

232. Implement Queue using Stacks

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. You may simulate a stack using a list or deque (double-ended queue) as long as you use only a stack's standard operations.

Example 1:

Input
["MyQueue", "push", "push", "peek", "pop", "empty"]

Output
[[], [], [2], [], []]

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

bool empty() {
    return input.empty() && output.empty();
}
```

Accepted Runtime: 0 ms

Case 1

Input
["MyQueue", "push", "push", "peek", "pop", "empty"]

Output
[[], [], [2], [], []]

225. Implement Stack using Queues

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 queue) as long as you use only a queue's standard operations.

Example 1:

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

Output
[[], [], [2], [], []]

```
class MyStack {
private:
    std::queue<int> q;
public:
    MyStack() {}

    void push(int x) {
        q.push(x);
        for (int i = 0; i < q.size() - 1; i++) {
            q.push(q.front());
            q.pop();
        }
    }

    int pop() {
        int top = q.front();
    }
}
```

Accepted Runtime: 0 ms

Case 1

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

Output
[[], [], [2], [], []]

Implement Stack using Linked List.

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
    Node(int val) : data(val), next(nullptr) {}
};

class Stack {
private:
    Node* top;
public:
    Stack() : top(nullptr) {}

    void push(int value) {
        Node* newNode = new Node(value);
        newNode->next = top;
        top = newNode;
        cout << value << " pushed to stack\n";
    }

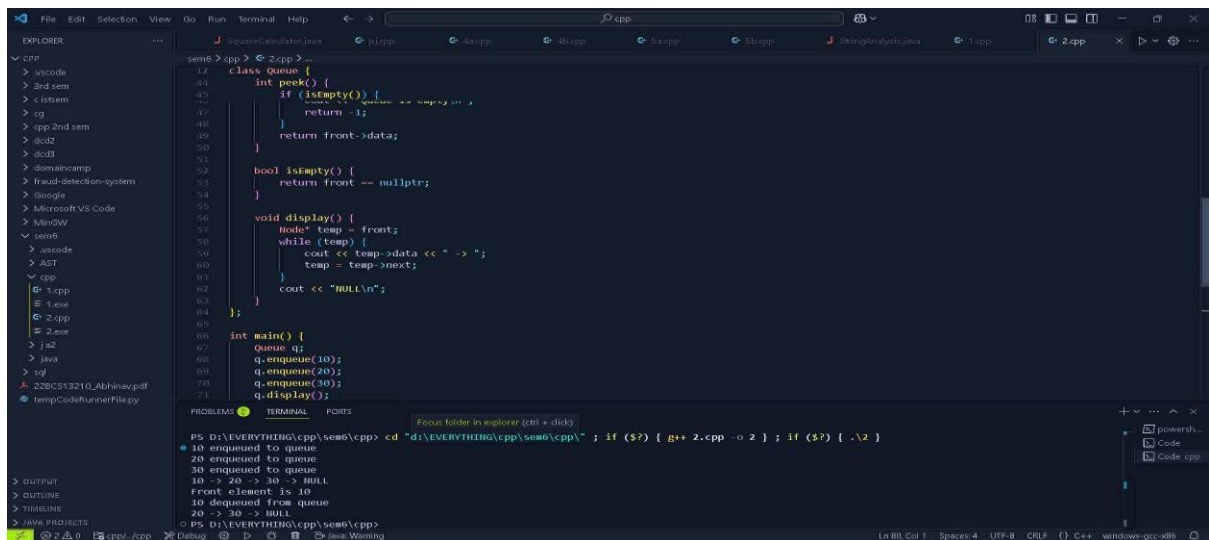
    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        top = top->next;
    }

    bool isEmpty() {
        return top == nullptr;
    }
};
```

PS D:\EVERYTHING\cpp> cd "d:\EVERYTHING\cpp\sem6\cpp" ; if (\$?) { g++ 1.cpp -o 1 ; if (\$?) { .\1 }

