

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

The screenshot shows a code editor interface with two panes. The top pane displays the beginning of a C++ file named `main.cpp`. The bottom pane displays the continuation of the same file, starting from line 25. The code implements a queue using an array and pointers for front and rear. It includes functions for checking if the queue is empty or full, and for enqueueing values while avoiding duplicates.

```
main.cpp

1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 5
5 int queueArr[SIZE];
6 int front = -1, rear = -1;
7
8 bool isEmpty() {
9     return (front == -1 && rear == -1);
10 }
11
12 bool isFull() {
13     return (rear == SIZE - 1);
14 }
15
16
17 bool existsInQueue(int value) {
18     if (isEmpty()) return false;
19
20     for (int i = front; i <= rear; i++) {
21         if (queueArr[i] == value)
22             return true;
23     }
24     return false;
25 }
26
27 // Modified enqueue with duplicate check
28 void enqueue(int value) {
29     if (isFull()) {
30         cout << "Queue Overflow! Cannot insert.\n";
31         return;
32     }
33
34     if (existsInQueue(value)) {
35         cout << "Duplicate value! Insertion not allowed.\n";
36         return;
37     }
38
39     if (isEmpty()) {
40         front = rear = 0;
41     } else {
42         rear++;
43     }
44
45     queueArr[rear] = value;
46     cout << value << " inserted into queue.\n";
47 }
```

```
main.cpp
47 }
48
49 // Dequeue
50 void dequeue() {
51     if (isEmpty()) {
52         cout << "Queue Underflow! Nothing to delete.\n";
53         return;
54     }
55
56     cout << queueArr[front] << " removed from queue.\n";
57
58     if (front == rear) {
59         front = rear = -1;
60     } else {
61         front++;
62     }
63 }
64
65 // Peek
66 void peek() {
67     if (isEmpty())
68         cout << "Queue is empty.\n";
69     else
70         cout << "Front element: " << queueArr[front] << endl;
```

```
main.cpp
70         cout << "Front element: " << queueArr[front] << endl;
71 }
72
73 // Display
74 void display() {
75     if (isEmpty()) {
76         cout << "Queue is empty.\n";
77         return;
78     }
79
80     cout << "Queue elements: ";
81     for (int i = front; i <= rear; i++)
82         cout << queueArr[i] << " ";
83     cout << endl;
84 }
85
86 int main() {
87     int choice, value;
88
89     while (true) {
90         cout << "\n== Queue Menu ==\n";
91         cout << "1. Enqueue\n";
92         cout << "2. Dequeue\n";
93         cout << "3. Peek\n";
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar containing icons for various programming languages: Python, R, SQL, Git, Java, C, C++, JavaScript, TypeScript, Go, and R. The main area displays a C++ file named `main.cpp`. The code implements a menu system for a queue. It includes functions for enqueueing, dequeuing, peeking, displaying, and exiting. The code uses `cout` for output and `cin` for input. The `enqueue` function is shown, along with other menu options.

```
94     cout << "4. Display\n";
95     cout << "5. Exit\n";
96     cout << "Enter choice: ";
97     cin >> choice;
98
99     switch (choice) {
100         case 1:
101             cout << "Enter value: ";
102             cin >> value;
103             enqueue(value);
104             break;
105
106         case 2: dequeue(); break;
107         case 3: peek(); break;
108         case 4: display(); break;
109
110         case 5:
111             cout << "Exiting...\n";
112             return 0;
113
114     default:
115         cout << "Invalid choice!\n";
116     }
117 }
```

CODE OUTPUT:

The output panel shows two separate runs of the program. Each run starts with the queue menu, followed by user input for enqueueing the value 10. In the first run, the output is "10 inserted into queue.". In the second run, the output is "Duplicate value! Insertion not allowed.", indicating that the enqueue operation failed because the value already exists in the queue.

