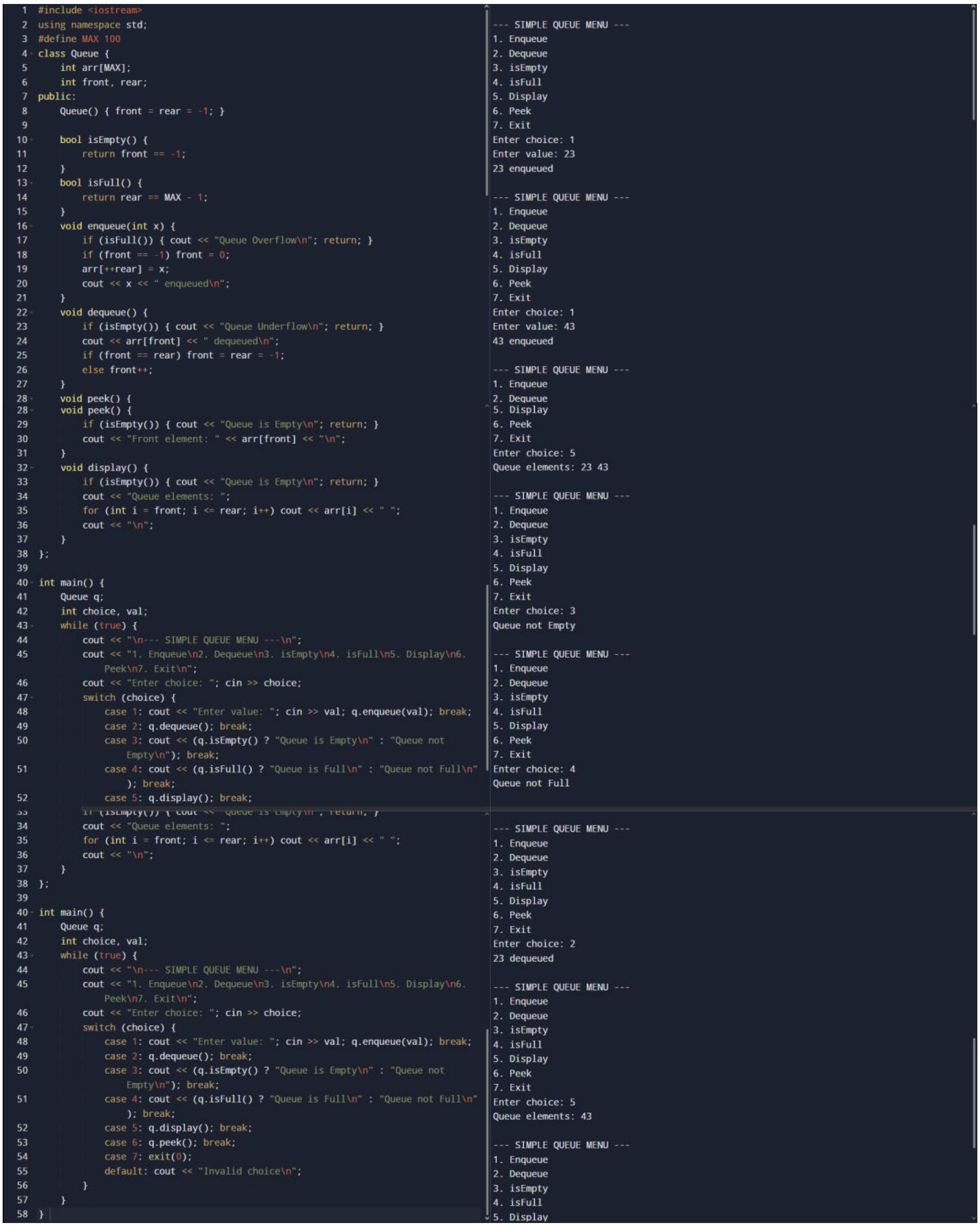
Assignment – 4
Ayush garg(1024030878)