10 pushed to stack
20 pushed to stack
30 pushed to stack
30 -> 20 -> 10 -> NULL
Top element is 30
30 popped from stack
20 -> 10 -> NULL
PS D:\EVERYTHING\cpp\sem6\cpp>

Queue using Linked List

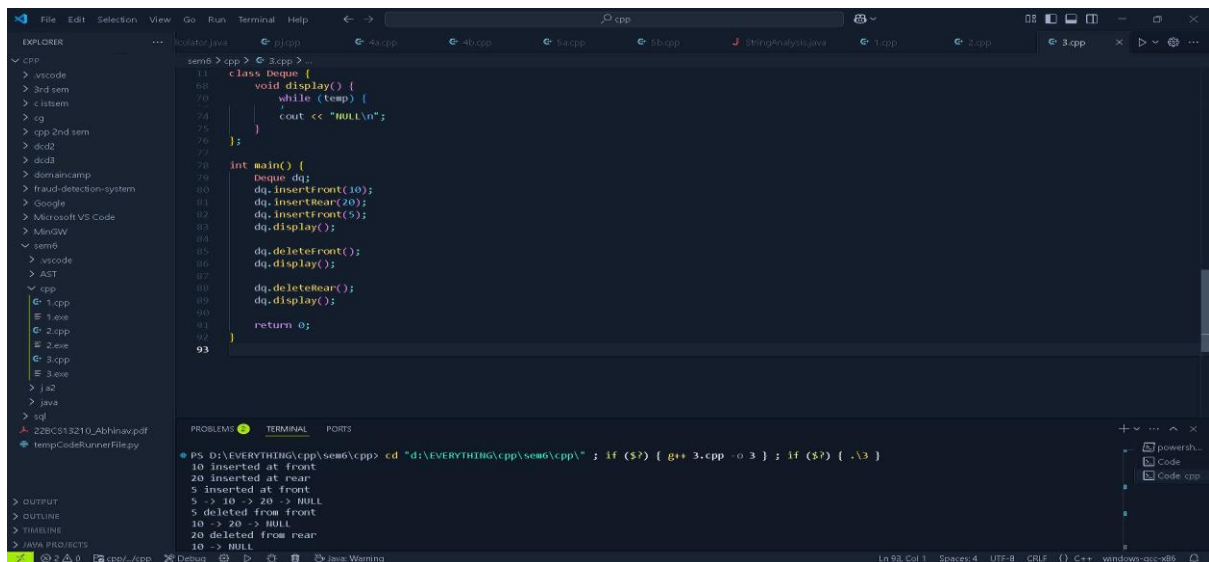


```
12 class Queue {
13     int peek() {
14         if (isEmpty()) {
15             return -1;
16         }
17         return front->data;
18     }
19
20     bool isEmpty() {
21         return front == nullptr;
22     }
23
24     void display() {
25         Node* temp = front;
26         while (temp) {
27             cout << temp->data << " -> ";
28             temp = temp->next;
29         }
30         cout << "NULL\n";
31     }
32
33     int main() {
34         Queue q;
35         q.enqueue(10);
36         q.enqueue(20);
37         q.enqueue(30);
38         q.display();
39     }
40 }
```

Terminal Output:

```
PS D:\EVERYTHING\cpp\sem6\cpp> cd "d:\EVERYTHING\cpp\sem6\cpp\" ; if ($?) { g++ 2.cpp -o 2 ; if ($?) { .\2 }
10 -> 20 -> 30 -> NULL
Front element is 10
10 dequeued from queue
20 -> 30 -> NULL
PS D:\EVERYTHING\cpp\sem6\cpp>
```

