LAB TASK

LAB TASK NO 5

SUBJECT: DATA STRUCTURE AND ALGORITHM

SUBMITTED TO: HIJAB DURRANI

SUBMITTED BY: MISBAH ULLAH JAN

CLASS CODE: BSSE-2024A

SEMESTER: 3RD

DATE: 11/27/2025

DEPARTMENT OF: SOFTWARE ENGINEERING

CECOS UNIVERSITY HAYATABAD PESHAWAR

LAB TASK 1

1. Modify your queue program so that:

- **Before inserting (enqueue), the program checks if the value already exists.**
- **If it exists, do not insert and print:**

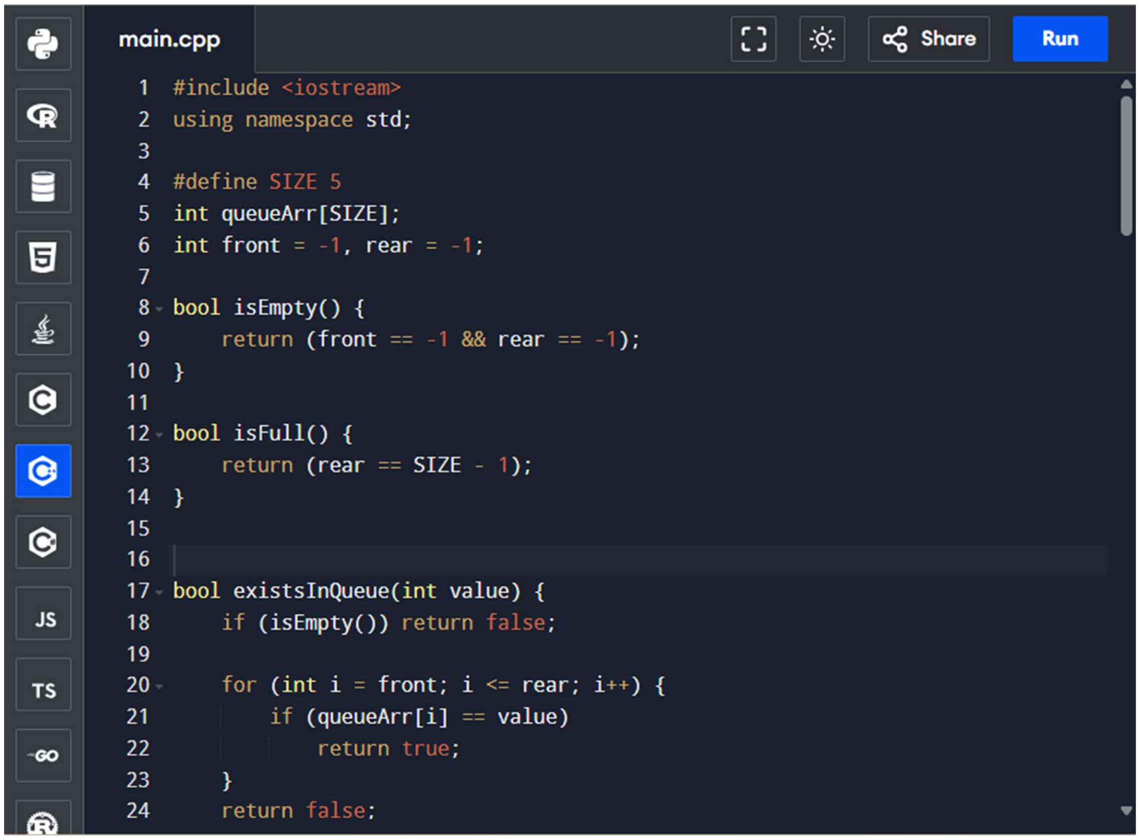
Duplicate value! Insertion not allowed.

Otherwise insert normally.

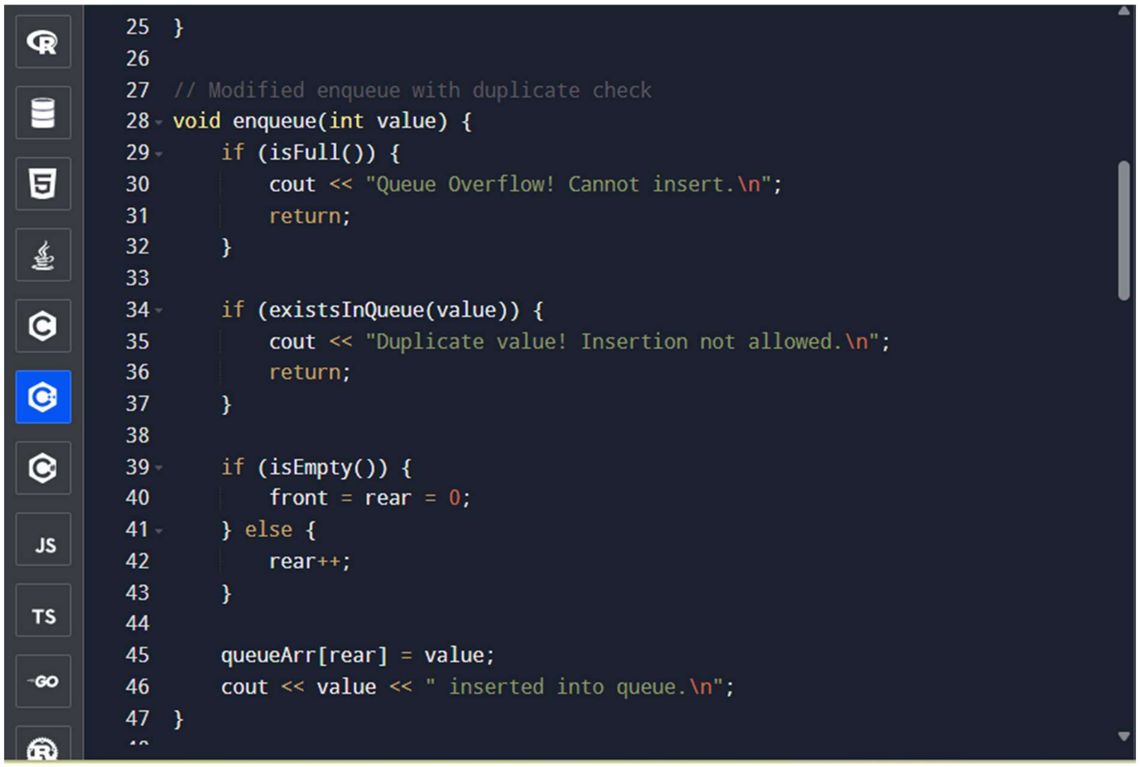
Algorithm for Enqueue with Duplicate Checking

Algorithm: enqueue(value)

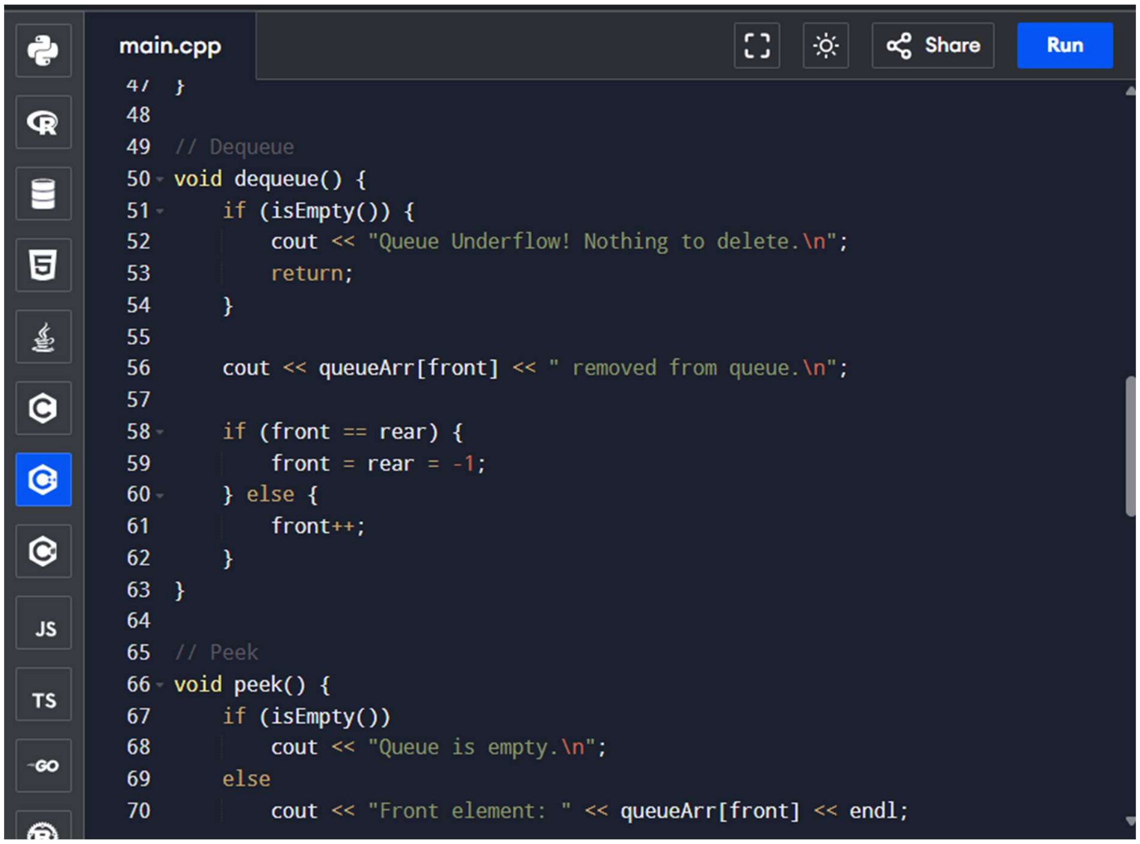
```
1. If queue is full
    o Print: "Queue Overflow! Cannot insert."
    o Stop.
2. Check for duplicate value
    o For each index from FRONT to REAR:
        ▪ If queueArr[i] == value:
            ▪ Print: "Duplicate value! Insertion not allowed."
            ▪ Stop.
3. If queue is empty
    o Set FRONT = 0
    o Set REAR = 0
4. Else (queue has elements)
    o Increase REAR by 1
5. Insert the value
    o queueArr[REAR] = value
6. Print
    o value inserted successfully
```



