PRN: 2020BTEIT00041

Circular Queue:

```cpp
/*
    PRN: 2020BTEIT00041
    Name: Om Vivek Gharge
*/

/*
    Circular Queue  -  Implementation of a circular queue using an linked list
*/

#include <bits/stdc++.h>
using namespace std;

// Node class
class Node{
public:
    int data;
    Node *next;

    Node(){
        this->data = 0;
        this->next = NULL;
    }

    Node(int data){
        this->data = data;
        this->next = NULL;
    }
};

// Queue class
class Queue{
public:
    Node* front;
    Node* rear;

    Queue(){
        this->front = NULL;
        this->rear = NULL;
    }

    void enqueue(int data);
    int dequeue();
    void Display();
};

// enqueue function
void Queue::enqueue(int data){
    // create a new node
```

```
    Node* new_node = new Node(data);

    // if queue is empty
    if(this->front == NULL){
        // make hte new node as the front and rear
        this->front = new_node;
        this->rear = new_node;
    }
    // if queue is not empty
    else{
        // make the new node as the rear
        this->rear->next = new_node;
        this->rear = new_node;
    }
}

// dequeue function
int Queue::dequeue(){
    // if queue is empty
    if(this->front == NULL){
        cout << "Queue is empty" << endl;
        return -1;
    }
    // if queue is not empty
    else{
        // store the front node in a temp variable
        Node* temp = this->front;

        // store the data of the front node
        int data = temp->data;

        // make the front as the next of the front
        this->front = this->front->next;

        // delete the temp node
        delete temp;

        // if the front is NULL
        if(this->front == NULL){
            // make the rear as NULL
            this->rear = NULL;
        }

        return data;
    }
}

void Queue::Display(){
```

```
    // if queue is empty
    if(this->front == NULL){
        cout << "Queue is empty" << endl;
        return;
    }
    // if queue is not empty
    else{
        // create a temp node
        Node* temp = this->front;

        // traverse the queue
        while(temp != NULL){
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
}

int main(){
    Queue q;

    // Menu driven program to implement a circular queue
    int choice;

    do{
        cout<<"---------------------Menu---------------------\n";
        cout<<"1. Enqueue"<<endl;
        cout<<"2. Dequeue"<<endl;
        cout<<"3. Display"<<endl;
        cout<<"4. Exit"<<endl;


        cout<<"Enter your choice: ";
        cin>>choice;

        switch (choice){
        case 1:
                int data;
                cout<<"Enter the data to be enqueued: ";
                cin>>data;
                q.enqueue(data);
                break;
            case 2:
                cout<<"Dequeued element: "<<q.dequeue()<<endl;
                break;
            case 3:
                cout<<"Display"<<endl;
                q.Display();
                break;
            case 4:
                cout<<"Exiting..."<<endl;
                break;
            default:
                cout<<"Invalid choice"<<endl;
        }
    }while(choice != 4);

    return 0;
}
```

OUTPUT:

```
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the data to be enqueued: 1
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the data to be enqueued: 2
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the data to be enqueued: 3
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Display
1 2 3
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued element: 1
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Display
2 3
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
Exiting...
```