Lab Task 7:

Name:Saif Majid Khan.

SAP-ID: 57114

DSA

Github:

https://github.com/saif01234567/LabTasks-DS.git

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

struct Node
{
    int data;
    Node *next;
};

class Queue
{
    Node *front, *rear;
    int count; // Variable to count the number of elements
    const int maxSize = 5; // Define a maximum size for the queue

public:
```

```
Queue()
 {
    front = rear = NULL;
    count = 0; // Initialize count
 }

 void Enqueue(int data) // Inserting from rear
 {
    if (count >= maxSize) // Check for overflow
    {
       cout << "Queue Overflow! Cannot add more elements." << endl;
       return;
    }

    Node *newnode = new Node;
    newnode->data = data;
    newnode->next = NULL;
```

```
if (front == NULL)
      {
         front = rear = newnode;
      }
      else
      {
         rear->next = newnode;
         rear = newnode;
      }
      count++; // Increment count
   }
void Dequeue() // Optional: Dequeue method for completeness
   {
      if (front == NULL) // Check for underflow
      {
         cout << "Queue Underflow! Cannot remove elements." << endl;
         return;
      }
```

```
Node *temp = front;
        front = front->next;
        delete temp;

        if (front == NULL) // If the queue is empty, reset rear
            rear = NULL;

        count--; // Decrement count
    }

    int Count() // Method to count elements in the queue
    {
        return count;
    }
```

```
void Clear() // Method to clear the entire queue
  {
     while (front != NULL)
     {
        Dequeue(); // Reuse the Dequeue method to clear the queue
     }
     cout << "Queue cleared." << endl;
  }

  void Display() // Method to display the queue contents
  {
     Node *temp = front;
     if (temp == NULL)
     {
        cout << "Queue is empty." << endl;
        return;
     }
```

```
        cout << "Queue contents: ";
        while (temp != NULL)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};
int main() // Entry point of the program
{
    Queue q; // Create a Queue object
    q.Enqueue(10);
    q.Enqueue(20);
    q.Enqueue(30);
    q.Enqueue(40);
    q.Enqueue(50);
    q.Enqueue(60); // This should trigger overflow
```

```
    cout << "Current queue size: " << q.Count() << endl; // Display count

    q.Display(); // Display the contents of the queue

    q.Clear(); // Clear the queue
    cout << "Current queue size after clearing: " << q.Count() << endl; //
Check count after clearing

    return 0; // Indicate that the program ended successfully
}
```

Output

Clear

```
/tmp/nKYqFtjvgj.o
Queue Overflow! Cannot add more elements.
Current queue size: 5
Queue contents: 10 20 30 40 50
Queue cleared.
Current queue size after clearing: 0


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