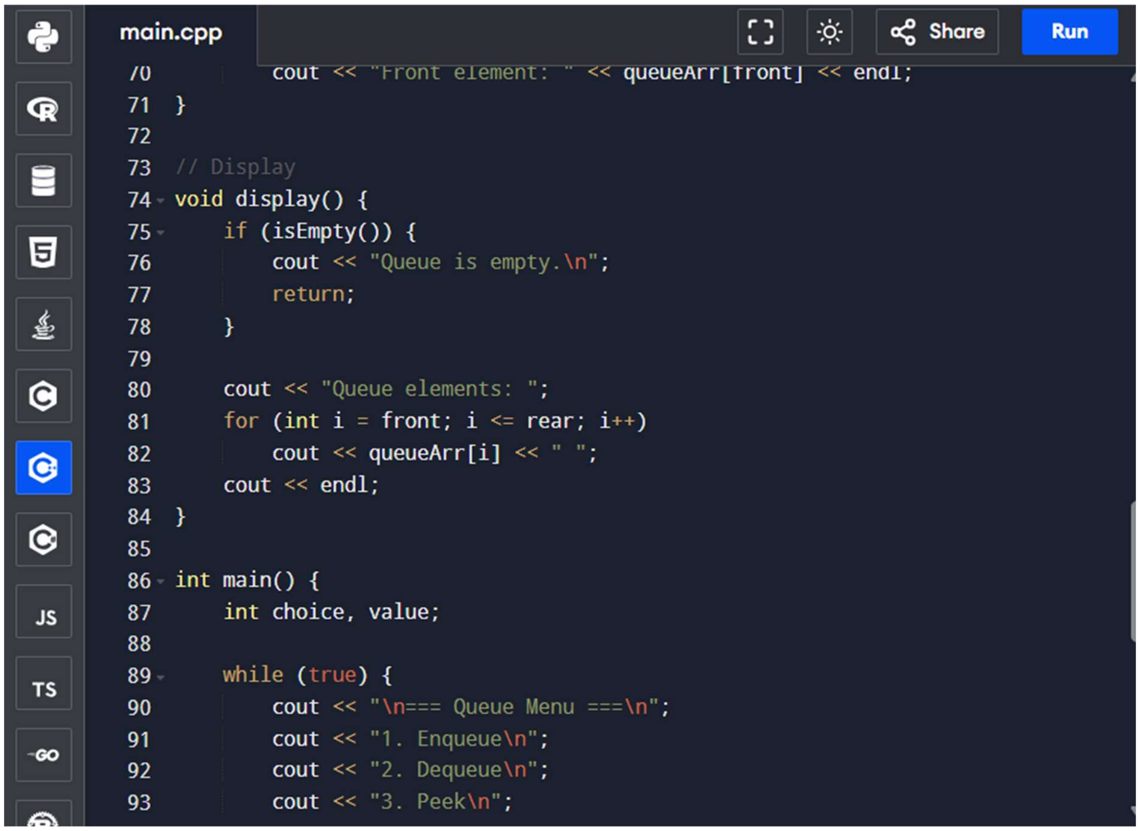
```cpp
#include <iostream>
using namespace std;

#define SIZE 5
int queueArr[SIZE];
int front = -1, rear = -1;

bool isEmpty() {
    return (front == -1 && rear == -1);
}

bool isFull() {
    return (rear == SIZE - 1);
}


bool existsInQueue(int value) {
    if (isEmpty()) return false;

    for (int i = front; i <= rear; i++) {
        if (queueArr[i] == value)
            return true;
    }
    return false;
```



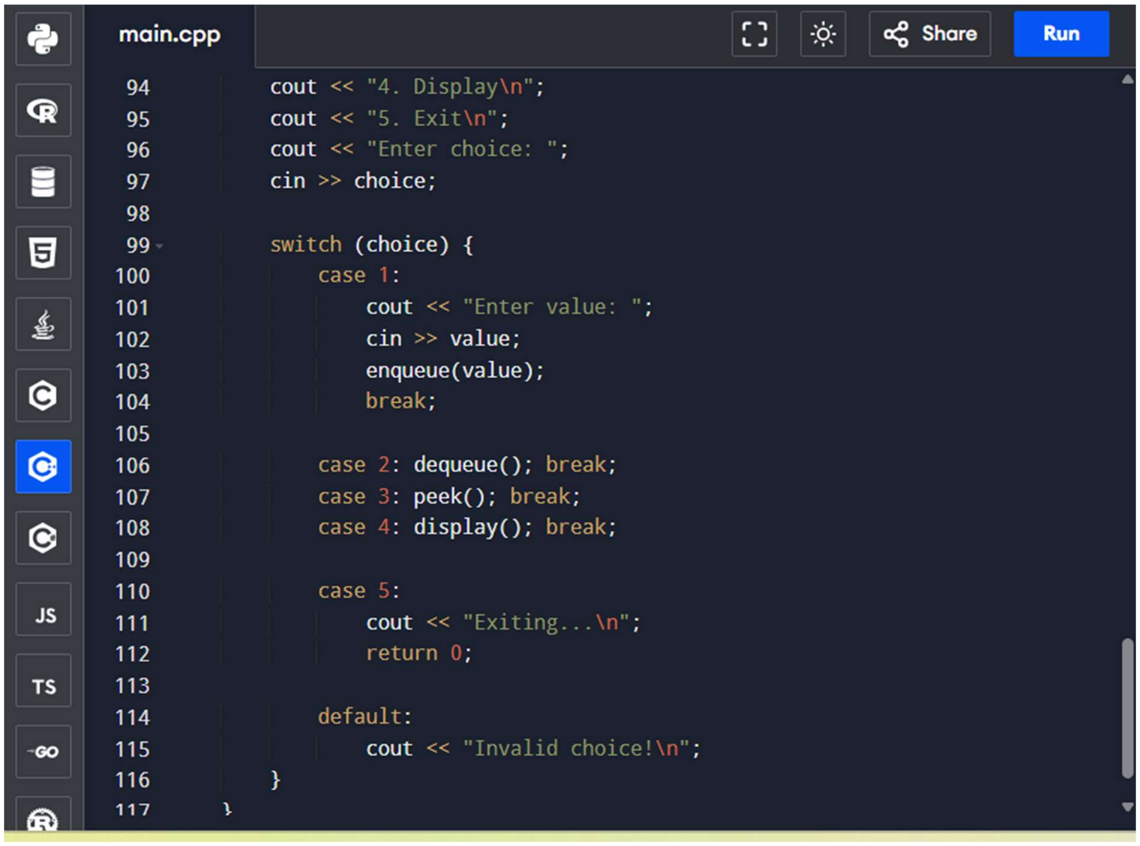
```cpp
}

// Modified enqueue with duplicate check
void enqueue(int value) {
    if (isFull()) {
        cout << "Queue Overflow! Cannot insert.\n";
        return;
    }

    if (existsInQueue(value)) {
        cout << "Duplicate value! Insertion not allowed.\n";
        return;
    }

    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear++;
    }

    queueArr[rear] = value;
    cout << value << " inserted into queue.\n";
}
```



```cpp
}

// Dequeue
void dequeue() {
    if (isEmpty()) {
        cout << "Queue Underflow! Nothing to delete.\n";
        return;
    }

    cout << queueArr[front] << " removed from queue.\n";

    if (front == rear) {
        front = rear = -1;
    } else {
        front++;
    }
}

// Peek
void peek() {
    if (isEmpty())
        cout << "Queue is empty.\n";
    else
        cout << "Front element: " << queueArr[front] << endl;
```

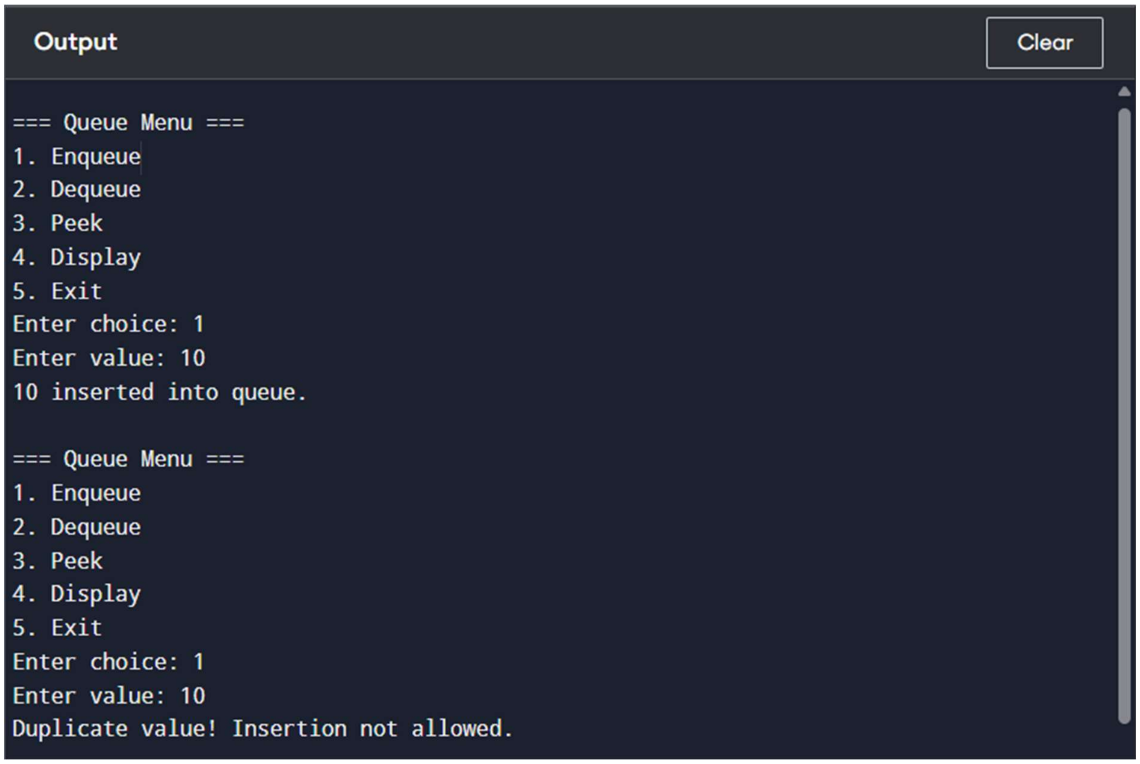


```cpp
        cout << "Front element: " << queueArr[front] << endl;
}

// Display
void display() {
    if (isEmpty()) {
        cout << "Queue is empty.\n";
        return;
    }

    cout << "Queue elements: ";
    for (int i = front; i <= rear; i++)
        cout << queueArr[i] << " ";
    cout << endl;
}

int main() {
    int choice, value;

    while (true) {
        cout << "\n=== Queue Menu ===\n";
        cout << "1. Enqueue\n";
        cout << "2. Dequeue\n";
        cout << "3. Peek\n";
```

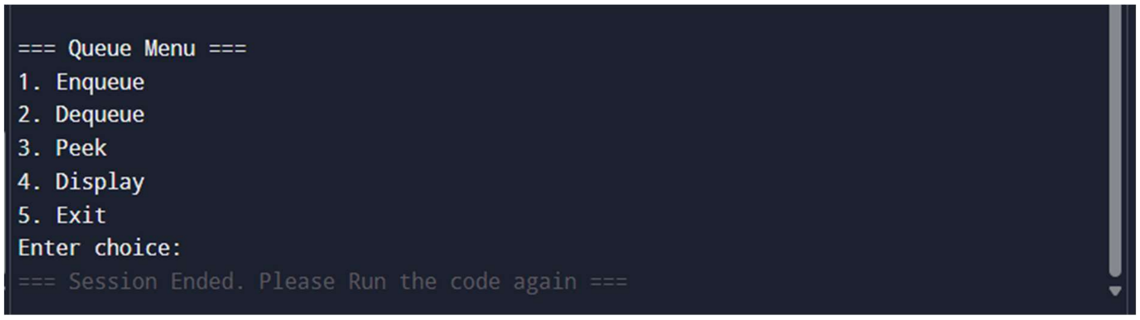


```cpp
        cout << "4. Display\n";
        cout << "5. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value: ";
                cin >> value;
                enqueue(value);
                break;

            case 2: dequeue(); break;
            case 3: peek(); break;
            case 4: display(); break;

            case 5:
                cout << "Exiting...\n";
                return 0;

            default:
                cout << "Invalid choice!\n";
        }
    }
```

CODE OUTPUT:



```
=== Queue Menu ===
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value: 10
10 inserted into queue.

=== Queue Menu ===
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value: 10
Duplicate value! Insertion not allowed.
```


```
=== Queue Menu ===
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter choice:
=== Session Ended. Please Run the code again ===
```

