Marwadi University	**Marwadi University** **Faculty of Technology** **Department of Information and Communication Technology**
Subject: DSC (01CT0308)	**Aim:** Implementations of circular queue menu-driven program.
Experiment No: 4	**Date:** **20- 10 - 2023** | **Enrolment No:-** 92200133030

Experiment – 4

Objective: Implementations of circular queue menu-driven program.

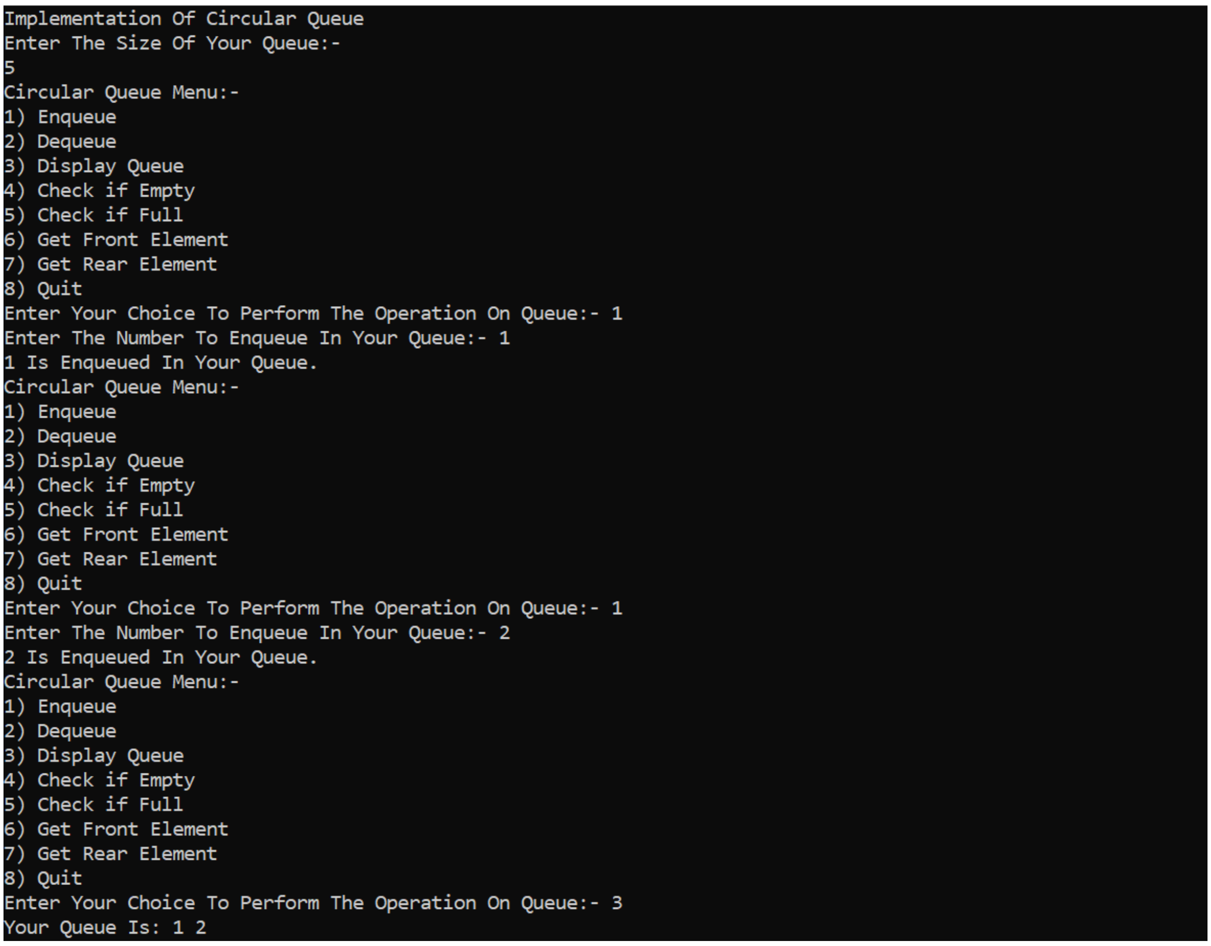
Code :-

1) Using Array

```
#include <iostream>
using namespace std;
class CircularQueue {
private:
   int size;
   int front;
   int rear;
   int* queue;
public:
   CircularQueue(int n) {
      size = n;
      front = -1;
      rear = -1;
      queue = new int[size];
   }
   bool isEmpty() {
      return front == -1;
   }
   bool isFull() {
      return (rear + 1) % size == front;
   }
   void enqueue(int num) {
      if (isFull()) {
         cout << "Queue Is Full." << endl;
         return;
      } else {
         if (isEmpty()) {
            front = rear = 0;
        } else {
            rear = (rear + 1) % size;
        }
        queue[rear] = num;
        cout << num << " Is Enqueued In Your Queue." << endl;
    }
}
void dequeue() {
    if (isEmpty()) {
        cout << "Queue Is Empty." << endl;
        return;
    } else {
        int dequeued = queue[front];
        if (front == rear) {
            front = rear = -1;
        } else {
            front = (front + 1) % size;
        }
        cout << dequeued << " Is Dequeued From Your Queue." << endl;
    }
}
void display() {
    if (isEmpty()) {
        cout << "Queue is Empty." << endl;
    } else {
        cout << "Your Queue Is: ";
        if (front <= rear) {
            for (int i = front; i <= rear; i++) {
                cout << queue[i] << " ";
            }
        } else {
            for (int i = front; i < size; i++) {
                cout << queue[i] << " ";
            }
            for (int i = 0; i <= rear; i++) {
                cout << queue[i] << " ";
            }
        }
        cout << endl;
    }
}
int getFront() {
    return queue[front];
}
int getRear() {
    return queue[rear];
    }
};
int main() {
    int size;
    int num;
    cout << "Implementation Of Circular Queue " << endl;
    cout << "Enter The Size Of Your Queue:-" << endl;
