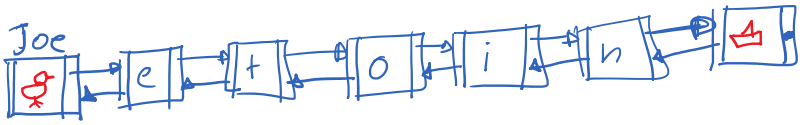
'LinkedList' Variants

Wednesday, September 25, 2019 4:57 PM

Joe <e, t, o, i, n>



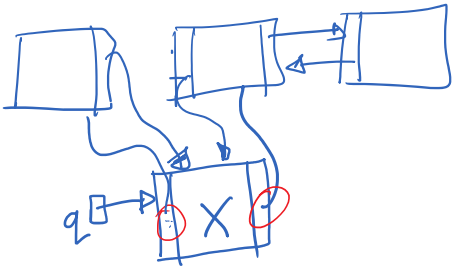
- The Doubly Linked List

```
template <typename T>
class DoublyLinkedList {
   T m_data;
   DoublyLinkedList* m_next;
   DoublyLinkedList* m_prev;
… …
};
```

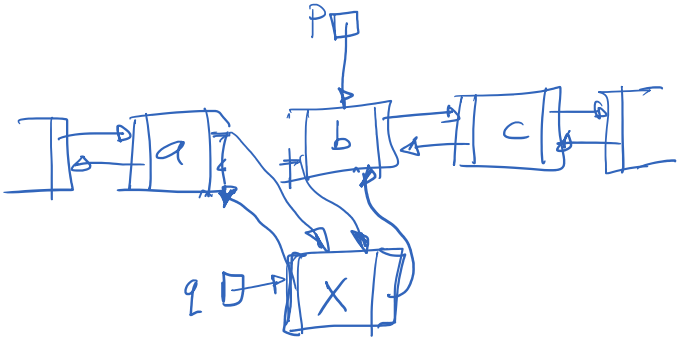
- Insert

Insert x at position p

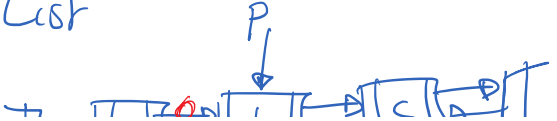
1 create new node.
2 insert element x
3 setup new container pointers
4 modify list's pointers



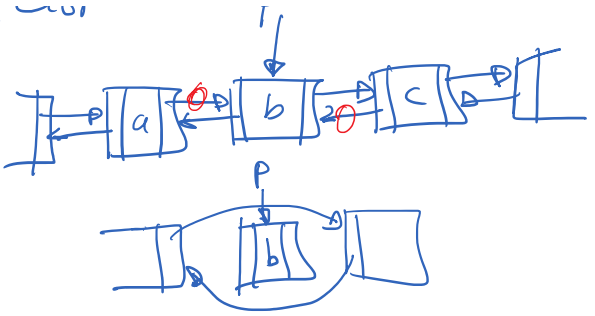

```
DoublyLinkedList::insert
   (DoublyLinkedList* p, T& x)
{
   DoublyLinkedList* q;
   q = new DoublyLinkedList;
   q->m_data = x;
   q->m_next = p;
   q->m_prev = p->m_prev;
   p->m_prev = q;
   q->m_prev->m_next = q;
}
```



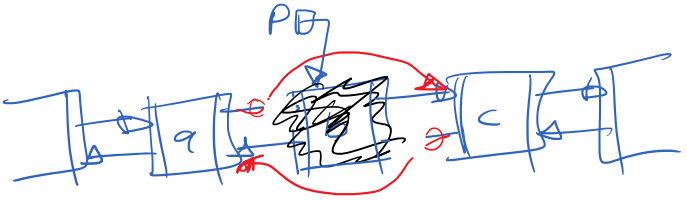
- Remove on Doubly Linked List



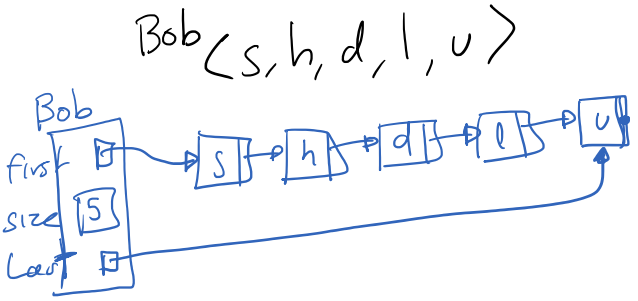
- Remove on Doubly Linked List




```
DoublyLinkedList::remove
   (DoublyLinkedList* p)
{
   p->m_prev->m_next = p->m_next;
   p->m_next->m_prev = p->m_prev;
   delete p;
}
```



- Encapsulated Linked List.

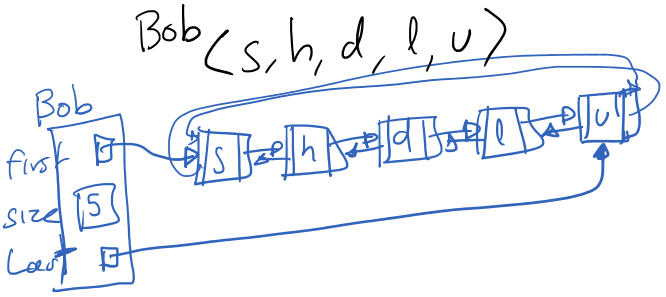



```
class LLNode
{
   T m_data;
   LLNode* m_next;
}

class LList
{
   int m_size;
   LLNode* m_first;
   LLNode* m_last;
};
```

Bob.insert_front(x);

Encapsulated Circular Doubly Linked List

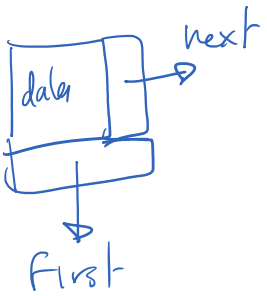



```
class LLNode
{
   T m_data;
   LLNode* m_next;
   LLNode* m_prev;
}

class LList
{
   int m_size;
   LLNode* m_first;
   LLNode* m_last;
```

Bob.insert_front(x);

```
};
```


next
data
first