Q1

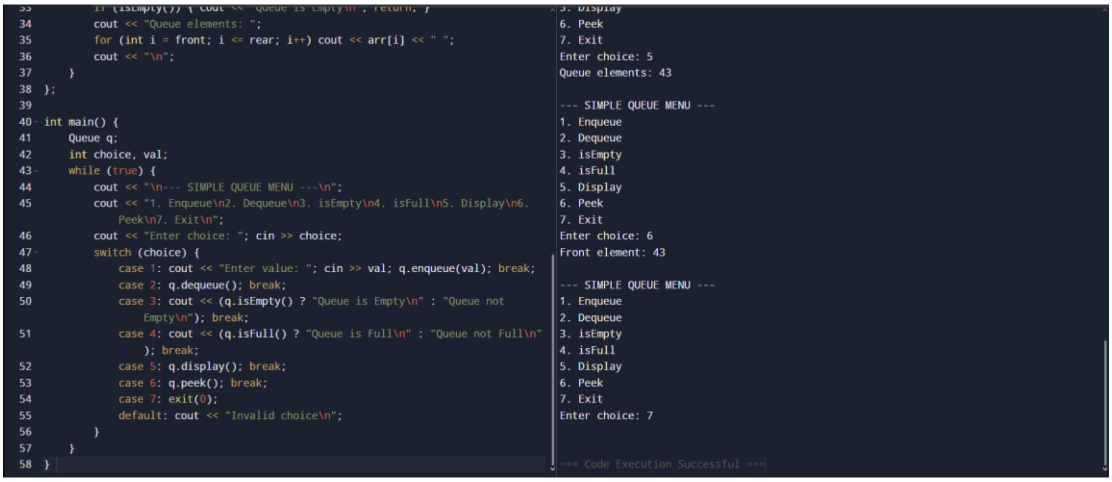


```cpp
#include <iostream>
using namespace std;
#define MAX 100
class Queue {
    int arr[MAX];
    int front, rear;
public:
    Queue() { front = rear = -1; }

    bool isEmpty() {
        return front == -1;
    }
    bool isFull() {
        return rear == MAX - 1;
    }
    void enqueue(int x) {
        if (isFull()) { cout << "Queue Overflow\n"; return; }
        if (front == -1) front = 0;
        arr[++rear] = x;
        cout << x << " enqueued\n";
    }
    void dequeue() {
        if (isEmpty()) { cout << "Queue Underflow\n"; return; }
        cout << arr[front] << " dequeued\n";
        if (front == rear) front = rear = -1;
        else front++;
    }
    void peek() {
    void peek() {
        if (isEmpty()) { cout << "Queue is Empty\n"; return; }
        cout << "Front element: " << arr[front] << "\n";
    }
    void display() {
        if (isEmpty()) { cout << "Queue is Empty\n"; return; }
        cout << "Queue elements: ";
        for (int i = front; i <= rear; i++) cout << arr[i] << " ";
        cout << "\n";
    }
};

int main() {
    Queue q;
    int choice, val;
    while (true) {
        cout << "\n--- SIMPLE QUEUE MENU ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n7. Exit\n";
        cout << "Enter choice: "; cin >> choice;
        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: cout << (q.isEmpty() ? "Queue is Empty\n" : "Queue not Empty\n"); break;
            case 4: cout << (q.isFull() ? "Queue is Full\n" : "Queue not Full\n"); break;
            case 5: q.display(); break;
```

```
--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 23
23 enqueued

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 43
43 enqueued

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
5. Display
6. Peek
7. Exit
Enter choice: 5
Queue elements: 23 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 3
Queue not Empty

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 4
Queue not Full
```

```cpp
        if (isEmpty()) { cout << "Queue is Empty\n"; return; }
        cout << "Queue elements: ";
        for (int i = front; i <= rear; i++) cout << arr[i] << " ";
        cout << "\n";
    }
};

int main() {
    Queue q;
    int choice, val;
    while (true) {
        cout << "\n--- SIMPLE QUEUE MENU ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n7. Exit\n";
        cout << "Enter choice: "; cin >> choice;
        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: cout << (q.isEmpty() ? "Queue is Empty\n" : "Queue not Empty\n"); break;
            case 4: cout << (q.isFull() ? "Queue is Full\n" : "Queue not Full\n"); break;
            case 5: q.display(); break;
            case 6: q.peek(); break;
            case 7: exit(0);
            default: cout << "Invalid choice\n";
        }
    }
}
```