```
Output
Clear

==== 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
```

main.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 10      // queue size
5 int queueArr[SIZE];
6 int front = -1, rear = -1;
7
8 // Check empty
9 bool isEmpty() {
10     return (front == -1 && rear == -1);
11 }
12
13 // Check full
14 bool isFull() {
15     return (rear == SIZE - 1);
16 }
17
18 // count number of elements
19 int countElements() {
20     if (isEmpty())
21         return 0;
22     return (rear - front + 1);
23 }
24
```

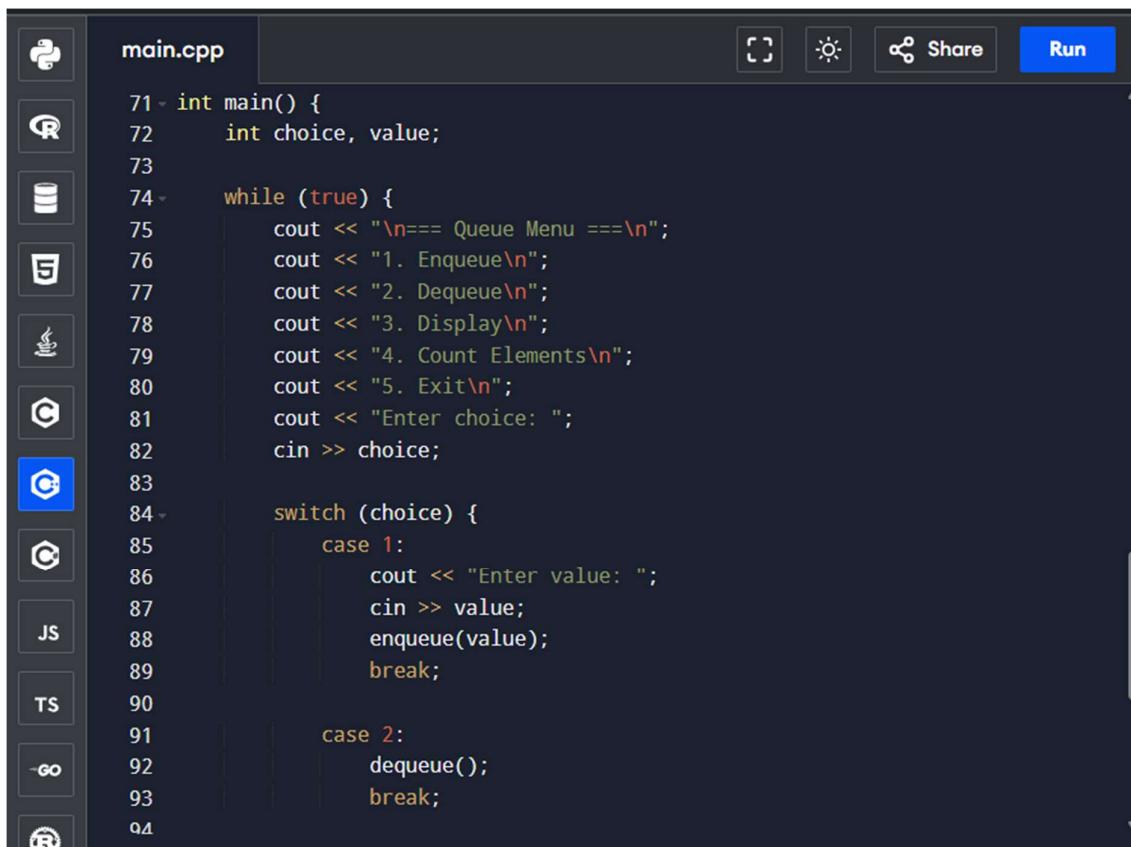
main.cpp

```
24
25 // Enqueue
26 void enqueue(int value) {
27     if (isFull()) {
28         cout << "Queue Overflow!\n";
29         return;
30     }
31
32     if (isEmpty()) {
33         front = rear = 0;
34     } else {
35         rear++;
36     }
37
38     queueArr[rear] = value;
39     cout << value << " inserted.\n";
40 }
41
42 // Dequeue
43 void dequeue() {
44     if (isEmpty()) {
45         cout << "Queue Underflow!\n";
46         return;
47     }
48 }
```



```
main.cpp

48     cout << queueArr[front] << " removed.\n";
49
50
51     if (front == rear) {
52         front = rear = -1;
53     } else {
54         front++;
55     }
56 }
57
58 // Display queue
59 void display() {
60     if (isEmpty()) {
61         cout << "Queue is empty.\n";
62         return;
63     }
64
65     cout << "Queue: ";
66     for (int i = front; i <= rear; i++) {
67         cout << queueArr[i] << " ";
68     }
69     cout << endl;
70 }
71 int main() {
```