LAB TASK 2:

2. Write a program using a queue (array) that adds an additional function:

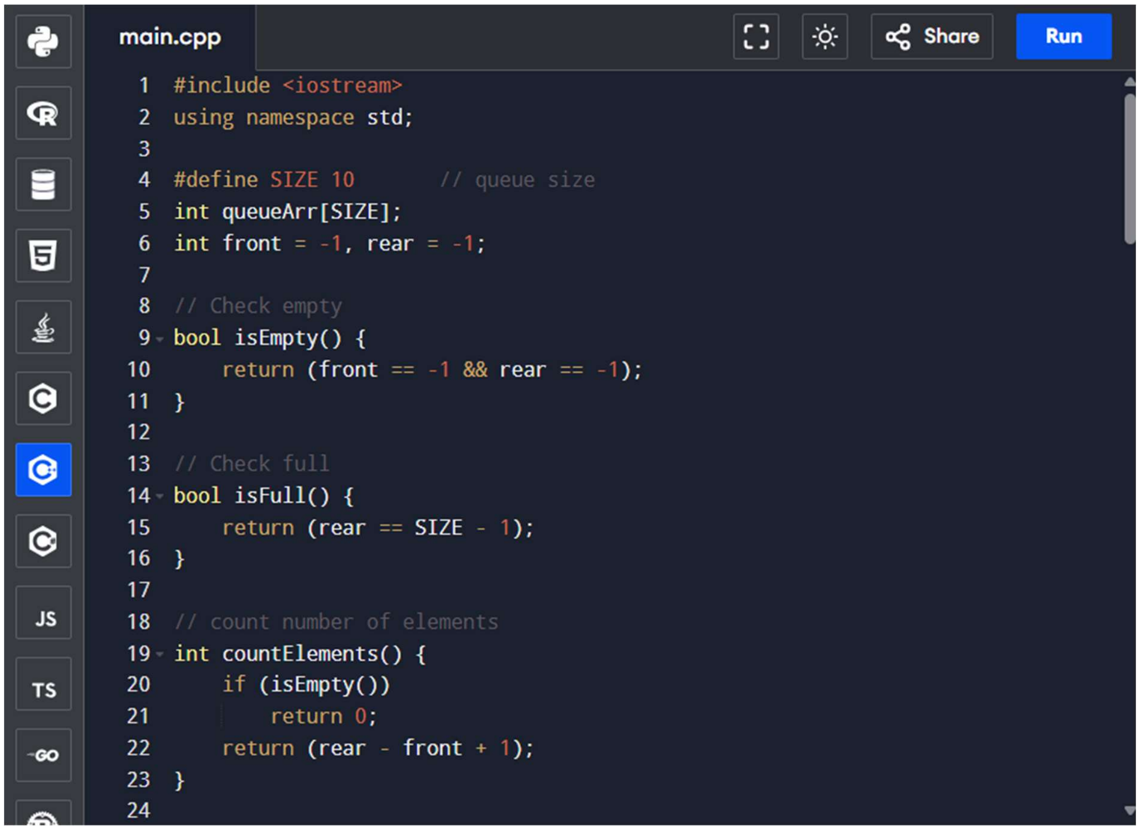
countElements()

This function returns the number of elements currently in the queue.

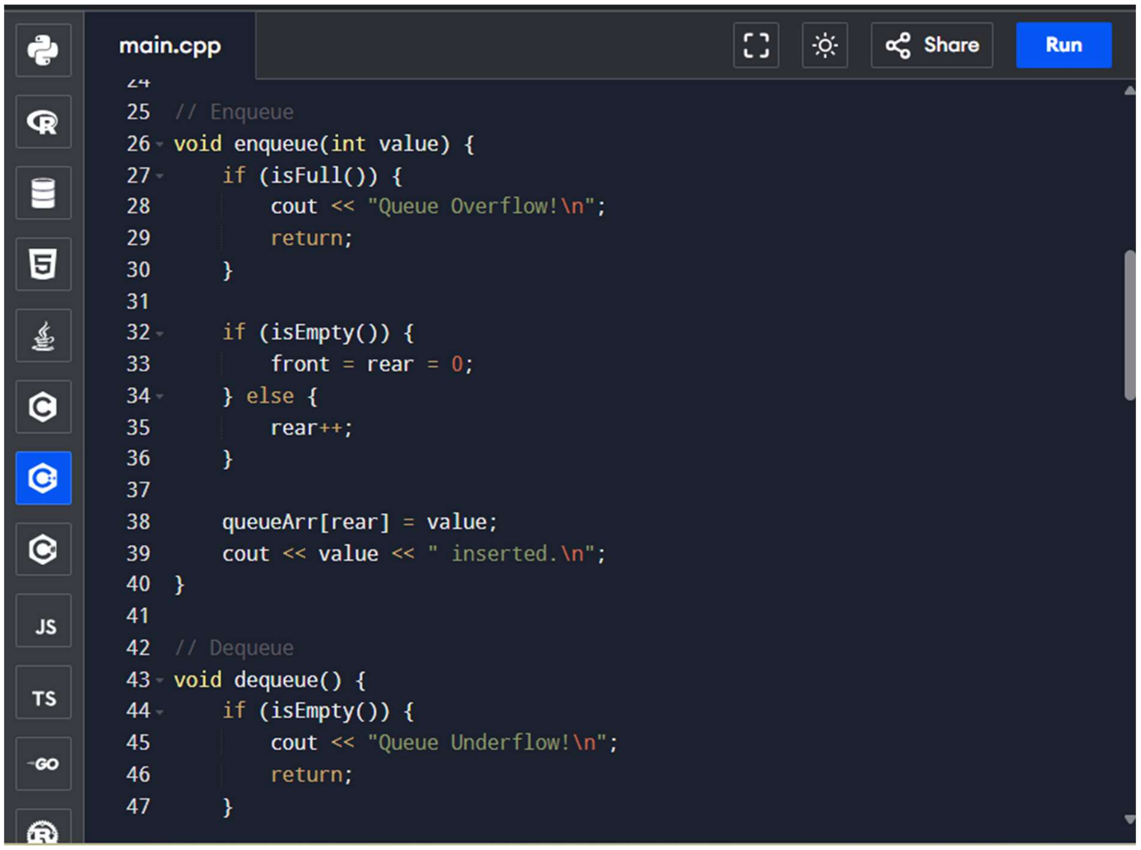
Algorithm for countElements()

Algorithm countElements():

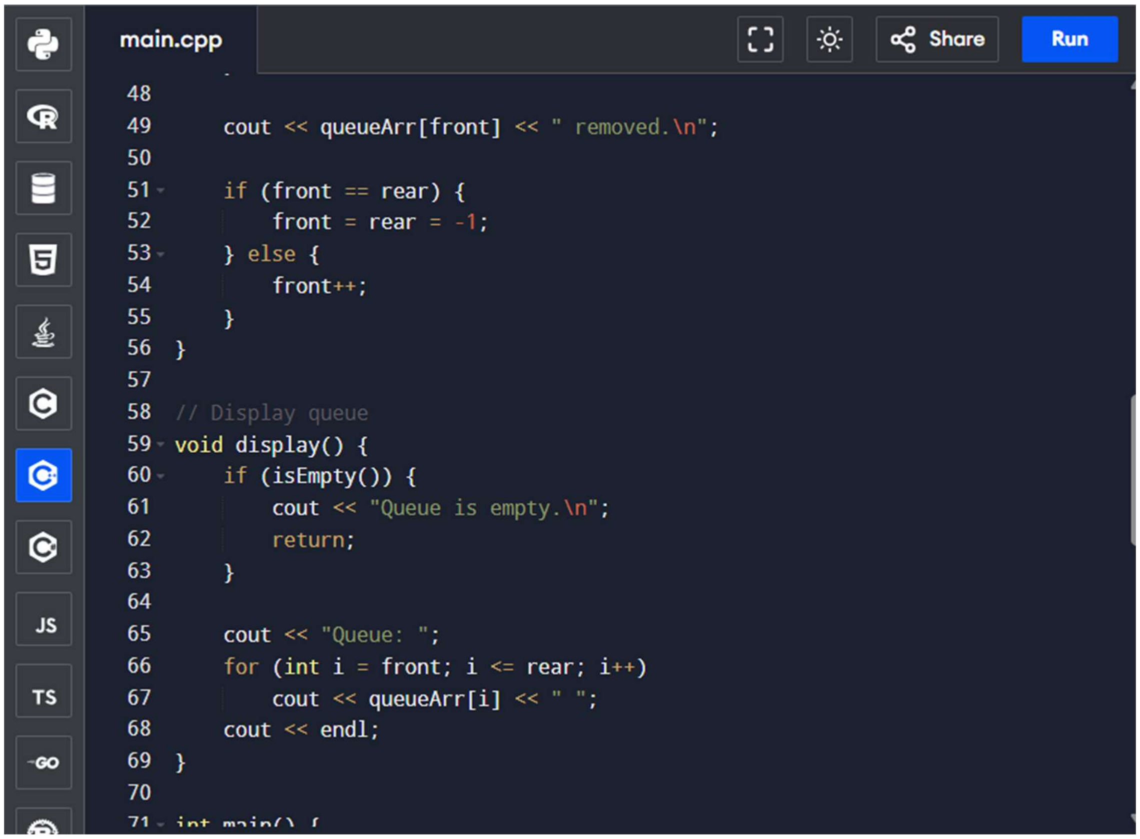
```
1. If queue is empty then
       return 0

2. Else
       count ← (REAR - FRONT + 1)
       return count
```



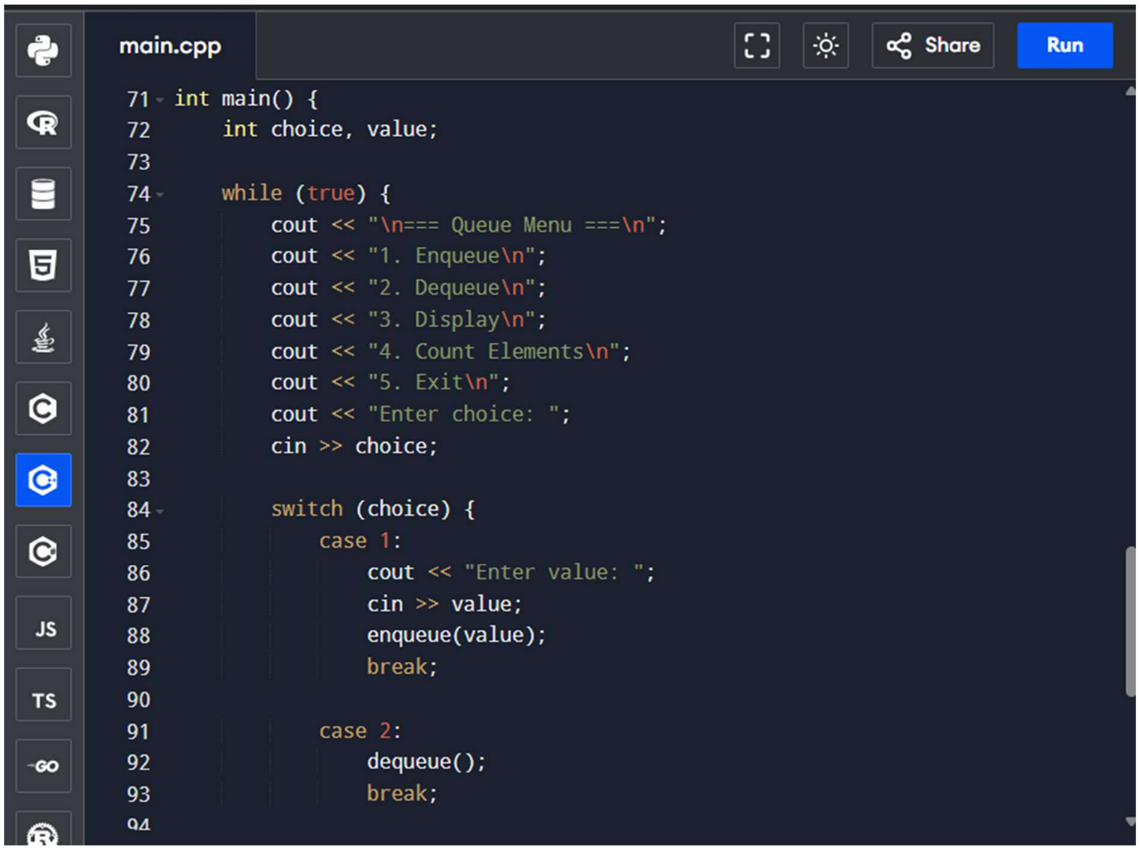
```cpp
#include <iostream>
using namespace std;

#define SIZE 10       // queue size
int queueArr[SIZE];
int front = -1, rear = -1;

// Check empty
bool isEmpty() {
    return (front == -1 && rear == -1);
}

// Check full
bool isFull() {
    return (rear == SIZE - 1);
}

// count number of elements
int countElements() {
    if (isEmpty())
        return 0;
    return (rear - front + 1);
}