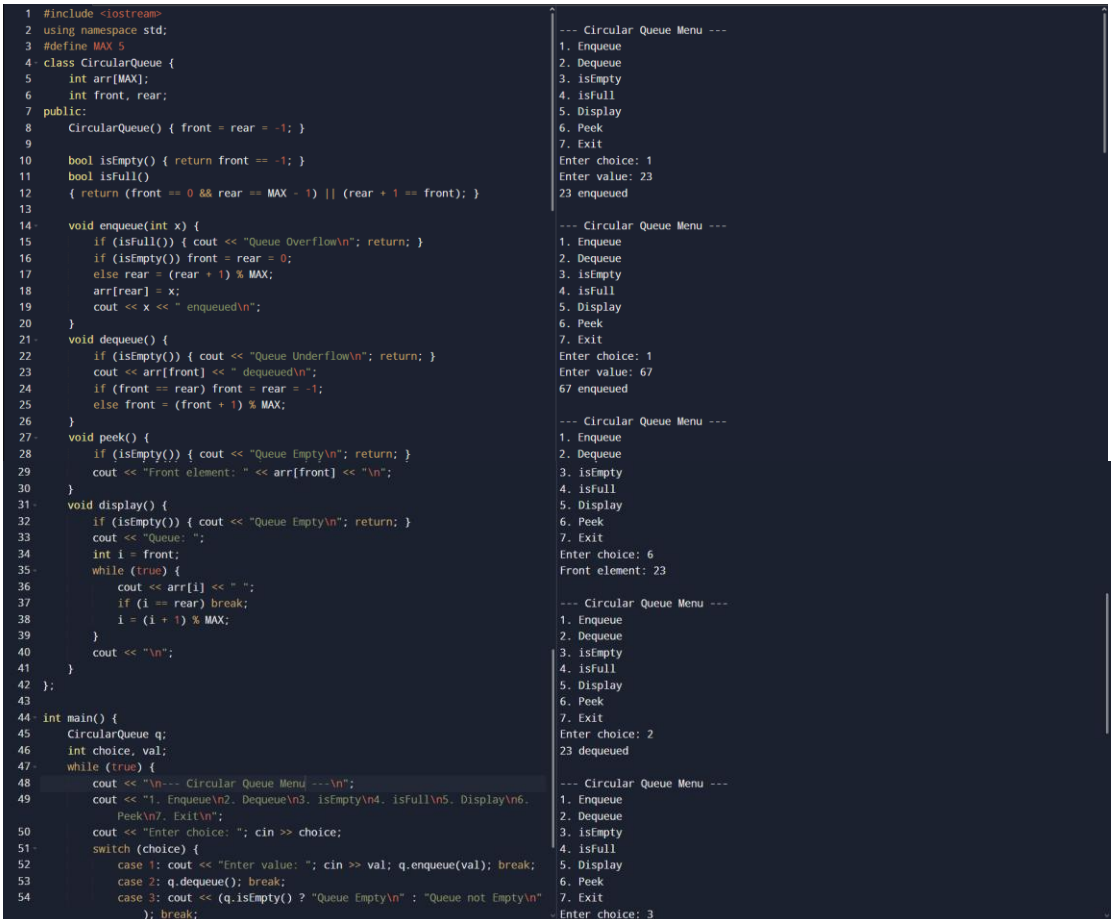
```
--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 2
23 dequeued

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 5
Queue elements: 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
```



```cpp
        if (isEmpty()) { cout << "Queue is Empty\n"; return; }
        cout << "Queue elements: ";
        for (int i = front; i <= rear; i++) cout << arr[i] << " ";
        cout << "\n";
    }
};

int main() {
    Queue q;
    int choice, val;
    while (true) {
        cout << "\n--- SIMPLE QUEUE MENU ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6.
            Peek\n7. Exit\n";
        cout << "Enter choice: "; cin >> choice;
        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: cout << (q.isEmpty() ? "Queue is Empty\n" : "Queue not
                Empty\n"); break;
            case 4: cout << (q.isFull() ? "Queue is Full\n" : "Queue not Full\n"
                ); break;
            case 5: q.display(); break;
            case 6: q.peek(); break;
            case 7: exit(0);
            default: cout << "Invalid choice\n";
        }
    }
}
```