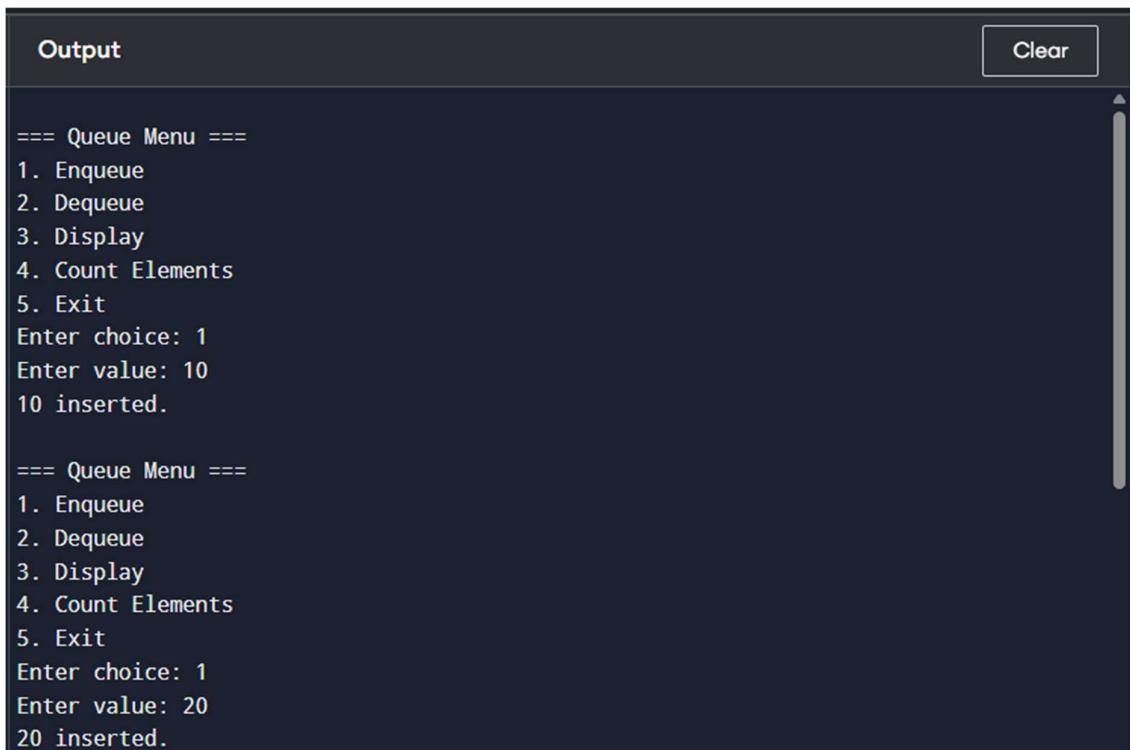
```
main.cpp

71 int main() {
72     int choice, value;
73
74     while (true) {
75         cout << "\n==== Queue Menu ====\n";
76         cout << "1. Enqueue\n";
77         cout << "2. Dequeue\n";
78         cout << "3. Display\n";
79         cout << "4. Count Elements\n";
80         cout << "5. Exit\n";
81         cout << "Enter choice: ";
82         cin >> choice;
83
84         switch (choice) {
85             case 1:
86                 cout << "Enter value: ";
87                 cin >> value;
88                 enqueue(value);
89                 break;
90
91             case 2:
92                 dequeue();
93                 break;
94         }
95     }
96 }
```



```
95     case 3:
96         display();
97         break;
98
99     case 4:
100        cout << "Total elements: " << countElements() << endl;
101        break;
102
103    case 5:
104        return 0;
105
106    default:
107        cout << "Invalid choice!\n";
108    }
109 }
110 }
```

Code output:



Output	Clear
==== 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.	

```
Output Clear  
==== 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

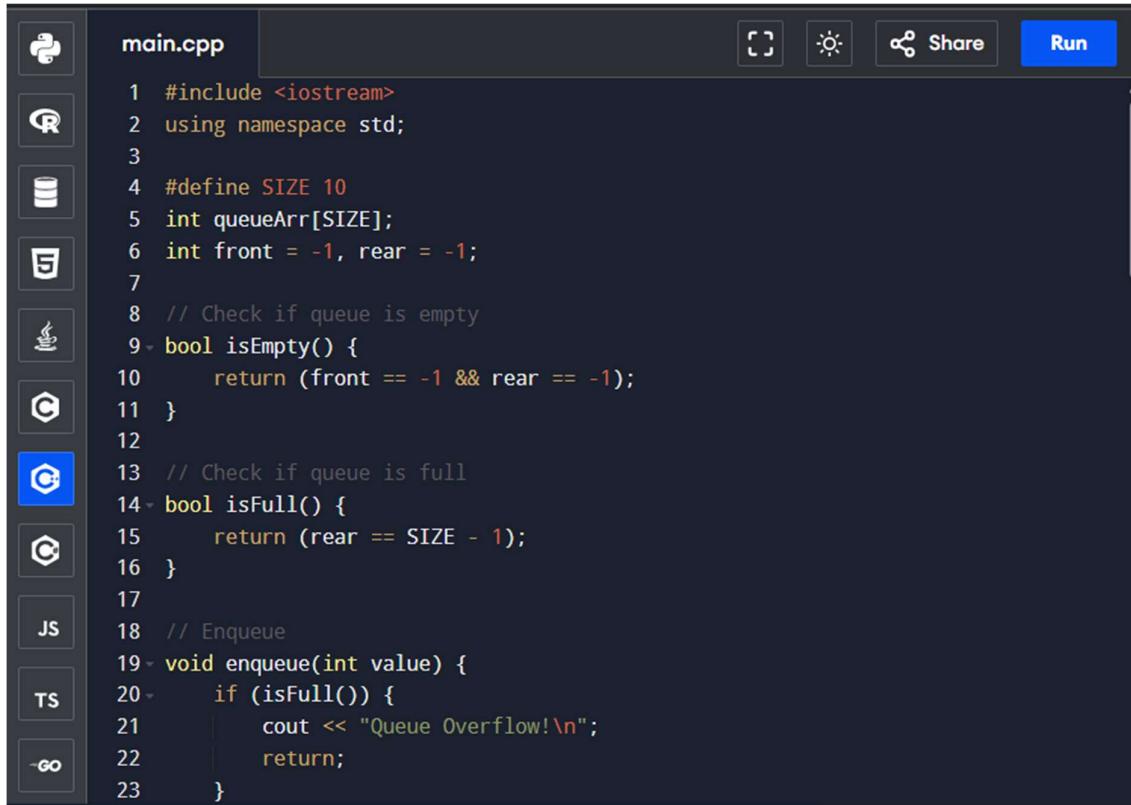
 temp[index] ← queueArr[i]

```
index ← index + 1
```

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

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

7. Print "Queue reversed successfully"



```
main.cpp
1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 10
5 int queueArr[SIZE];
6 int front = -1, rear = -1;
7
8 // Check if queue is empty
9 bool isEmpty() {
10     return (front == -1 && rear == -1);
11 }
12
13 // Check if queue is full
14 bool isFull() {
15     return (rear == SIZE - 1);
16 }
17
18 // Enqueue
19 void enqueue(int value) {
20     if (isFull()) {
21         cout << "Queue Overflow!\n";
22         return;
23     }
```

The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar containing icons for various programming languages: Python, R, SQL, HTML, CSS, Java, C, C++, JavaScript, TypeScript, Go, and Docker. The main area displays a C++ file named `main.cpp`. The code implements a queue reversal algorithm using an array. It includes functions for enqueueing values, reversing the queue, and displaying its contents.

```
main.cpp

23     }
24     if (isEmpty()) {
25         front = rear = 0;
26     } else {
27         rear++;
28     }
29     queueArr[rear] = value;
30 }
31
32 // Reverse queue
33 void reverseQueue() {
34     if (isEmpty()) {
35         cout << "Queue is empty!\n";
36         return;
37     }
38
39     int temp[SIZE];
40     int index = 0;
41
42     // Copy in reverse order from original queue
43     for (int i = rear; i >= front; i--) {
44         temp[index++] = queueArr[i];
45     }
46 }
```

This screenshot shows the continuation of the C++ code from the previous editor. It completes the reversal process by copying the reversed elements back into the queue array, resetting the front and rear pointers, and displaying the successfully reversed queue.

```
main.cpp

46
47     // Copy back from temp to queueArr
48     for (int i = 0; i < index; i++) {
49         queueArr[i] = temp[i];
50     }
51
52     // Reset front and rear
53     front = 0;
54     rear = index - 1;
55
56     cout << "Queue reversed successfully!\n";
57 }
58
59 // Display queue
60 void display() {
61     if (isEmpty()) {
62         cout << "Queue is empty!\n";
63         return;
64     }
65
66     cout << "Queue: ";
67     for (int i = front; i <= rear; i++)
68         cout << queueArr[i] << " ";
```

```
68         cout << queue[i] << " ";
69     cout << endl;
70 }
71
72 int main() {
73     int n, value;
74
75     cout << "How many values to insert? ";
76     cin >> n;
77
78     cout << "Enter values:\n";
79     for (int i = 0; i < n; i++) {
80         cin >> value;
81         enqueue(value);
82     }
83
84     cout << "\nOriginal Queue:\n";
85     display();
86
87     reverseQueue();
88
89     cout << "\nReversed Queue:\n";
90     display();
91 }
```