    cin >> size;
    CircularQueue circularqueue(size);
    int choice = -1;
    do {
        cout << "Circular Queue Menu:- " << endl;
        cout << "1) Enqueue" << endl;
        cout << "2) Dequeue" << endl;
        cout << "3) Display Queue" << endl;
        cout << "4) Check if Empty" << endl;
        cout << "5) Check if Full" << endl;
        cout << "6) Get Front Element" << endl;
        cout << "7) Get Rear Element" << endl;
        cout << "8) Quit" << endl;
        cout << "Enter Your Choice To Perform The Operation On Queue:- ";
        cin >> choice;
        switch (choice) {
            case 1: {
                cout << "Enter The Number To Enqueue In Your Queue:- ";
                cin >> num;
                circularqueue.enqueue(num);
                break;
            }
            case 2: {
                circularqueue.dequeue();
                break;
            }
            case 3: {
                circularqueue.display();
                break;
            }
            case 4: {
                if (circularqueue.isEmpty()) {
                    cout << "Queue is empty." << endl;
                } else {
                    cout << "Queue is not empty." << endl;
                }
                break;
            }
            case 5: {
                if (circularqueue.isFull()) {
```

```
                cout << "Queue is full." << endl;
            } else {
                cout << "Queue is not full." << endl;
            }
            break;
        }
        case 6: {
            if (!circularqueue.isEmpty()) {
                cout << "Front element: " << circularqueue.getFront() << endl;
            } else {
                cout << "Queue is empty." << endl;
            }
            break;
        }
        case 7: {
            if (!circularqueue.isEmpty()) {
                cout << "Rear element: " << circularqueue.getRear() << endl;
            } else {
                cout << "Queue is empty." << endl;
            }
            break;
        }
        case 8: {
            cout << "Program Ends!" << endl;
            return 0;
        }
        default: {
            cout << "Enter a valid choice." << endl;
            break;
        }
      }
    } while (choice != 8);
    return 0;
}
```

Output:

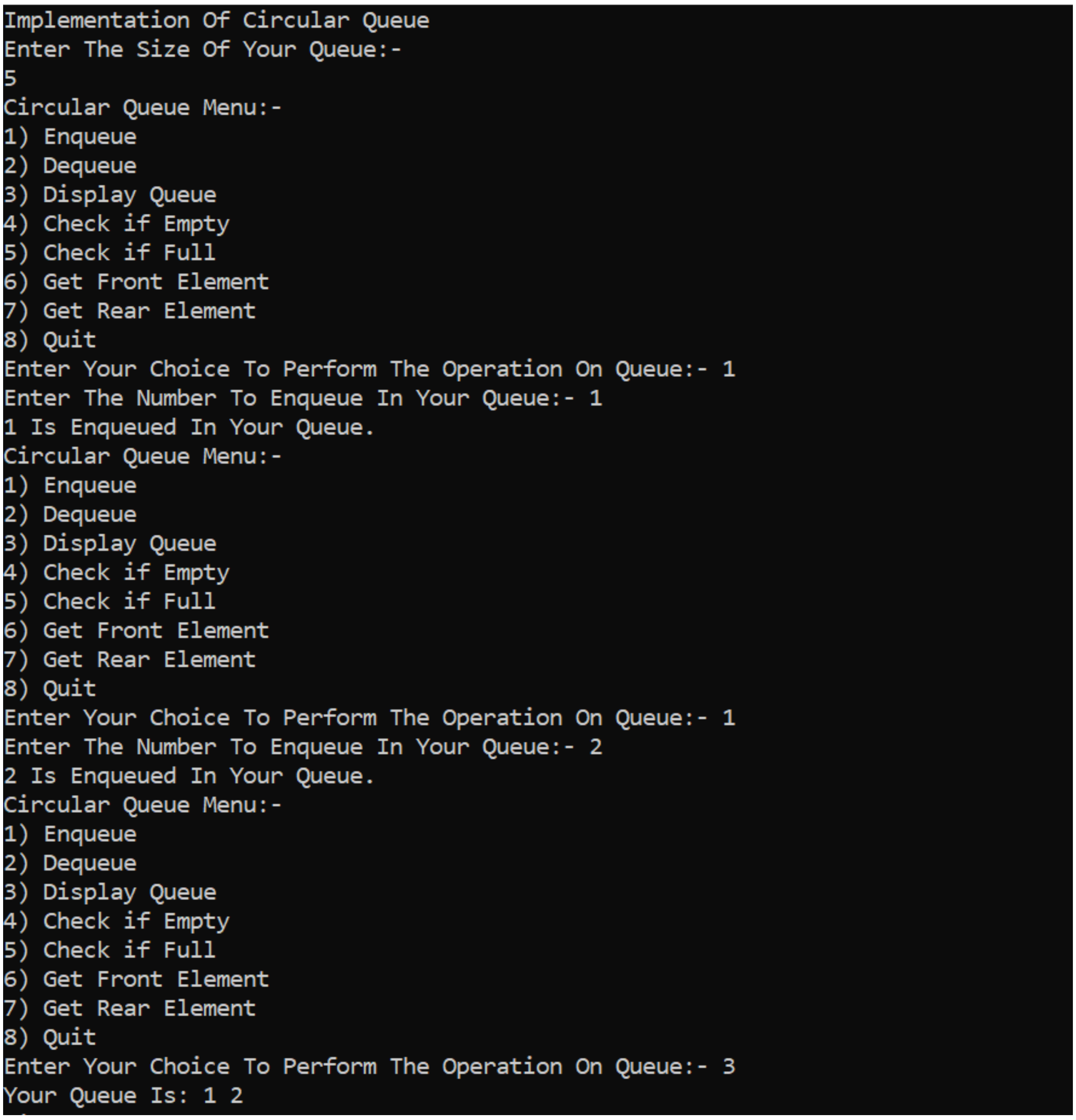


```
Implementation Of Circular Queue
Enter The Size Of Your Queue:-
5
Circular Queue Menu:-
1) Enqueue
2) Dequeue
3) Display Queue
4) Check if Empty
5) Check if Full
6) Get Front Element
7) Get Rear Element
8) Quit
Enter Your Choice To Perform The Operation On Queue:- 1
Enter The Number To Enqueue In Your Queue:- 1
1 Is Enqueued In Your Queue.
Circular Queue Menu:-
1) Enqueue
2) Dequeue
3) Display Queue
4) Check if Empty
5) Check if Full
6) Get Front Element
7) Get Rear Element
8) Quit
Enter Your Choice To Perform The Operation On Queue:- 1
Enter The Number To Enqueue In Your Queue:- 2
2 Is Enqueued In Your Queue.
Circular Queue Menu:-
1) Enqueue
2) Dequeue
3) Display Queue
4) Check if Empty
5) Check if Full
6) Get Front Element
7) Get Rear Element
8) Quit
Enter Your Choice To Perform The Operation On Queue:- 3
Your Queue Is: 1 2
```