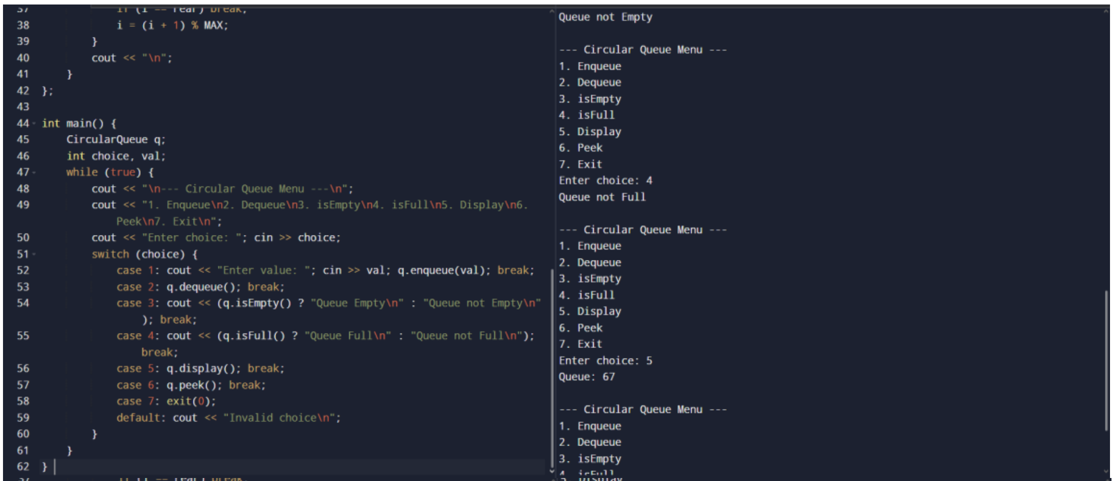
```
5. Display
6. Peek
7. Exit
Enter choice: 5
Queue elements: 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 6
Front element: 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 7


=== Code Execution Successful ===
```