```



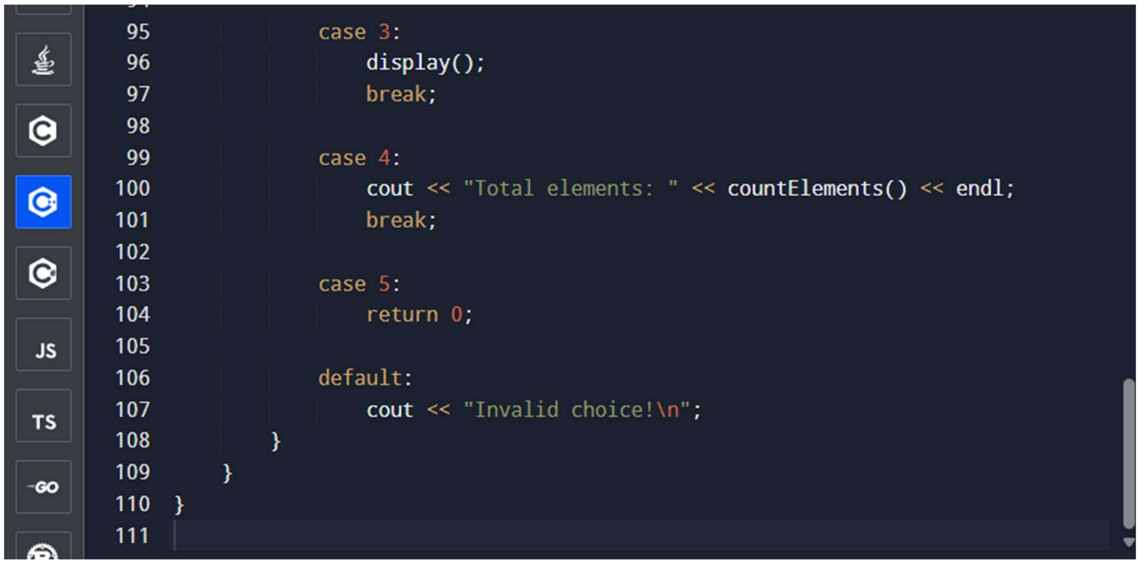
```cpp
// Enqueue
void enqueue(int value) {
    if (isFull()) {
        cout << "Queue Overflow!\n";
        return;
    }

    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear++;
    }

    queueArr[rear] = value;
    cout << value << " inserted.\n";
}

// Dequeue
void dequeue() {
    if (isEmpty()) {
        cout << "Queue Underflow!\n";
        return;
    }
```



```
    cout << queueArr[front] << " removed.\n";

    if (front == rear) {
        front = rear = -1;
    } else {
        front++;
    }
}

// Display queue
void display() {
    if (isEmpty()) {
        cout << "Queue is empty.\n";
        return;
    }

    cout << "Queue: ";
    for (int i = front; i <= rear; i++)
        cout << queueArr[i] << " ";
    cout << endl;
}

```

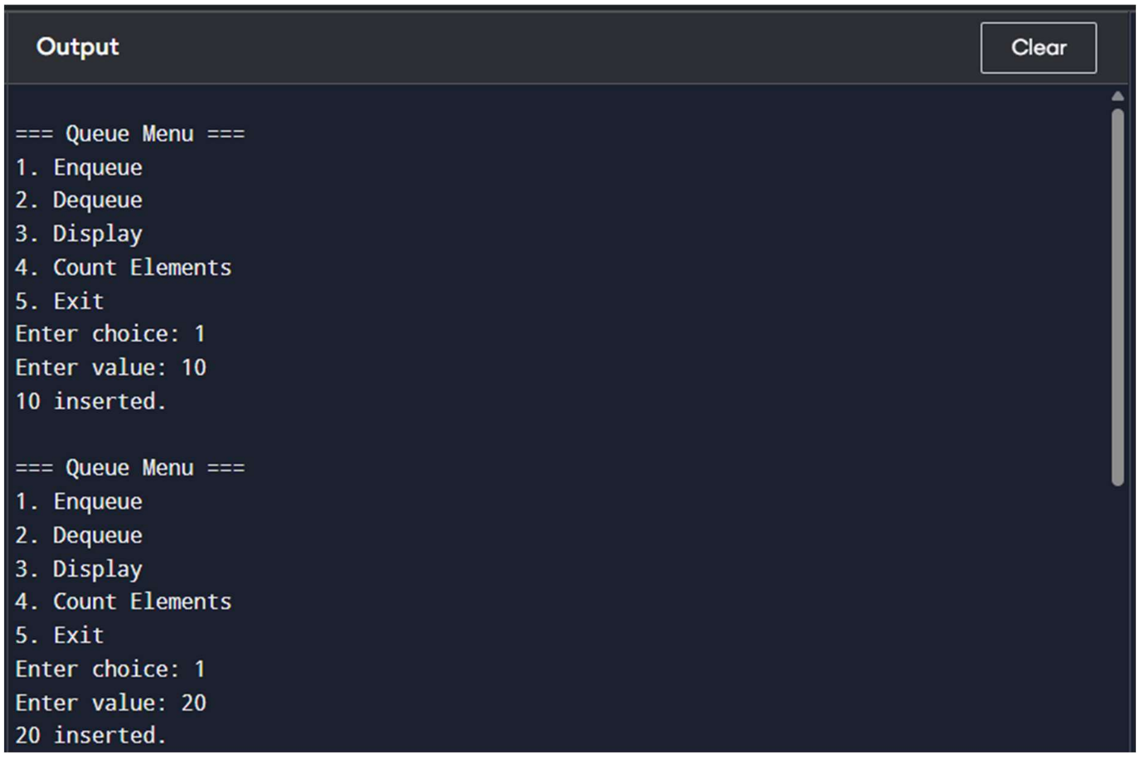


```
int main() {
    int choice, value;

    while (true) {
        cout << "\n=== Queue Menu ===\n";
        cout << "1. Enqueue\n";
        cout << "2. Dequeue\n";
        cout << "3. Display\n";
        cout << "4. Count Elements\n";
        cout << "5. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value: ";
                cin >> value;
                enqueue(value);
                break;

            case 2:
                dequeue();
                break;
```

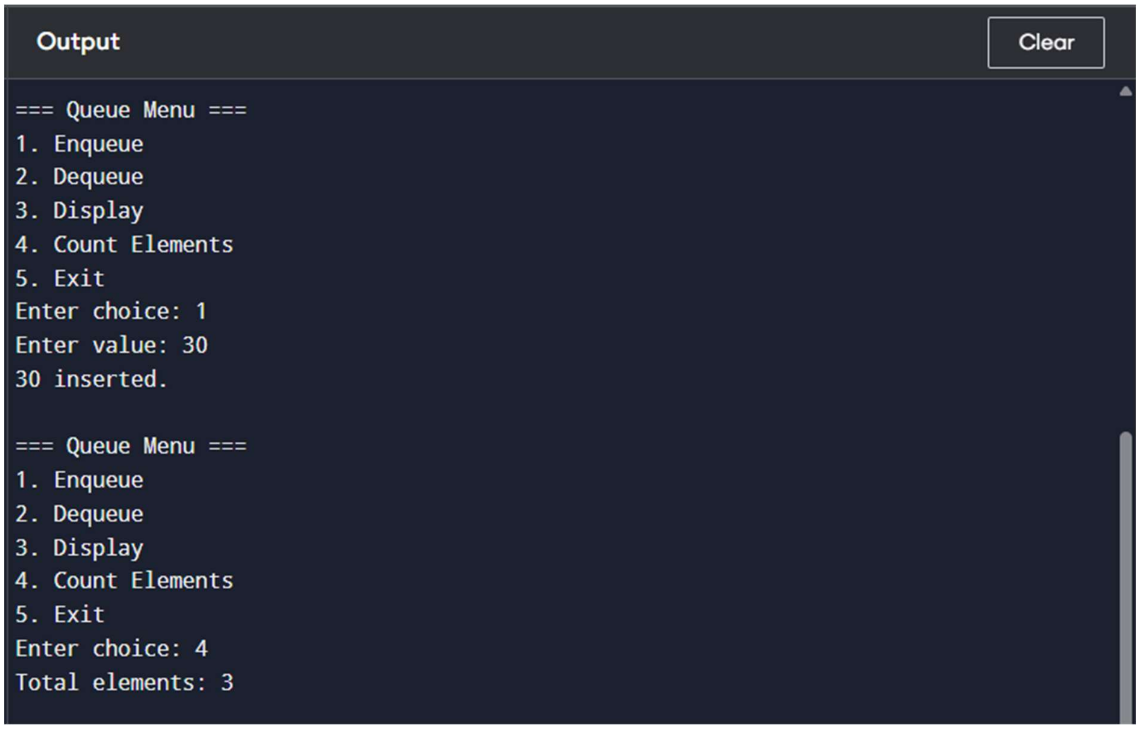


```
            case 3:
                display();
                break;

            case 4:
                cout << "Total elements: " << countElements() << endl;
                break;

            case 5:
                return 0;

            default:
                cout << "Invalid choice!\n";
        }
    }
}

```

Code output:



```
=== Queue Menu ===
1. Enqueue
2. Dequeue
3. Display
4. Count Elements
5. Exit
Enter choice: 1
Enter value: 10
10 inserted.

=== Queue Menu ===
1. Enqueue
2. Dequeue
3. Display
4. Count Elements
5. Exit
Enter choice: 1
Enter value: 20
20 inserted.
```



```
=== Queue Menu ===
1. Enqueue
2. Dequeue
3. Display
4. Count Elements
5. Exit
Enter choice: 1
Enter value: 30
30 inserted.

=== Queue Menu ===
1. Enqueue
2. Dequeue
3. Display
4. Count Elements
5. Exit
Enter choice: 4
Total elements: 3
```

Lab task 3:

3. Write a program to:

- Insert values into a queue using an array
- Reverse the queue without using any library functions
- Display the reversed queue