```
91
92     return 0;
93 }
94
```

Code output:

Output	Clear
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:

The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar containing icons for various programming languages: Python, R, SQL, Markdown, LaTeX, C, C++, Go, and Rust. The main area displays the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 20
5 int queueArr[SIZE];
6 int front = -1, rear = -1;
7
8 // Check empty
9 bool isEmpty() {
10     return (front == -1 && rear == -1);
11 }
12
13 // Check full
14 bool isFull() {
15     return (rear == SIZE - 1);
16 }
17
18 // Enqueue
19 void enqueue(int value) {
20     if (isFull()) {
21         cout << "Queue Overflow!\n";
22         return;
23     }
24     if (isEmpty()) {
```

The screenshot shows the continuation of the C++ code from the previous editor:

```
25         front = rear = 0;
26     } else {
27         rear++;
28     }
29     queueArr[rear] = value;
30 }
31
32 // Sort the queue
33 void sortQueue() {
34     if (isEmpty()) {
35         cout << "Queue is empty!\n";
36         return;
37     }
38
39     for (int i = front; i <= rear; i++) {
40         for (int j = i + 1; j <= rear; j++) {
41             if (queueArr[i] > queueArr[j]) {
42                 int temp = queueArr[i];
43                 queueArr[i] = queueArr[j];
44                 queueArr[j] = temp;
45             }
46         }
47     }
48 }
```

The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar containing icons for various programming languages: Python, R, Shell, Go, C, C++, JavaScript, TypeScript, and Go. The main window displays a C++ file named `main.cpp`. The code implements a queue sorting algorithm using insertion sort. It includes functions for enqueueing values, displaying the queue, and sorting it. The code is annotated with line numbers from 48 to 64.

```
main.cpp
...
48
49     cout << "Queue sorted successfully!\n";
50 }
51
52 // Display
53 void display() {
54     if (isEmpty()) {
55         cout << "Queue is empty!\n";
56         return;
57     }
58
59     cout << "Queue: ";
60     for (int i = front; i <= rear; i++) {
61         cout << queueArr[i] << " ";
62     cout << endl;
63 }
64
65 int main() {
66     int n, value;
67
68     cout << "How many values to insert? ";
69     cin >> n;
70
71     cout << "Enter values:\n";
72     for (int i = 0; i < n; i++) {
73         cin >> value;
74         enqueue(value);
75     }
76
77     cout << "\nOriginal Queue:\n";
78     display();
79
80     sortQueue();
81
82     cout << "\nSorted Queue:\n";
83     display();
84
85     return 0;
86 }
87
```