Q2



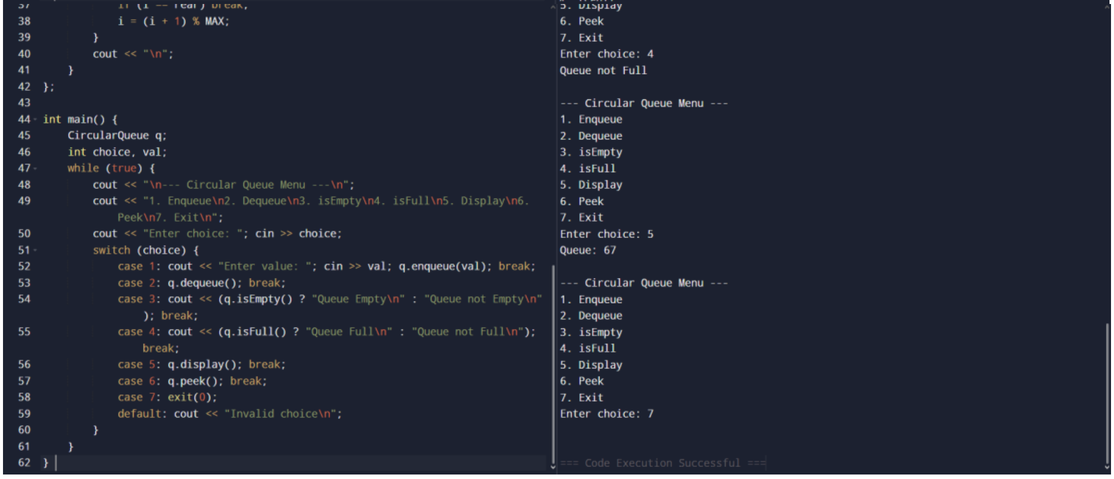
```cpp
#include <iostream>
using namespace std;
#define MAX 5
class CircularQueue {
    int arr[MAX];
    int front, rear;
public:
    CircularQueue() { front = rear = -1; }

    bool isEmpty() { return front == -1; }
    bool isFull()
    { return (front == 0 && rear == MAX - 1) || (rear + 1 == front); }

    void enqueue(int x) {
        if (isFull()) { cout << "Queue Overflow\n"; return; }
        if (isEmpty()) front = rear = 0;
        else rear = (rear + 1) % MAX;
        arr[rear] = x;
        cout << x << " enqueued\n";
    }
    void dequeue() {
        if (isEmpty()) { cout << "Queue Underflow\n"; return; }
        cout << arr[front] << " dequeued\n";
        if (front == rear) front = rear = -1;
        else front = (front + 1) % MAX;
    }
    void peek() {
        if (isEmpty()) { cout << "Queue Empty\n"; return; }
        cout << "Front element: " << arr[front] << "\n";
    }
    void display() {
        if (isEmpty()) { cout << "Queue Empty\n"; return; }
        cout << "Queue: ";
        int i = front;
        while (true) {
            cout << arr[i] << " ";
            if (i == rear) break;
            i = (i + 1) % MAX;
        }
        cout << "\n";
    }
};

int main() {
    CircularQueue q;
    int choice, val;
    while (true) {
        cout << "\n--- Circular Queue Menu ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6.
            Peek\n7. Exit\n";
        cout << "Enter choice: "; cin >> choice;
        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: cout << (q.isEmpty() ? "Queue Empty\n" : "Queue not Empty\n"
                ); break;
```

```

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 23
23 enqueued

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 67
67 enqueued

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 6
Front element: 23

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 2
23 dequeued

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 3
```



```
            if (i == rear) break;
            i = (i + 1) % MAX;
        }
        cout << "\n";
    }
};

int main() {
    CircularQueue q;
    int choice, val;
    while (true) {
        cout << "\n--- Circular Queue Menu ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6.
            Peek\n7. Exit\n";
        cout << "Enter choice: "; cin >> choice;
        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: cout << (q.isEmpty() ? "Queue Empty\n" : "Queue not Empty\n"
                ); break;
            case 4: cout << (q.isFull() ? "Queue Full\n" : "Queue not Full\n");
                break;
            case 5: q.display(); break;
            case 6: q.peek(); break;
            case 7: exit(0);
            default: cout << "Invalid choice\n";
        }
    }
}
```