Algorithm for Reversing a Queue (Using Array Only):

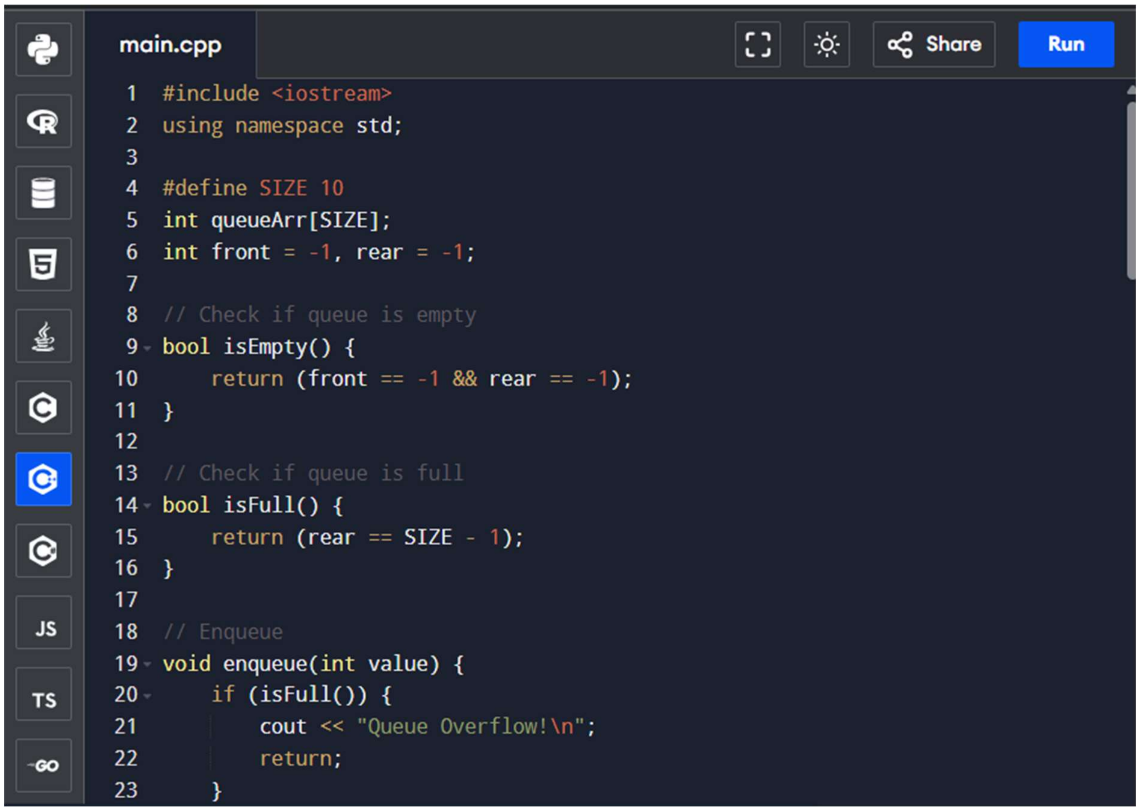
```
1. If queue is empty
    Print "Queue is empty"
    Exit

2. Create a temporary array temp[]

3. Set index = 0

4. For i ← REAR down to FRONT do
    temp[index] ← queueArr[i]
```

    index ← index + 1

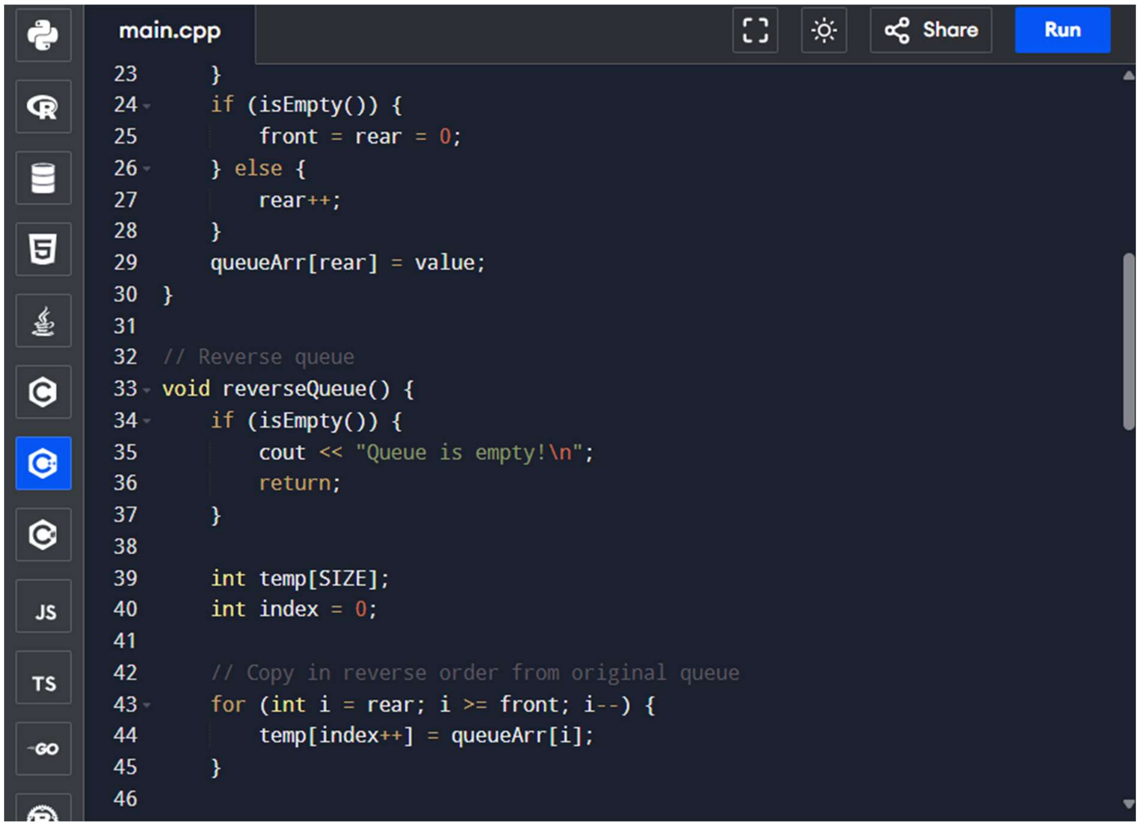
5. Now copy temp[] back to queueArr[] starting from FRONT

6. Set REAR = FRONT + (index - 1)

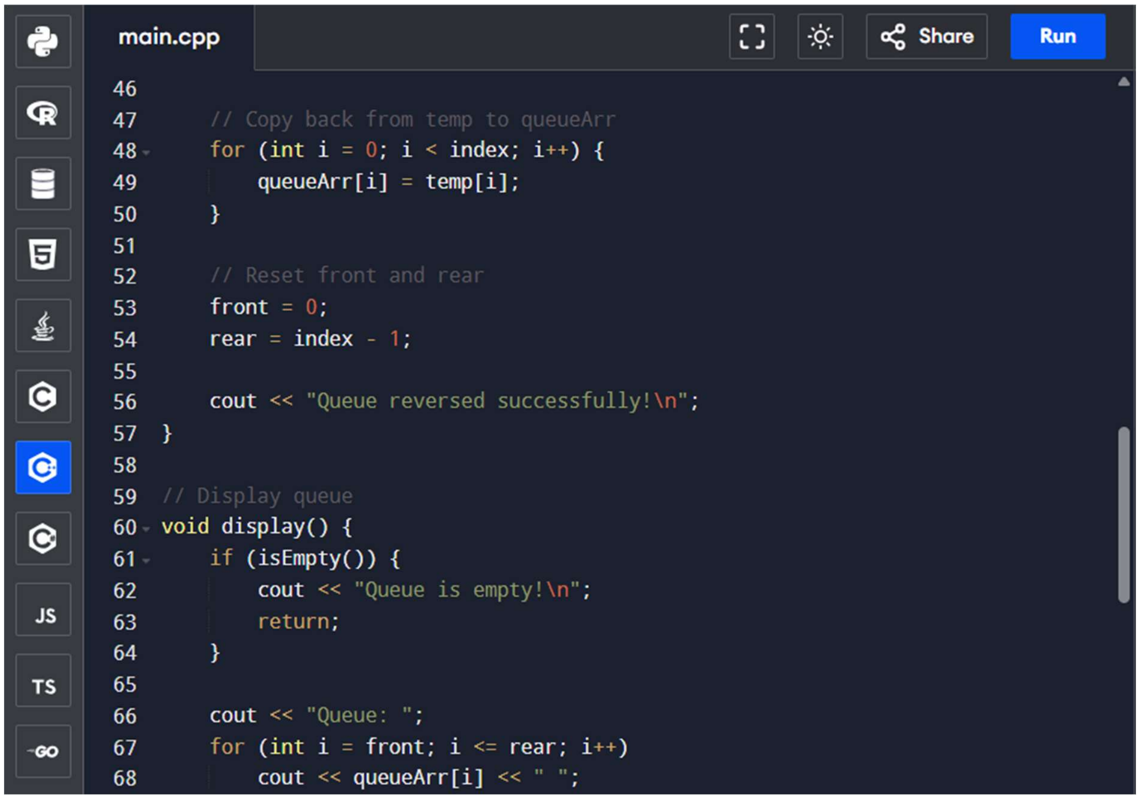
7. Print "Queue reversed successfully"



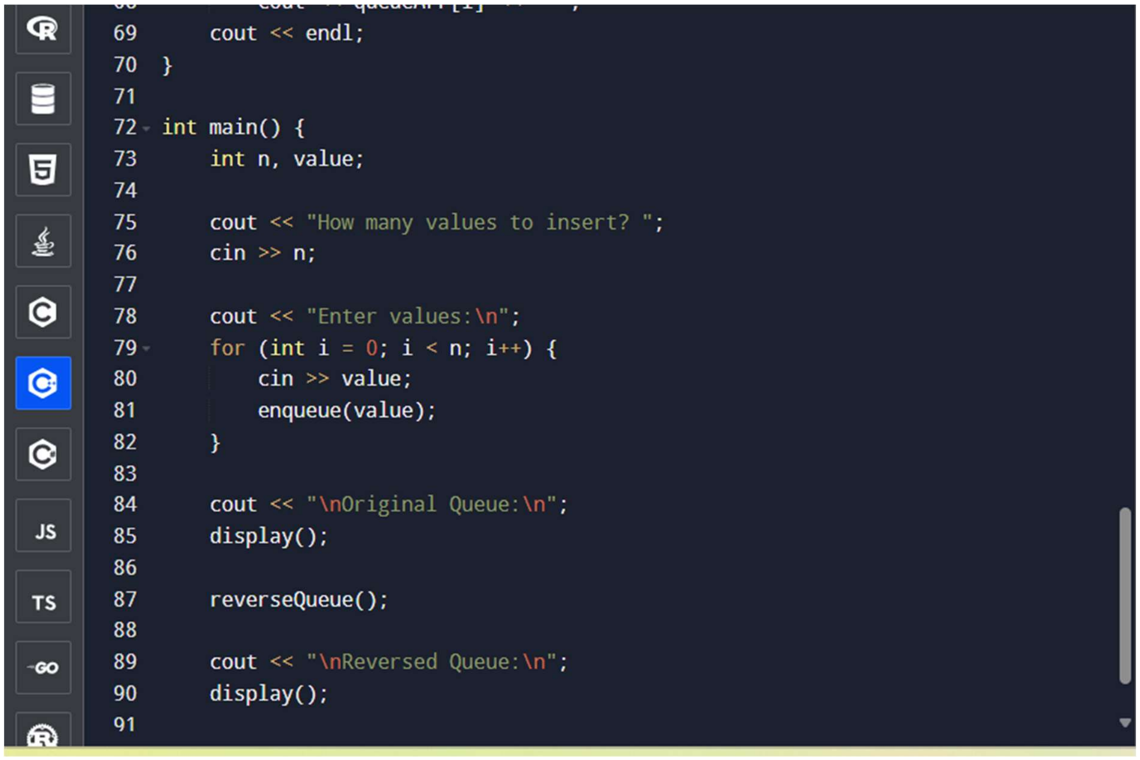
```cpp
#include <iostream>
using namespace std;

#define SIZE 10
int queueArr[SIZE];
int front = -1, rear = -1;

// Check if queue is empty
bool isEmpty() {
    return (front == -1 && rear == -1);
}

// Check if queue is full
bool isFull() {
    return (rear == SIZE - 1);
}

// Enqueue
void enqueue(int value) {
    if (isFull()) {
        cout << "Queue Overflow!\n";
        return;
    }
```