This screenshot shows the continuation of the C++ code from the previous window. The code completes the `main()` function by returning 0. The code is annotated with line numbers from 71 to 87.

```
71     cout << "Enter values:\n";
72     for (int i = 0; i < n; i++) {
73         cin >> value;
74         enqueue(value);
75     }
76
77     cout << "\nOriginal Queue:\n";
78     display();
79
80     sortQueue();
81
82     cout << "\nSorted Queue:\n";
83     display();
84
85     return 0;
86 }
87
```

```
Output Clear  
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:

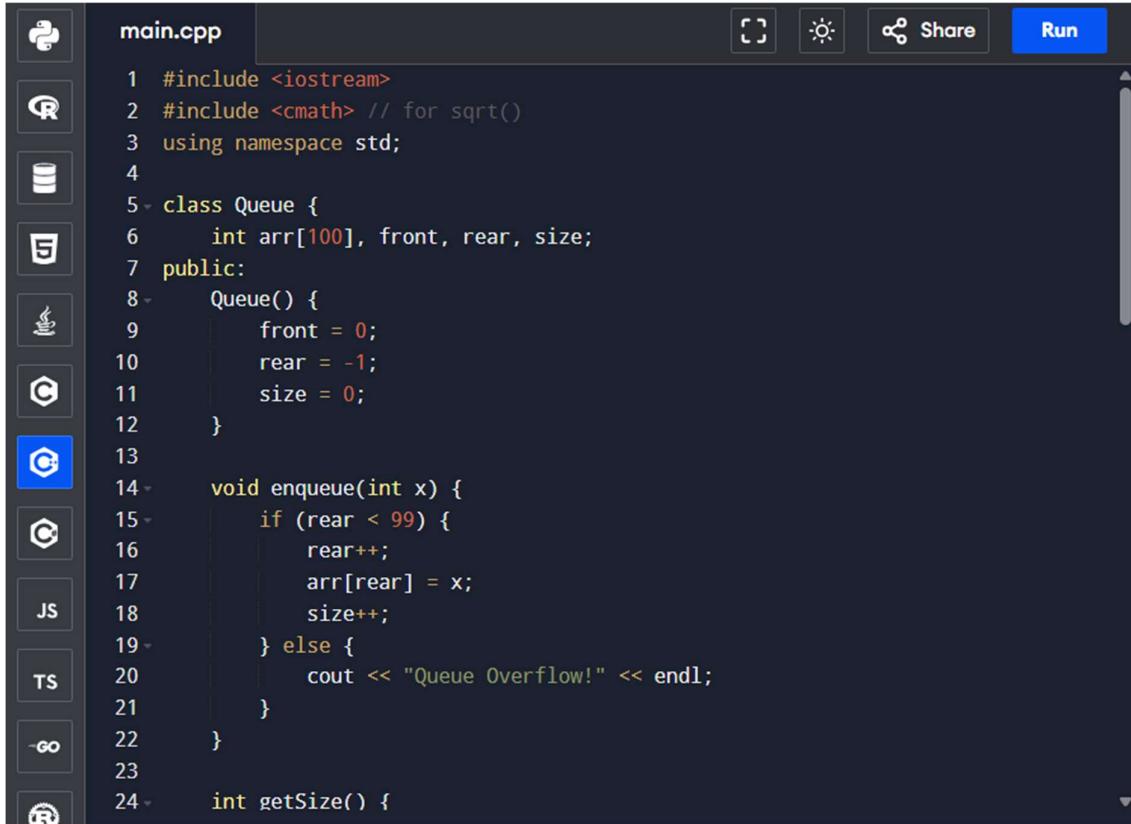
5. Write a program to find the mean, variance and standard deviation of all elements of a Queue

Algorithm:

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

6. Display Results.

code:



The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar containing icons for various programming languages: Python, R, SQL, Markdown, C, C++, Go, and Rust. The main window title is "main.cpp". The code itself is a C++ program defining a Queue class with enqueue and getSize methods. The code is as follows:

```
1 #include <iostream>
2 #include <cmath> // for sqrt()
3 using namespace std;
4
5 class Queue {
6     int arr[100], front, rear, size;
7 public:
8     Queue() {
9         front = 0;
10    rear = -1;
11    size = 0;
12 }
13
14 void enqueue(int x) {
15     if (rear < 99) {
16         rear++;
17         arr[rear] = x;
18         size++;
19     } else {
20         cout << "Queue Overflow!" << endl;
21     }
22 }
23
24 int getSize() {
```

The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar containing icons for various programming languages: Python, R, SQL, HTML, CSS, Java, C, C++, JavaScript, TypeScript, Go, and Rust. The main window title is "main.cpp". The code is as follows:

```
25     return size;
26 }
27
28 void calculateStatistics() {
29     if (size == 0) {
30         cout << "Queue is empty!" << endl;
31         return;
32     }
33
34     double sum = 0.0;
35     for (int i = front; i <= rear; i++)
36         sum += arr[i];
37
38     double mean = sum / size;
39
40     double varianceSum = 0.0;
41     for (int i = front; i <= rear; i++)
42         varianceSum += (arr[i] - mean) * (arr[i] - mean);
43
44     double variance = varianceSum / size;
45     double stdDev = sqrt(variance);
46
47     cout << "Mean = " << mean << endl;
48     cout << "Variance = " << variance << endl;
```

The screenshot shows the same code editor interface with the same toolbar. The main window title is "main.cpp". The code is now complete with the addition of the main function and the inclusion of the queue class definition from the previous screenshot.

```
47     cout << "Mean = " << mean << endl;
48     cout << "Variance = " << variance << endl;
49     cout << "Standard Deviation = " << stdDev << endl;
50 }
51 };
52
53 int main() {
54     Queue q;
55     int n, element;
56
57     cout << "Enter number of elements in queue: ";
58     cin >> n;
59
60     cout << "Enter elements: ";
61     for (int i = 0; i < n; i++) {
62         cin >> element;
63         q.enqueue(element);
64     }
65
66     q.calculateStatistics();
67
68     return 0;
69 }
```

Code output:

Output **Clear**

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

== Code Execution Successful ==
```