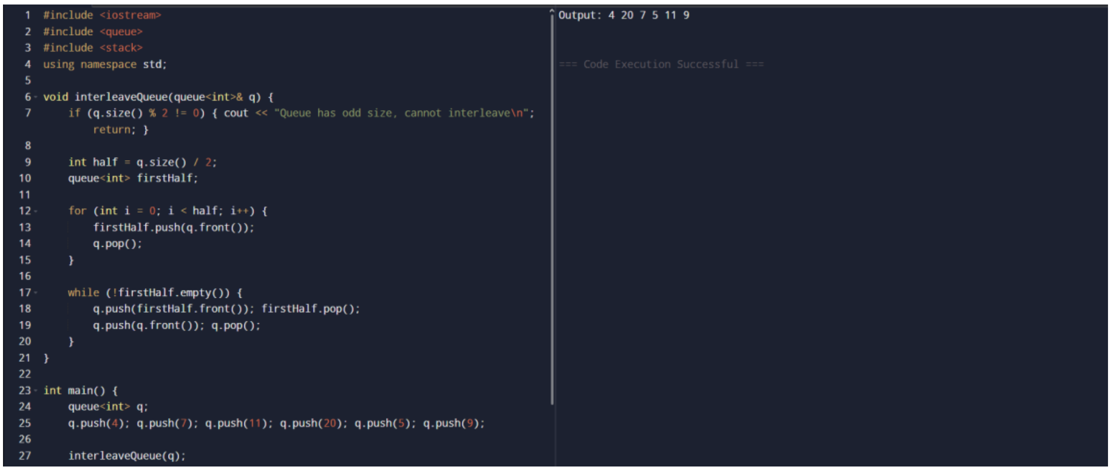
```
Queue not Empty

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 4
Queue not Full

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 5
Queue: 67

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
```



```
            if (i == rear) break;
            i = (i + 1) % MAX;
        }
        cout << "\n";
    }
};

int main() {
    CircularQueue q;
    int choice, val;
    while (true) {
        cout << "\n--- Circular Queue Menu ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6.
            Peek\n7. Exit\n";
        cout << "Enter choice: "; cin >> choice;
        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: cout << (q.isEmpty() ? "Queue Empty\n" : "Queue not Empty\n"
                ); break;
            case 4: cout << (q.isFull() ? "Queue Full\n" : "Queue not Full\n");
                break;
            case 5: q.display(); break;
            case 6: q.peek(); break;
            case 7: exit(0);
            default: cout << "Invalid choice\n";
        }
    }
}
```

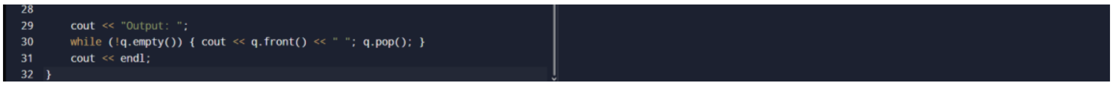
```
5. Display
6. Peek
7. Exit
Enter choice: 4
Queue not Full

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 5
Queue: 67

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 7


=== Code Execution Successful ===
```

