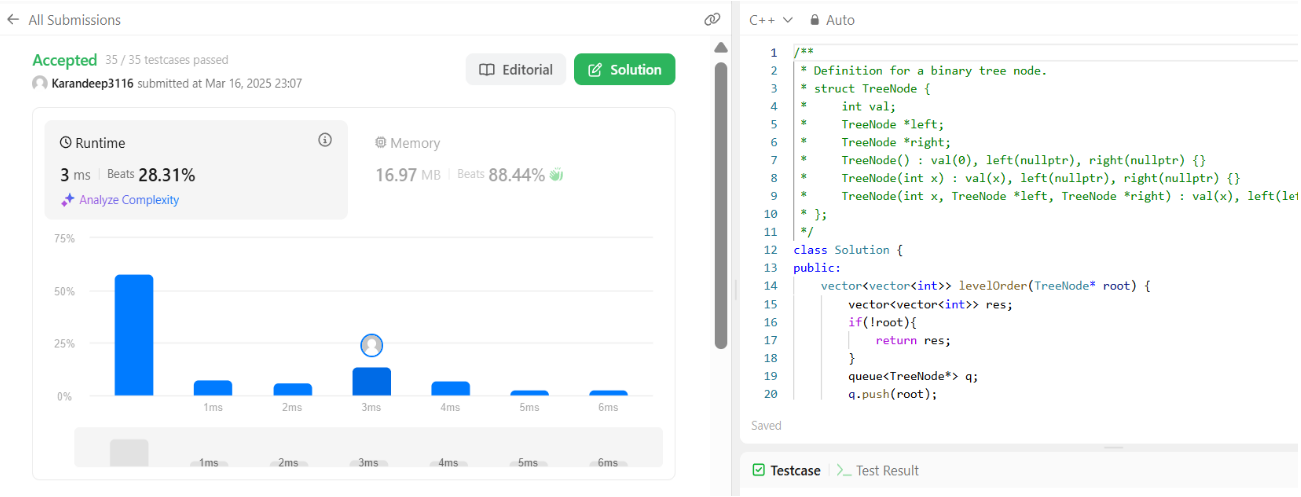
<u>CODE:</u>

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> res;
        if(!root){
            return res;
        }
        queue<TreeNode*> q;
        q.push(root);
        while(!q.empty()){
            int n= q.size();
            vector<int> ans;
            while(n--){
                TreeNode* temp = q.front();
                q.pop();
                ans.push_back(temp->val);
                if(temp->left)
                    q.push(temp->left);

                if(temp->right)
                    q.push(temp->right);
            }
                res.push_back(ans);


        }
```

```
            return res;
        }
    };
```



Accepted 35 / 35 testcases passed

Karandeep3116 submitted at Mar 16, 2025 23:07

Runtime
3 ms | Beats 28.31%

Memory
16.97 MB | Beats 88.44%

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(lef
 * };
 */
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> res;
        if(!root){
            return res;
        }
        queue<TreeNode*> q;
        q.push(root);
```

6. Implement Priority Queue using Stack

CODE:

```
class PriorityQueue {
    stack<int> stack1;
    stack<int> stack2;

public:

    void push(int x) {
        while (!stack1.empty() && stack1.top() > x) {
            stack2.push(stack1.top());
            stack1.pop();
        }

        stack1.push(x);

        while (!stack2.empty()) {
            stack1.push(stack2.top());
            stack2.pop();
        }
    }
    int pop() {
        if (stack1.empty()) {
            cout << "Priority Queue is empty!" << endl;
            return -1;
        }
        int topElement = stack1.top();
        stack1.pop();
        return topElement;
    }
    int peek() {
        if (stack1.empty()) {
            cout << "Priority Queue is empty!" << endl;
            return -1;
        }
        return stack1.top();
    }
    bool empty() {
        return stack1.empty();
    }
};
```

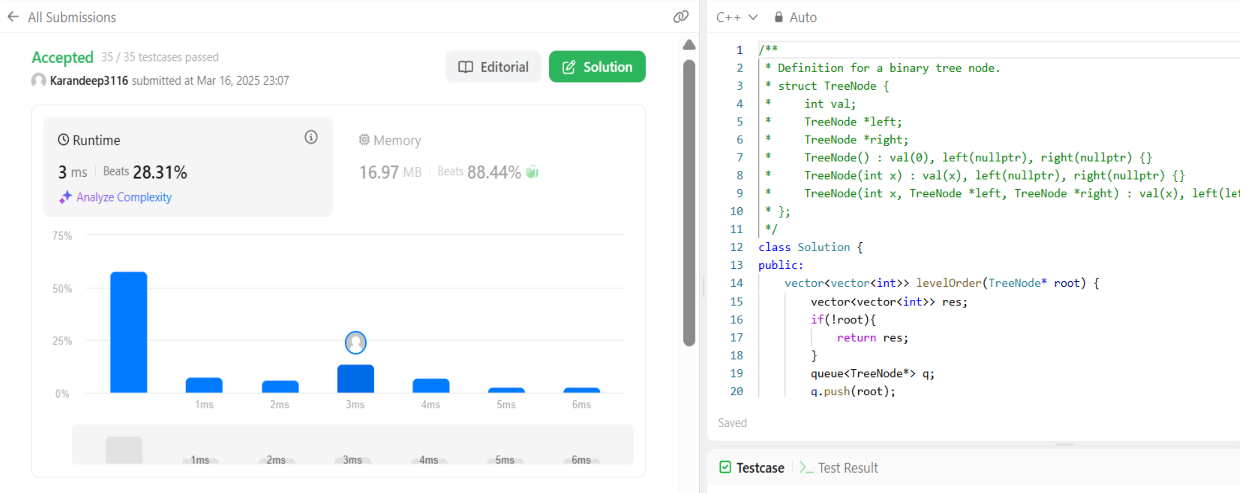
7. Implement BST Level Order Traversal using Queue (BFS)

CODE:

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> res;
        if(!root){
            return res;
        }
        queue<TreeNode*> q;
        q.push(root);
        while(!q.empty()){
            int n= q.size();
            vector<int> ans;
            while(n--){
                TreeNode* temp = q.front();
                q.pop();
                ans.push_back(temp->val);
                if(temp->left)
                    q.push(temp->left);

                if(temp->right)
                    q.push(temp->right);
            }
                res.push_back(ans);


        }
        return res;
    }
};
```



Accepted 35 / 35 testcases passed

Karandeep3116 submitted at Mar 16, 2025 23:07

Runtime: 3 ms | Beats 28.31%

Memory: 16.97 MB | Beats 88.44%

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(le
 * };
 */
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> res;
        if(!root){
            return res;
        }
        queue<TreeNode*> q;
        q.push(root);
```