template <typename T>
class LinkedList {
    T m_data;
    LinkedList* m_next;
    ... ...
};

LinkedList Bob;

T& at( int i )
{
    LinkedList* p = this;
    int k = 0;
    while( k<i && p->m_next != NULL ){
        p = p->m_next;
        k++;
    }

    if( p->m_next == NULL )
        error();

    return p->m_data;
}
1. Create new node
2. Copy data to new container
3. Insert new element
4. Set new node pointer
5. Set calling object pointer

**LinkedList::insert_front(T& x)**

```
LinkedList *p = new LinkedList;
p->m_data = m_data;
m_data = x;
p->m_next = m_next;
m_next = p;
```

**LinkedList::insert(LinkedList* p, T& x)**

```
LinkedList* q = new LinkedList;
q->m_data = p->m_data;
p->m_data = x;
q->m_next = p->m_next;
p->m_next = q;
```

---

1. Make a new node
2. Copy data
3. Insert new element
4. Set pointer of new container
5. Set pointer at position.

---

1. Copy next element into p's element
2. Make second pointer point to p's next
3. Change p's next to bypass node
3 change p's next to bypass node
4 delete q

```cpp
LinkedList::remove(LinkedList* p)
{
    p->m_data = p->m_next->m_data;
    LinkedList *q = p->m_next;
    p->m_next = q->m_next;
    delete q;
}
```