Deque using Doubly Linked List

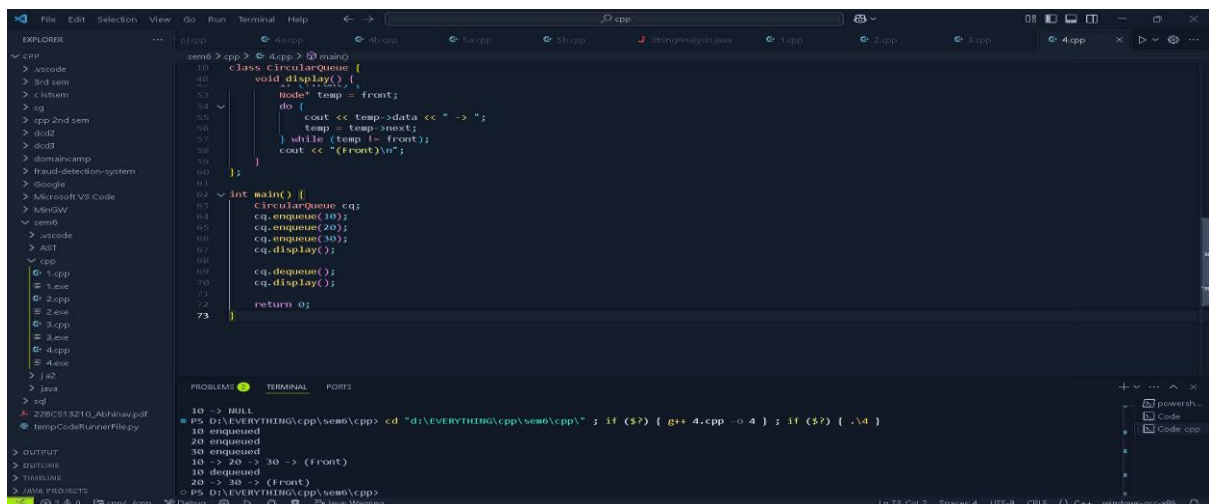


```
11 class Deque {
12     void display() {
13         while (temp) {
14             cout << "NULL\n";
15         }
16     }
17
18     int main() {
19         Deque dq;
20         dq.insertFront(10);
21         dq.insertRear(20);
22         dq.insertFront(5);
23         dq.display();
24
25         dq.deleteFront();
26         dq.display();
27
28         dq.deleteRear();
29         dq.display();
30
31         return 0;
32     }
33 }
```

Terminal Output:

```
PS D:\EVERYTHING\cpp\sem6\cpp> cd "d:\EVERYTHING\cpp\sem6\cpp\" ; if ($?) { g++ 3.cpp -o 3 ; if ($?) { .\3 }
10 inserted at front
20 inserted at rear
5 inserted at front
5 -> 10 -> 20 -> NULL
5 deleted from front
10 -> 20 -> NULL
20 deleted from rear
10 -> NULL
PS D:\EVERYTHING\cpp\sem6\cpp>
```

Circular Queue using Linked List



```
10 class CircularQueue {
11     void display() {
12         Node* temp = front;
13         do {
14             cout << temp->data << " -> ";
15             temp = temp->next;
16         } while (temp != front);
17         cout << "(front)\n";
18     }
19
20     int main() {
21         CircularQueue cq;
22         cq.enqueue(10);
23         cq.enqueue(20);
24         cq.enqueue(30);
25         cq.display();
26
27         cq.dequeue();
28         cq.display();
29
30         return 0;
31     }
32 }
```

Terminal Output:

```
PS D:\EVERYTHING\cpp\sem6\cpp> cd "d:\EVERYTHING\cpp\sem6\cpp\" ; if ($?) { g++ 4.cpp -o 4 ; if ($?) { .\4 }
10 -> NULL
10 dequeued
20 dequeued
10 -> 20 -> 30 -> (front)
20 -> 30 -> (front)
PS D:\EVERYTHING\cpp\sem6\cpp>
```

