Problem 1:

```
class LinkedList
{
public:
	LinkedList();
	~LinkedList();
	void addToList(int value); // add to the head of the linked list
	void reverse(); // Reverse the linked list
 private:
	struct Node
	{
		int num;
		Node *next;
	};
	Node *m_head;
};

LinkedList::LinkedList()
:m_head(nullptr)
{}

LinkedList::~LinkedList()
{
	Node *temp;
	while(m_head != nullptr) {
		temp = m_head;
		m_head = m_head->next;
		delete temp;
	}
}

void LinkedList::addToList(int value)
{
	Node *newNode = new Node;
	newNode->num = value;
	newNode->next = m_head;
	m_head = newNode;
}

//What about adding to the tail of the linked list? Try it out yourself

void LinkedList::reverse()
{
	Node *nextNode = nullptr, *prevNode = nullptr, *current = m_head;
	while(current) {
		nextNode = current->next;
		current->next = prevNode;
		prevNode = current;
		current = nextNode;
```

```
        }
        m_head = prevNode;
}

//What about removing a node from the linked list?
```

Problem 2 Doubly-linked list (Not circular, no dummy node)

```
class LinkedList
{
public:
        LinkedList();
        ~LinkedList();
        void addToList(int value); // add to the head of the linked list
        void remove(Node *node)
private:
        struct Node
        {
                int num;
                Node *next;
                Node *prev
        };
        Node *m_head;
        Node *m_tail;
};

void LinkedList::addToList(int value)
{
        Node *newNode = new Node;
        newNode->num = value;
        newNode->prev = nullptr;
        newNode->next = m_head;
        if(m_head == nullptr)
        {
                m_tail = newNode;
        }else{
                m_head->prev = newNode;
        }
        m_head = newNode;
}

void LinkedList::remove(Node *node)
{
        if(node == nullptr)
                return;
        if(node != m_head)
                node -> prev -> next = node -> next;
        else
                m_head = m_head -> next;
        if (node != m_tail)
```

```
                node -> next -> prev = node -> m_prev;
        else
                m_tail = m_tail ->m_prev;
        delete node;
}
```

Problem 3 circular doubly-linked list with a dummy node

```
class LinkedList
{
public:
        LinkedList();
        ~LinkedList();
        void addToList(int value); // add to the head of the linked list
        void remove(Node *node)
private:
        struct Node
        {
                int num;
                Node *next;
                Node *prev
        };
        Node *m_head;
};

LinkedList::LinkedList()
{
        Node *newNode = new Node;
        newNode->prev = newNode;
        newNode->next = newNode;
        m_head = newNode;
}

LinkedList::~LinkedList()
{
        Node *temp;
        Node *dummy = head;
        head = head->next;
        while(head != dummy) {
                temp = head;
                head = head->next;
                delete temp;
        }
        delete dummy;
}

void LinkedList::addToList(int value)
{
        Node *newNode = new Node;
```

```
        newNode->num = value;
        newNode->prev = m_head;
        newNode->next = m_head -> next;
        m_head->next->prev = newNode;
        m_head->next = newNode;
}

bool LinkedList::remove(Node *node)
{
        if(node==m_head)
                return false;
        node->prev->next = node->next;
        node->next->prev = node->prev;
        delete node;
}
```