Q3



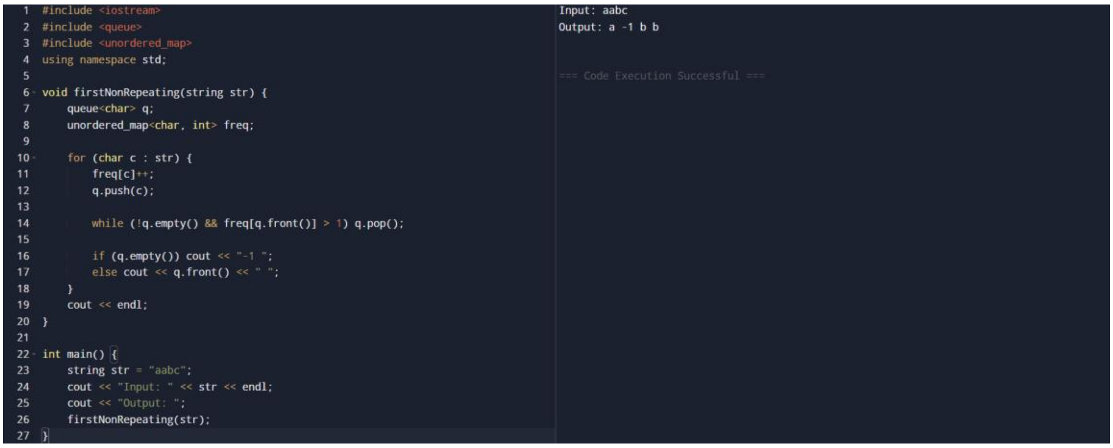
```
#include <iostream>
#include <queue>
#include <stack>
using namespace std;

void interleaveQueue(queue<int>& q) {
    if (q.size() % 2 != 0) { cout << "Queue has odd size, cannot interleave\n";
        return; }

    int half = q.size() / 2;
    queue<int> firstHalf;

    for (int i = 0; i < half; i++) {
        firstHalf.push(q.front());
        q.pop();
    }

    while (!firstHalf.empty()) {
        q.push(firstHalf.front()); firstHalf.pop();
        q.push(q.front()); q.pop();
    }
}

int main() {
    queue<int> q;
    q.push(4); q.push(7); q.push(11); q.push(20); q.push(5); q.push(9);

    interleaveQueue(q);
```

```
Output: 4 20 7 5 11 9

=== Code Execution Successful ===
```



```

    cout << "Output: ";
    while (!q.empty()) { cout << q.front() << " "; q.pop(); }
    cout << endl;
}
```

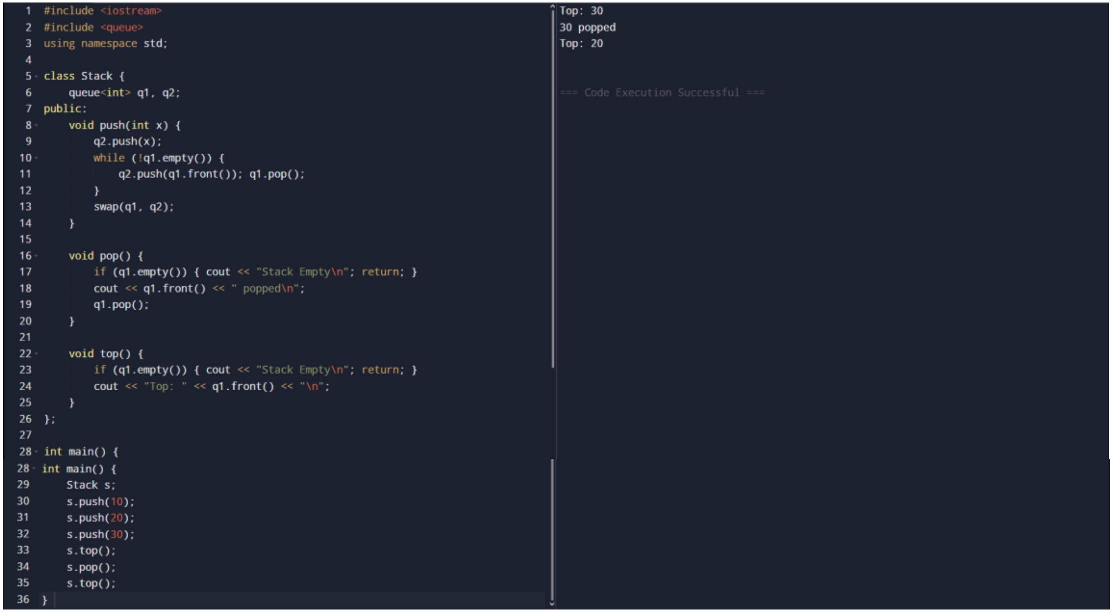
Q4



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

void firstNonRepeating(string str) {
    queue<char> q;
    unordered_map<char, int> freq;

    for (char c : str) {
        freq[c]++;
        q.push(c);

        while (!q.empty() && freq[q.front()] > 1) q.pop();

        if (q.empty()) cout << "-1 ";
        else cout << q.front() << " ";
    }
    cout << endl;
}

int main() {
    string str = "aabc";
    cout << "Input: " << str << endl;
    cout << "Output: ";
    firstNonRepeating(str);
}
```