Min Stack using Linked List.

```
1 class MinStack {
2     Node* top;
3 public:
4     MinStack() : top(nullptr) {}
5
6     void push(int value) {
7         int minval = (top == nullptr) ? value : min(value, top->min);
8         Node* newnode = new Node(value, minval);
9         newnode->next = top;
10        top = newnode;
11        cout << value << " pushed\n";
12    }
13
14    void pop() {
15        if (!top) {
16            cout << "Stack Underflow\n";
17            return;
18        }
19        Node* temp = top;
20        top = top->next;
21        cout << temp->data << " popped\n";
22        delete temp;
23    }
24
25    int getMin() {
26        if (!top) {
27            cout << "Stack is empty\n";
28            return -1;
29        }
30    }
31
32 };
33
34 int main() {
35     MinStack s;
36     s.push(10);
37     s.push(20);
38     s.push(5);
39     cout << "Minimum value: " << s.getMin() << endl;
40     s.pop();
41     cout << "Minimum value: " << s.getMin() << endl;
42     return 0;
43 }
```

Terminal Output:

```
10 pushed
20 pushed
5 pushed
Minimum value: 5
5 popped
20 > 10 -> NULL
Minimum value: 10
```

Hash Table using Linked List

```
1 class HashTable {
2     Node* head;
3 public:
4     HashTable() : head(nullptr) {}
5
6     void insert(int key, int value) {
7         Node* newnode = new Node(key, value);
8         newnode->next = head;
9         head = newnode;
10    }
11
12    Node* search(int key) {
13        Node* temp = head;
14        while (temp != nullptr) {
15            if (temp->key == key) {
16                return temp;
17            }
18            temp = temp->next;
19        }
20        return nullptr;
21    }
22
23    void display() {
24        Node* temp = head;
25        while (temp != nullptr) {
26            cout << temp->key << " : " << temp->value << " ";
27            temp = temp->next;
28        }
29        cout << endl;
30    }
31
32 };
33
34 int main() {
35     HashTable ht;
36     ht.insert(15, 100);
37     ht.insert(25, 200);
38     ht.insert(35, 300);
39     ht.display();
40     cout << "Value for key 25: " << ht.search(25)->value << endl;
41     ht.remove(25);
42     ht.display();
43     return 0;
44 }
```

Terminal Output:

```
15 : 100
25 : 200
35 : 300
Value for key 25: 200
15 : 100
35 : 300
```

