Queue operation using Linked List

Enqueue & Dequeue(Insertion & Deletion)-

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

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

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

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

    Queue() {
        front = nullptr;
        rear = nullptr;
    }

    bool isEmpty() {
        return front == nullptr;
    }
```

```
void Enqueue(int data) {
    Node* newNode = new Node(data);
    if (rear == nullptr) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    cout << "Inserted Element: " << data << endl;
}

void Dequeue() {
    if (isEmpty()) {
        cout << "Queue is empty, cannot dequeue." << endl;
        return;
    }
    Node* temp = front;
    front = front->next;
    if (front == nullptr) {
        rear = nullptr;
    }
    cout << "Deleted Element: " << temp->data << endl;
    delete temp;
}

void Display() {
    if (isEmpty()) {
        cout << "Queue is empty" << endl;
        return;
    }
    Node* temp = front;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
```

```
    }
};

int main() {
    Queue q;
    q.Enqueue(10);
    q.Enqueue(20);
    q.Enqueue(30);
    q.Display();

    q.Dequeue();
    q.Display();

    q.Dequeue();
    q.Display();

    return 0;
}
```

Output-

```
Inserted Element: 10
Inserted Element: 20
Inserted Element: 30
10 20 30
Deleted Element: 10
20 30
Deleted Element: 20
30
```