```
Input: aabc
Output: a -1 b b


=== Code Execution Successful ===
```

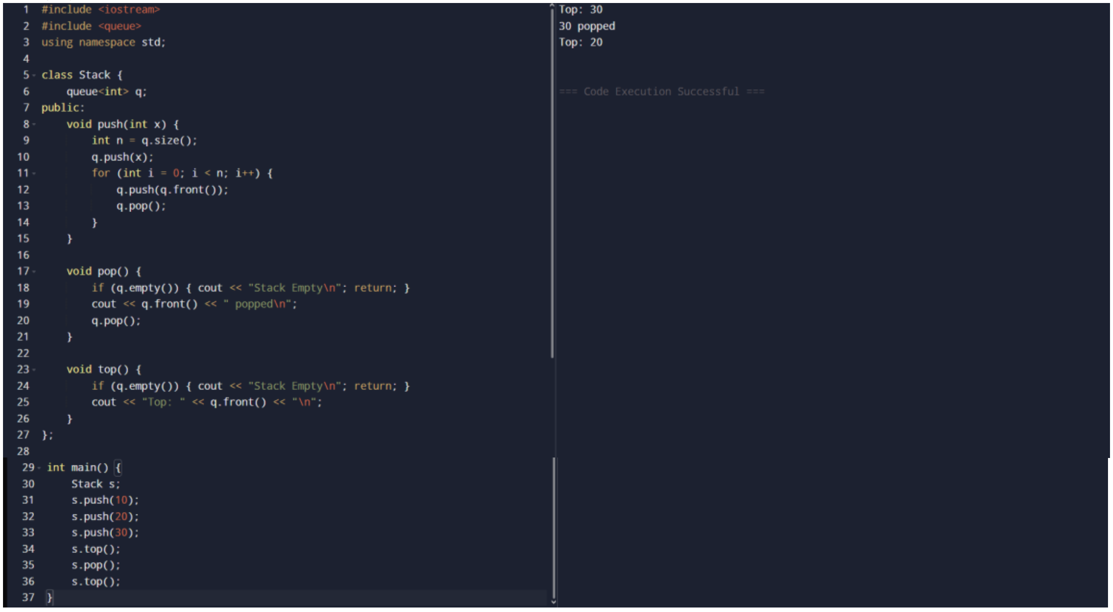
Q5(a)



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

class Stack {
    queue<int> q1, q2;
public:
    void push(int x) {
        q2.push(x);
        while (!q1.empty()) {
            q2.push(q1.front()); q1.pop();
        }
        swap(q1, q2);
    }

    void pop() {
        if (q1.empty()) { cout << "Stack Empty\n"; return; }
        cout << q1.front() << " popped\n";
        q1.pop();
    }

    void top() {
        if (q1.empty()) { cout << "Stack Empty\n"; return; }
        cout << "Top: " << q1.front() << "\n";
    }
};

int main() {
int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);
    s.top();
    s.pop();
    s.top();
}
```

```
Top: 30
30 popped
Top: 20


=== Code Execution Successful ===
```

Q5(b)



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

class Stack {
    queue<int> q;
public:
    void push(int x) {
        int n = q.size();
        q.push(x);
        for (int i = 0; i < n; i++) {
            q.push(q.front());
            q.pop();
        }
    }

    void pop() {
        if (q.empty()) { cout << "Stack Empty\n"; return; }
        cout << q.front() << " popped\n";
        q.pop();
    }

    void top() {
        if (q.empty()) { cout << "Stack Empty\n"; return; }
        cout << "Top: " << q.front() << "\n";
    }
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);
    s.top();
    s.pop();
    s.top();
}
```

```
Top: 30
30 popped
Top: 20


=== Code Execution Successful ===
```