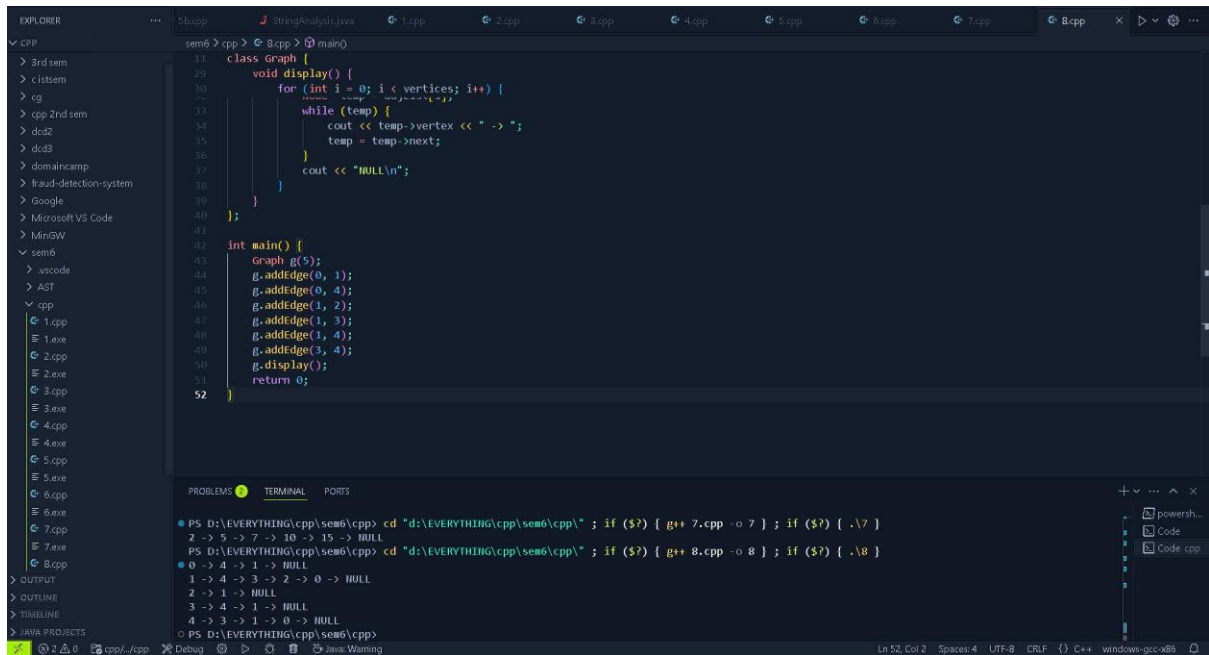
BST using Linked List

```
1 class BST {
2     Node* root;
3 public:
4     BST() : root(nullptr) {}
5
6     void insert(int key) {
7         Node* newnode = new Node(key);
8         if (root == nullptr) {
9             root = newnode;
10        } else {
11            insertRec(root, newnode);
12        }
13    }
14
15    void insertRec(Node* root, Node* newnode) {
16        if (root == nullptr) {
17            root = newnode;
18            return;
19        }
20        if (key < root->key) {
21            insertRec(root->left, newnode);
22        } else {
23            insertRec(root->right, newnode);
24        }
25    }
26
27    Node* search(int key) {
28        Node* temp = root;
29        while (temp != nullptr) {
30            if (temp->key == key) {
31                return temp;
32            }
33            if (key < temp->key) {
34                temp = temp->left;
35            } else {
36                temp = temp->right;
37            }
38        }
39        return nullptr;
40    }
41
42    void display() {
43        Node* temp = root;
44        while (temp != nullptr) {
45            cout << temp->key << " ";
46            temp = temp->left;
47        }
48        cout << endl;
49    }
50
51 };
52
53 int main() {
54     BST bst;
55     bst.insert(10);
56     bst.insert(5);
57     bst.insert(15);
58     bst.insert(2);
59     bst.insert(7);
60     bst.display();
61     cout << "Value for key 7: " << bst.search(7)->key << endl;
62     return 0;
63 }
```

Terminal Output:

```
2 5 10 15 7
Value for key 7: 7
```

Graph using Linked List



```
sem6 > c++ 8.cpp > main()
11 class Graph {
12     void display() {
13         for (int i = 0; i < vertices; i++) {
14             while (temp) {
15                 cout << temp->vertex << " -> ";
16                 temp = temp->next;
17             }
18             cout << "NULL\n";
19         }
20     }
21 };
22
23 int main() {
24     Graph g(5);
25     g.addedge(0, 1);
26     g.addedge(0, 4);
27     g.addedge(1, 2);
28     g.addedge(1, 3);
29     g.addedge(1, 4);
30     g.addedge(3, 4);
31     g.display();
32     return 0;
33 }
```

PROBLEMS TERMINAL PORTS

```
PS D:\EVERYTHING\cpp\sem6\cpp> cd "d:\EVERYTHING\cpp\sem6\cpp\" ; if ($?) { g++ 7.cpp -o 7 } ; if ($?) { .\7 }
2 -> 5 -> 7 -> 10 -> 15 -> NULL
PS D:\EVERYTHING\cpp\sem6\cpp> cd "d:\EVERYTHING\cpp\sem6\cpp\" ; if ($?) { g++ 8.cpp -o 8 } ; if ($?) { .\8 }
0 -> 4 -> 1 -> NULL
1 -> 4 -> 3 -> 2 -> 0 -> NULL
2 -> 1 -> NULL
3 -> 4 -> 1 -> NULL
4 -> 3 -> 1 -> 0 -> NULL
PS D:\EVERYTHING\cpp\sem6\cpp>
```

Ln 52, Col 2 Spaces: 4 UTF-8 CRLF C++ windows-gcc-x86