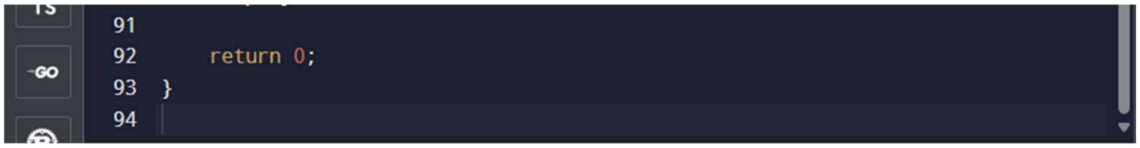
```
    }
    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear++;
    }
    queueArr[rear] = value;
}

// Reverse queue
void reverseQueue() {
    if (isEmpty()) {
        cout << "Queue is empty!\n";
        return;
    }

    int temp[SIZE];
    int index = 0;

    // Copy in reverse order from original queue
    for (int i = rear; i >= front; i--) {
        temp[index++] = queueArr[i];
    }

```



```

    // Copy back from temp to queueArr
    for (int i = 0; i < index; i++) {
        queueArr[i] = temp[i];
    }

    // Reset front and rear
    front = 0;
    rear = index - 1;

    cout << "Queue reversed successfully!\n";
}

// Display queue
void display() {
    if (isEmpty()) {
        cout << "Queue is empty!\n";
        return;
    }

    cout << "Queue: ";
    for (int i = front; i <= rear; i++)
        cout << queueArr[i] << " ";
```

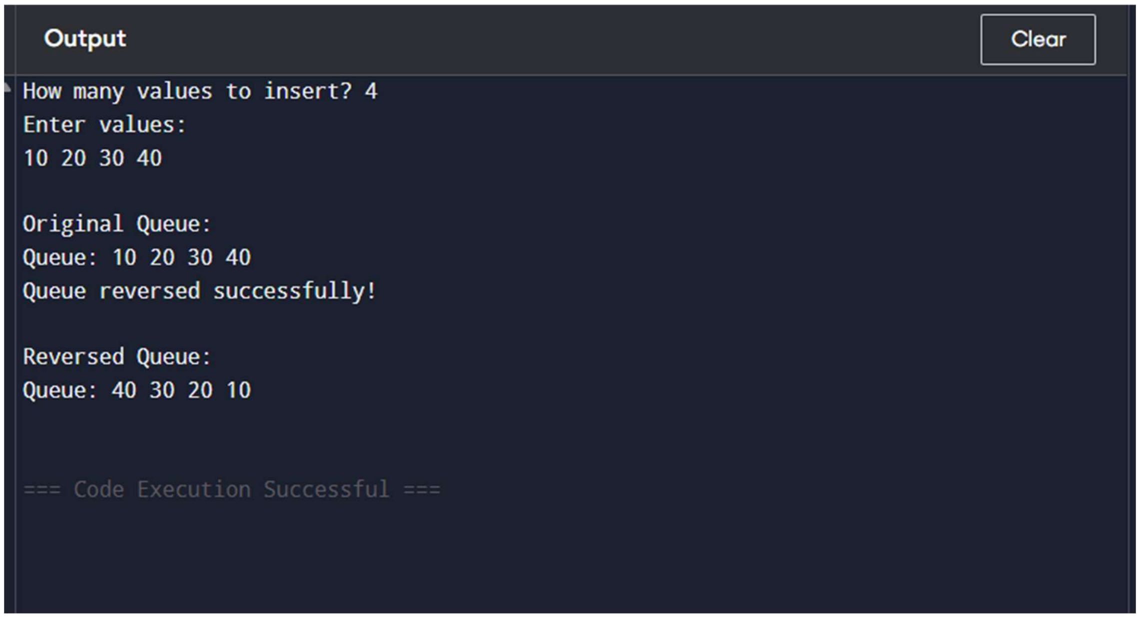


```
    cout << endl;
}

int main() {
    int n, value;

    cout << "How many values to insert? ";
    cin >> n;

    cout << "Enter values:\n";
    for (int i = 0; i < n; i++) {
        cin >> value;
        enqueue(value);
    }

    cout << "\nOriginal Queue:\n";
    display();

    reverseQueue();

    cout << "\nReversed Queue:\n";
    display();

```



```

    return 0;
}

```

Code output:



```
How many values to insert? 4
Enter values:
10 20 30 40

Original Queue:
Queue: 10 20 30 40
Queue reversed successfully!

Reversed Queue:
Queue: 40 30 20 10


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

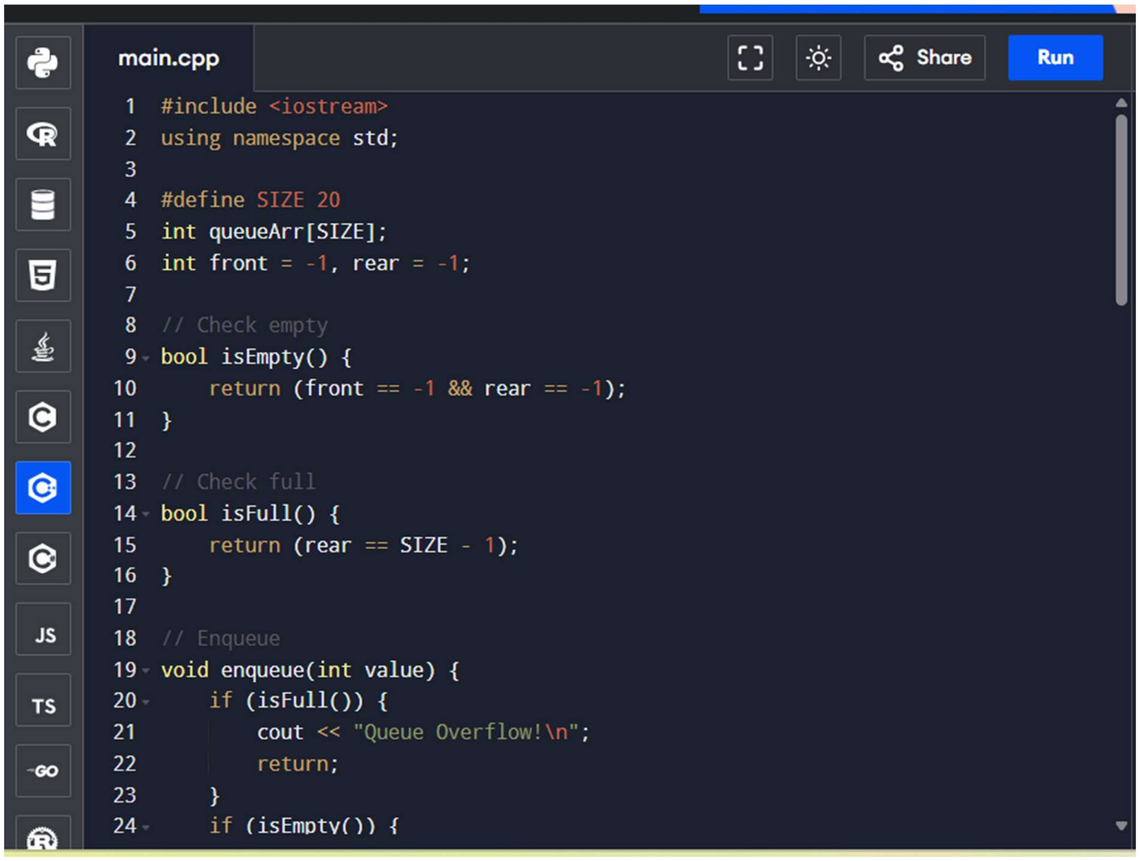
Lab task 4:

4. Write a program to sort the elements of a queue.

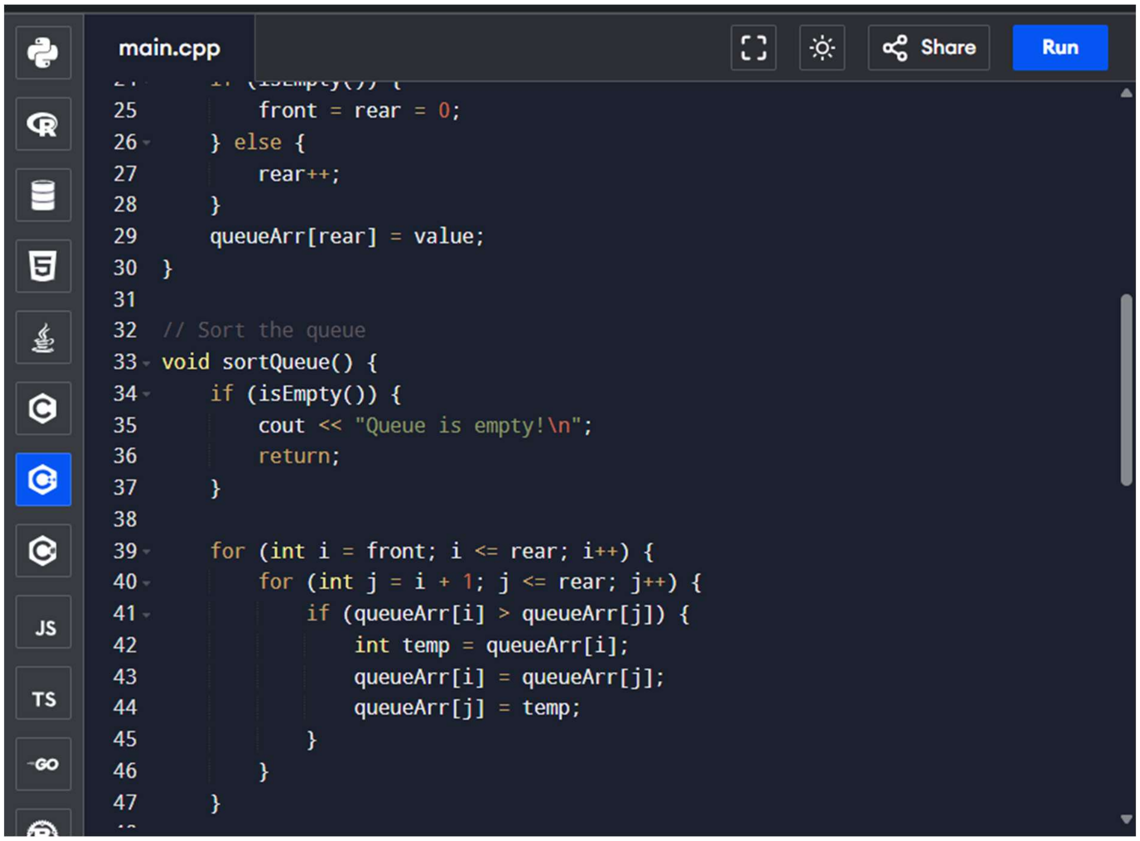
Algorithm to Sort Queue Using Array:

```
Algorithm sortQueue():


1. If queue is empty:
    Print "Queue is empty"
    Exit


2. For i ← FRONT to REAR:
    For j ← i+1 to REAR:
        If queueArr[i] > queueArr[j]:
            Swap queueArr[i] and queueArr[j]


3. Print "Queue sorted successfully"
```

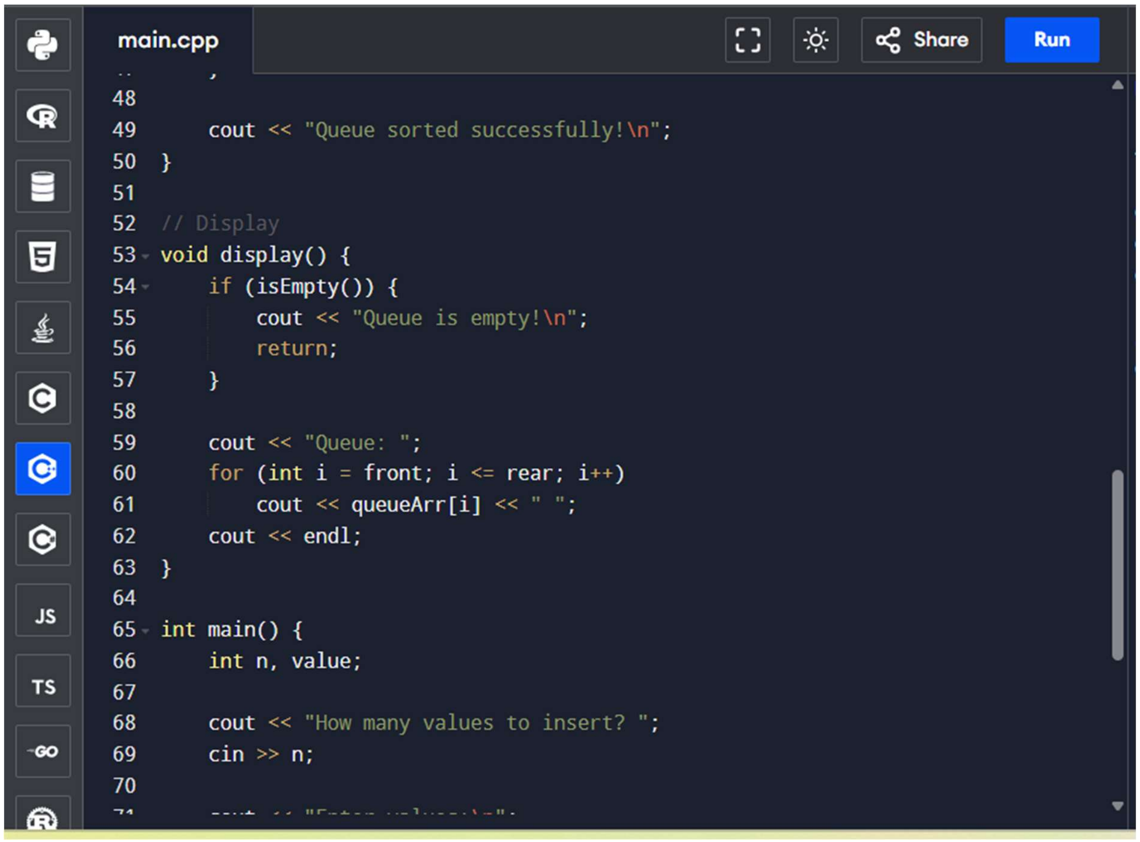
Code:



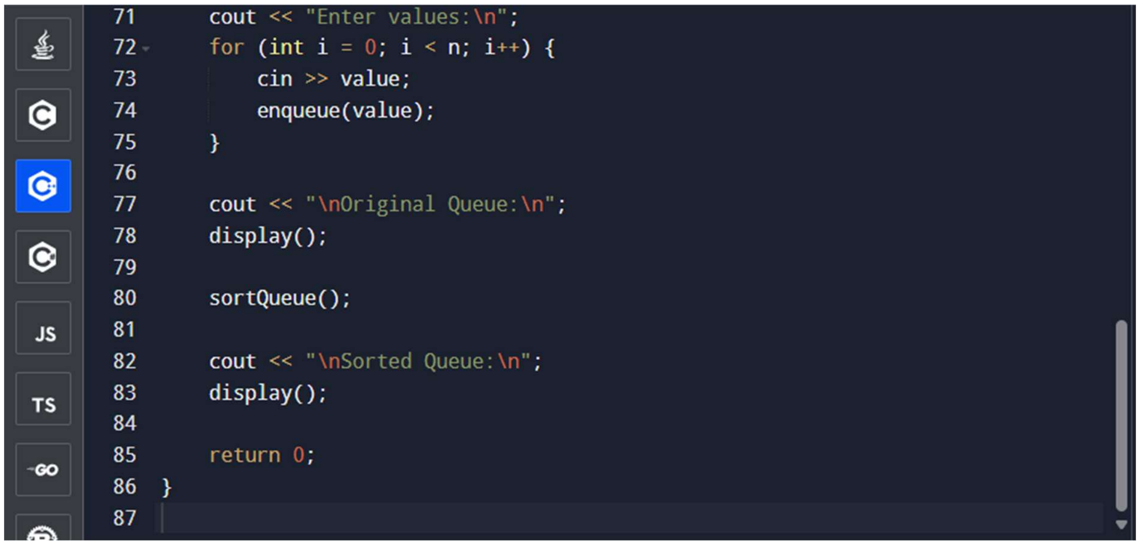
```cpp
#include <iostream>
using namespace std;

#define SIZE 20
int queueArr[SIZE];
int front = -1, rear = -1;

// Check empty
bool isEmpty() {
    return (front == -1 && rear == -1);
}

// Check full
bool isFull() {
    return (rear == SIZE - 1);
}

// Enqueue
void enqueue(int value) {
    if (isFull()) {
        cout << "Queue Overflow!\n";
        return;
    }
    if (isEmpty()) {
```



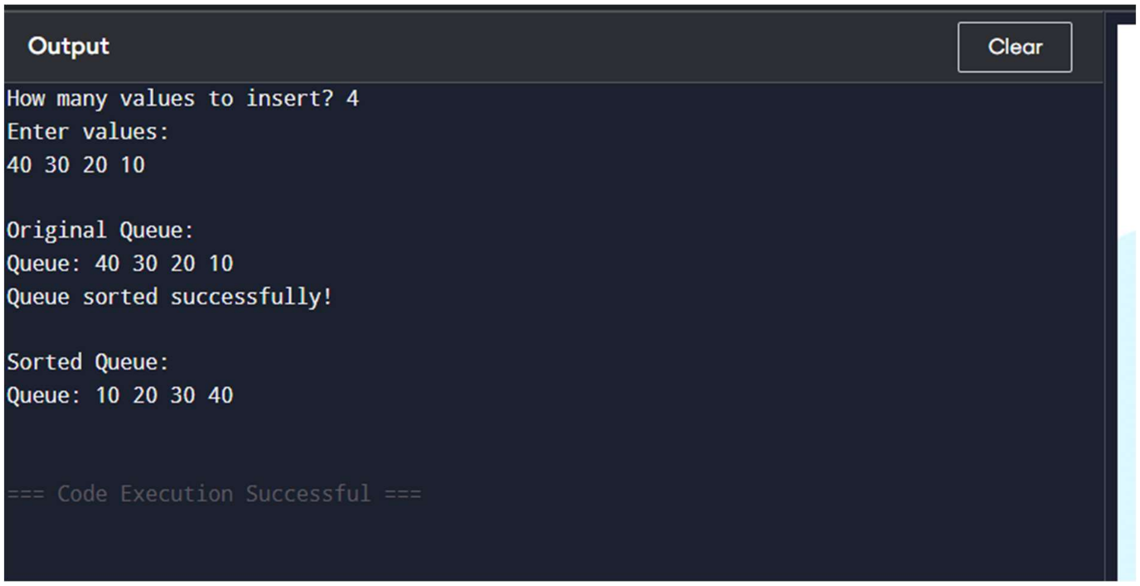
```cpp
        front = rear = 0;
    } else {
        rear++;
    }
    queueArr[rear] = value;
}

// Sort the queue
void sortQueue() {
    if (isEmpty()) {
        cout << "Queue is empty!\n";
        return;
    }

    for (int i = front; i <= rear; i++) {
        for (int j = i + 1; j <= rear; j++) {
            if (queueArr[i] > queueArr[j]) {
                int temp = queueArr[i];
                queueArr[i] = queueArr[j];
                queueArr[j] = temp;
            }
        }
    }
```



```cpp

    cout << "Queue sorted successfully!\n";
}

// Display
void display() {
    if (isEmpty()) {
        cout << "Queue is empty!\n";
        return;
    }

    cout << "Queue: ";
    for (int i = front; i <= rear; i++)
        cout << queueArr[i] << " ";
    cout << endl;
}

int main() {
    int n, value;

    cout << "How many values to insert? ";
    cin >> n;

```



```cpp
    cout << "Enter values:\n";
    for (int i = 0; i < n; i++) {
        cin >> value;
        enqueue(value);
    }

    cout << "\nOriginal Queue:\n";
    display();

    sortQueue();

    cout << "\nSorted Queue:\n";
    display();

    return 0;
}

```



```
How many values to insert? 4
Enter values:
40 30 20 10

Original Queue:
Queue: 40 30 20 10
Queue sorted successfully!