2) Using LinkedList

```
#include <iostream>
using namespace std;
class Node {
public:
    int data;
    Node* next;
    Node(int val) {
        data = val;
        next = NULL;
    }
};
class CircularQueue {
private:
    Node* front;
    Node* rear;
    int size;
    int no_of_elements = 0;
public:
    CircularQueue(int num) { // Added public access specifier
        front = NULL;
        rear = NULL;
        size = num;
    }
    bool isEmpty() {
        return front == NULL;
    }
    bool isFull() {
        return no_of_elements == size;
    }
    void enqueue(int num) {
        if (isFull()) {
            cout << "Queue Is Full." << endl;
            return;
        }
        Node* n = new Node(num);
        if (isEmpty()) {
            front = n;
        } else {
            rear->next = n;
        }
        rear = n;
        rear->next = front;
        no_of_elements++; // Increment the number of elements
        cout << num << " Is Enqueued In Your Queue." << endl;
    }
    void dequeue() {
        if (isEmpty()) {
            cout << "Queue Is Empty." << endl;
        } else {
            int remove = front->data;
            if (front == rear) {
                delete front;
                front = rear = NULL;
            } else {
                Node* todelete = front;
                front = front->next;
                rear->next = front;
                delete todelete;
            }
            no_of_elements--; // Decrement the number of elements
            cout << remove << " Is Dequeued From Your Queue." << endl;
        }
    }
    void display() {
        if (isEmpty()) {
            cout << "Queue Is Empty." << endl;
        } else {
            Node* temp = front;
            cout << "Your Queue Is: ";
            do {
                cout << temp->data << " ";
                temp = temp->next;
            } while (temp != front);
            cout << endl;
        }
    }
    int getFront() {
        return front->data;
    }
    int getRear() {
        return rear->data;
    }
};
int main() {
    int size;
    int num;
    cout << "Implementation Of Circular Queue " << endl;
    cout << "Enter The Size Of Your Queue:-" << endl;
    cin >> size;
    CircularQueue circularqueue(size);
    int choice = -1;
    do {
        cout << "Circular Queue Menu:- " << endl;
        cout << "1) Enqueue" << endl;
        cout << "2) Dequeue" << endl;
        cout << "3) Display Queue" << endl;
        cout << "4) Check if Empty" << endl;
        cout << "5) Check if Full" << endl;
        cout << "6) Get Front Element" << endl;
        cout << "7) Get Rear Element" << endl;
        cout << "8) Quit" << endl;
        cout << "Enter Your Choice To Perform The Operation On Queue:- ";
        cin >> choice;
        switch (choice) {
            case 1: {
                cout << "Enter The Number To Enqueue In Your Queue:- ";
                cin >> num;
                circularqueue.enqueue(num);
                break;
            }
            case 2: {
                circularqueue.dequeue();
                break;
            }
            case 3: {
                circularqueue.display();
                break;
            }
            case 4: {
                if (circularqueue.isEmpty()) {
                    cout << "Queue is Empty." << endl;
                } else {
                    cout << "Queue is not Empty." << endl;
                }
                break;
            }
            case 5: {
                if (circularqueue.isFull()) {
                    cout << "Queue is Full." << endl;
                } else {
                    cout << "Queue is not Full." << endl;
                }
                break;
            }
            case 6: {
                if (!circularqueue.isEmpty()) {
                    cout << "Front Element: " << circularqueue.getFront() << endl;
                } else {
                    cout << "Queue is Empty." << endl;
                }
                break;
            }
            case 7: {
                if (!circularqueue.isEmpty()) {
                    cout << "Rear Element: " << circularqueue.getRear() << endl;
                } else {
                    cout << "Queue is Empty." << endl;
                }
                break;
            }
            case 8: {
                cout << "Program Ends!" << endl;
                return 0;
            }
            default: {
                cout << "Enter a valid choice." << endl;
                break;
            }
        }
    } while (choice != 8);
    return 0;
}
```

Output:



```
Implementation Of Circular Queue
Enter The Size Of Your Queue:-
5
Circular Queue Menu:-
1) Enqueue
2) Dequeue
3) Display Queue
4) Check if Empty
5) Check if Full
6) Get Front Element
7) Get Rear Element
8) Quit
Enter Your Choice To Perform The Operation On Queue:- 1
Enter The Number To Enqueue In Your Queue:- 1
1 Is Enqueued In Your Queue.
Circular Queue Menu:-
1) Enqueue
2) Dequeue
3) Display Queue
4) Check if Empty
5) Check if Full
6) Get Front Element
7) Get Rear Element
8) Quit
Enter Your Choice To Perform The Operation On Queue:- 1
Enter The Number To Enqueue In Your Queue:- 2
2 Is Enqueued In Your Queue.
Circular Queue Menu:-
1) Enqueue
2) Dequeue
3) Display Queue
4) Check if Empty
5) Check if Full
6) Get Front Element
7) Get Rear Element
8) Quit
Enter Your Choice To Perform The Operation On Queue:- 3
Your Queue Is: 1 2
```