Stack using Linked List

```
class linkedStack
{
public:
    linkedStack() : m_top(nullptr) {}
    void push(float data)
    {
        Node* new_node = new Node(data);
        new_node->m_next = m_top;
        m_top = new_node;
    }
    void pop()
    {
        if (m_top)
        {
            Node* temp = m_top;
            m_top = m_top->m_next;
            delete temp;
        }
        else std::cerr << "Stack is empty. Cannot pop.\n";
    }
    void display()
    {
        Node* current = m_top;
        if (!current)
        {
            std::cout << "Stack is empty.\n";
            return;
        }
        std::cout << "The data in the stack is: \n";
        while (current)
        {
            std::cout << current->m_data << " ";
            current = current->m_next;
        }
        std::cout << "\n";
    }
    ~linkedStack()
    {
        Node* current = m_top;
        while (current)
        {
            Node* next = current->m_next;
            delete current;
            current = next;
        }
    }
private:
    Node* m_top;
};
```

Queue using Linked List

```
class linkedQueue
{
public:
    linkedQueue() : m_front(nullptr), m_rear(nullptr) {}

    void enqueue(float data)
    {
        Node* new_node = new Node(data);
        if (!m_front)
        {
            m_front = new_node;
            m_rear = new_node;
        }
        else
        {
            m_rear->m_next = new_node;
            m_rear = new_node;
        }
    }

    void dequeue()
    {
        if (m_front)
        {
            Node* temp = m_front;
            m_front = m_front->m_next;
            delete temp;
            // If the queue becomes empty after dequeue
            if (!m_front) m_rear = nullptr;
            return;
        }
        std::cerr << "linkedQueue is empty. Cannot dequeue.\n";
    }

    void display()
    {
        Node* current = m_front;
        if (!current)
        {
            std::cout << "linkedQueue is empty.\n";
            return;
        }
        std::cout << "The data in the queue is: \n";
        while (current)
        {
            std::cout << current->m_data << " ";
            current = current->m_next;
        }
        std::cout << "\n";
    }
```

```
    ~linkedQueue()
    {
        Node* current = m_front;
        while (current)
        {
            Node* next = current->m_next;
            delete current;
            current = next;
        }
    }

private:
    Node* m_front;
    Node* m_rear;
};
```

Output

Stack

```
Stack Menu
(0) to push data,
(1) to pop data,
(2) to display the stack,
(3) to exit the program
0
Enter the data to push.
30

Stack Menu
(0) to push data,
(1) to pop data,
(2) to display the stack,
(3) to exit the program
2
The data in the stack is:
30 20 10

Stack Menu
(0) to push data,
(1) to pop data,
(2) to display the stack,
(3) to exit the program
1

Stack Menu
(0) to push data,
(1) to pop data,
(2) to display the stack,
(3) to exit the program
2
The data in the stack is:
20 10

Stack Menu
(0) to push data,
(1) to pop data,
(2) to display the stack,
(3) to exit the program
3
```

Queue

```
Queue Menu
(0) to enqueue data,
(1) to dequeue data,
(2) to display the queue,
(3) to exit the program
0
Enter the data to enqueue.
10
Queue Menu
(0) to enqueue data,
(1) to dequeue data,
(2) to display the queue,
(3) to exit the program
0
Enter the data to enqueue.
20
Queue Menu
(0) to enqueue data,
(1) to dequeue data,
(2) to display the queue,
(3) to exit the program
0
Enter the data to enqueue.
30
Queue Menu
(0) to enqueue data,
(1) to dequeue data,
(2) to display the queue,
(3) to exit the program
2
The data in the queue is:
10 20 30
Queue Menu
(0) to enqueue data,
(1) to dequeue data,
(2) to display the queue,
(3) to exit the program
1
Queue Menu
(0) to enqueue data,
(1) to dequeue data,
(2) to display the queue,
(3) to exit the program
2
The data in the queue is:
20 30
```