Sorted Queue:
Queue: 10 20 30 40


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

Lab task 5:

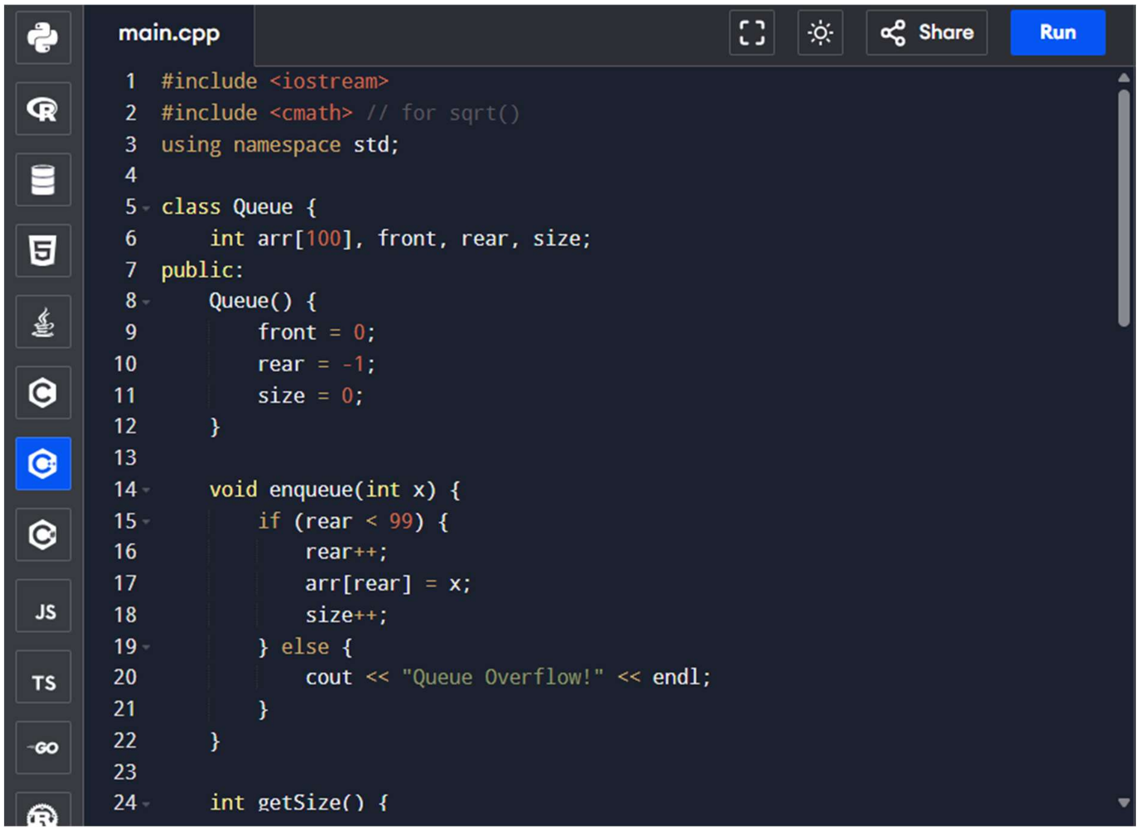
5. Write a program to find the mean, variance and standard deviation of all elements of a Queue

Algorithm:

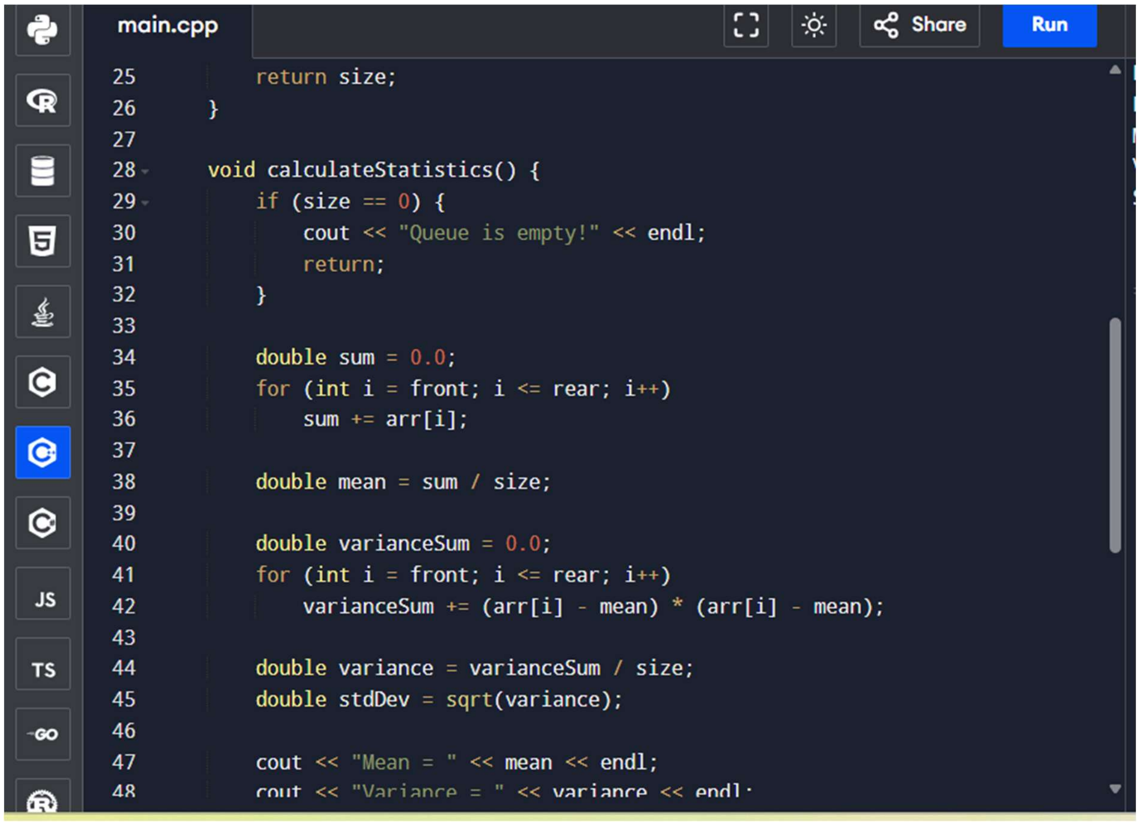
1. Initialize Queue:
    - Create a queue using an array.
    - Keep track of front and rear indices.
2. Input Elements:
    - Insert elements into the queue until the user stops.
3. Calculate Mean:
    - Sum all elements in the queue.
    - Divide the sum by the number of elements.
4. Calculate Variance:
    - For each element, calculate the squared difference from the mean.
    - Sum all squared differences.
    - Divide by the number of elements.
5. Calculate Standard Deviation:
    - Take the square root of the variance.

6. Display Results.

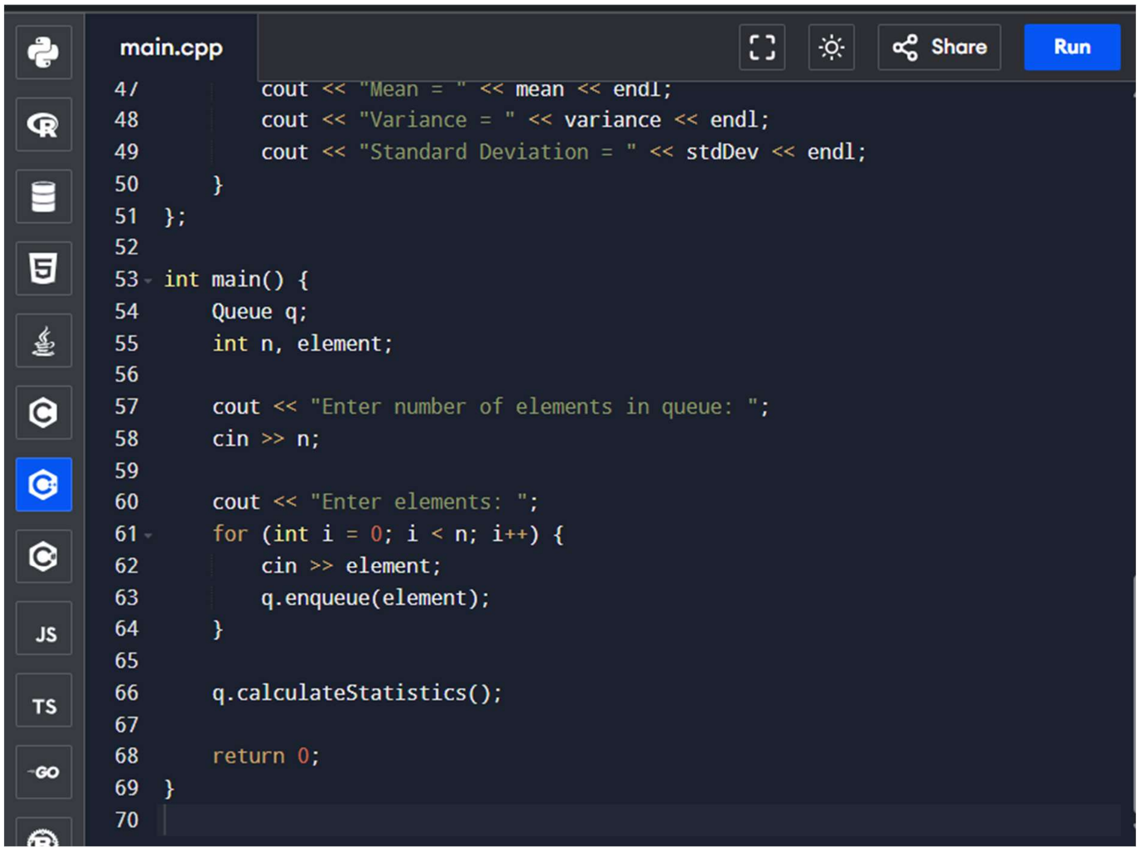
code:



```cpp
#include <iostream>
#include <cmath> // for sqrt()
using namespace std;

class Queue {
    int arr[100], front, rear, size;
public:
    Queue() {
        front = 0;
        rear = -1;
        size = 0;
    }

    void enqueue(int x) {
        if (rear < 99) {
            rear++;
            arr[rear] = x;
            size++;
        } else {
            cout << "Queue Overflow!" << endl;
        }
    }

    int getSize() {
```

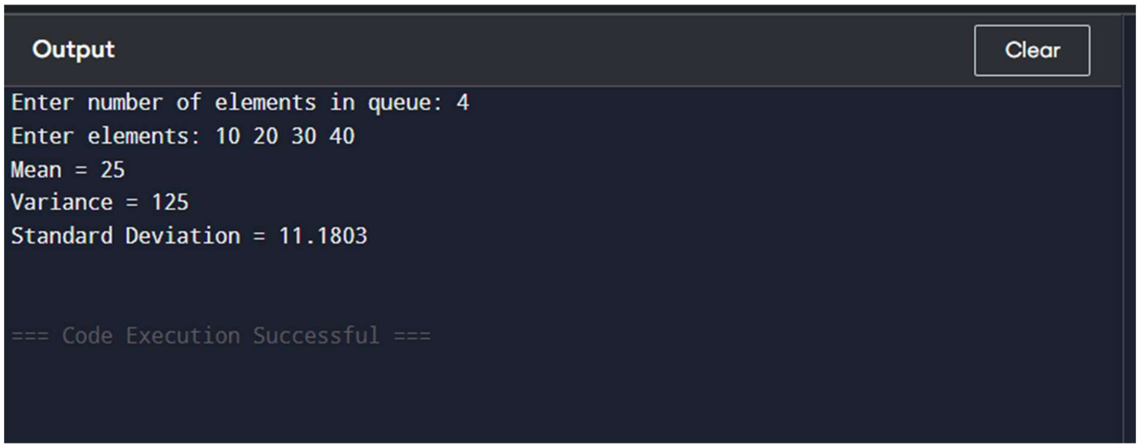

main.cpp

```
            return size;
        }

        void calculateStatistics() {
            if (size == 0) {
                cout << "Queue is empty!" << endl;
                return;
            }

            double sum = 0.0;
            for (int i = front; i <= rear; i++)
                sum += arr[i];

            double mean = sum / size;

            double varianceSum = 0.0;
            for (int i = front; i <= rear; i++)
                varianceSum += (arr[i] - mean) * (arr[i] - mean);

            double variance = varianceSum / size;
            double stdDev = sqrt(variance);

            cout << "Mean = " << mean << endl;
            cout << "Variance = " << variance << endl;
```


main.cpp

```
            cout << "Mean = " << mean << endl;
            cout << "Variance = " << variance << endl;
            cout << "Standard Deviation = " << stdDev << endl;
        }
    };

    int main() {
        Queue q;
        int n, element;

        cout << "Enter number of elements in queue: ";
        cin >> n;

        cout << "Enter elements: ";
        for (int i = 0; i < n; i++) {
            cin >> element;
            q.enqueue(element);
        }

        q.calculateStatistics();

        return 0;
    }
```

Code output:



```
Enter number of elements in queue: 4
Enter elements: 10 20 30 40
Mean = 25
Variance = 125
Standard Deviation = 11.1803


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