Pointer Assignment

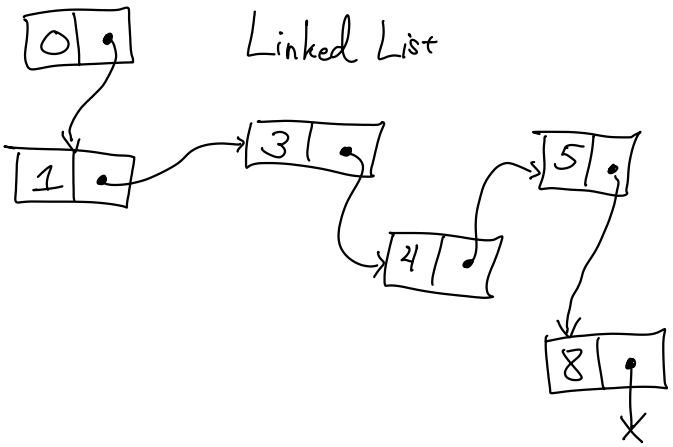
```
int x = 5;
int y = x;
x = 8;
```

```
Foo* x = new Foo();
Foo* y = x;
x = new Foo();
```

List ADT

abstract data type

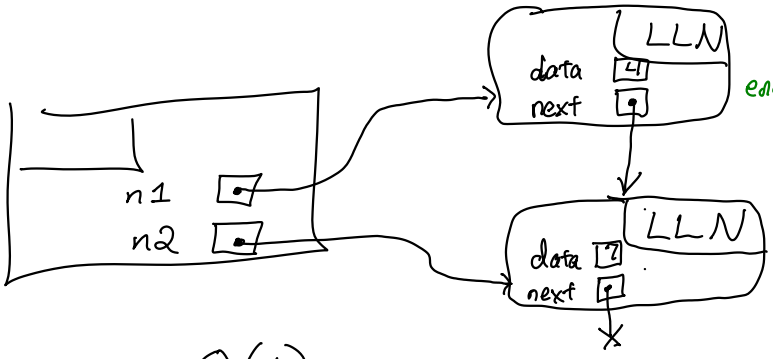
```
virtual int getSize() = 0;
virtual T getFirst() = 0;
virtual T removeFirst() = 0;
virtual void insertFirst(T element) = 0;
virtual T getLast() = 0;
virtual T removeLast() = 0;
virtual void insertLast(T element) = 0;
```



Example code: (not what we would do)

```
LinkedListNode<int> * n1 = new LinkedListNode<int>();
LinkedListNode<int> * n2 = new LinkedListNode<int>();
n1->next = n2;
n1->data = 4;
n2->next = nullptr;
n2->data = 7;
```

```
template <typename T>
class LinkedListNode {
  public:
    T data;
    LinkedListNode<T> * next;
};
```




```
template <typename T>
class LinkedList : public List<T> {
  private:
    LinkedListNode<T>* head;
  public:
    int size;
};
```

encapsulation

Invariant: something which is "always" true

length of chain starting at "head" is equal to "size"

O(1)

O(n)

Method getFirst():

```
If this->head ≠ nullptr Then
    Return this->head->data
Else:
    (!!)
End
```

EndMethod

Method getSize():

```
num ← 0
current ← this->head
While current ≠ nullptr:
    num ← num + 1
    current ← current->next
End While
Return num
```

EndMethod

Method insertFirst():

```
make new node
put at front of list
make next point to old front
increment size
```

EndMethod

Method getSize()

```
Return this->size
```

EndMethod