EINFÜGEN IN SORTIERTE LISTE

Einfügen

```
// pre: verkettete Liste, erstes Element first
// post: value in Liste aufsteigend sortiert eingefügt
public void Insert (int value)
{
// Wenn Liste leer, dann first = neues Element ohne Nachfolger
// sonst suche Einfügeposition und füge ein
}
```

Einfügen

```
public void Insert (int value)
{
    if (first == null) {
        first = new Node(value, null);
    }
    else {
        // suche Einfügeposition und füge ein.
    }
}
```

Einfügen

```
public void Insert (int value)
{
     if (first == null) {
          first = new Node(value, null);
     }
     else {
          if (value <= first.value) // Fall a
               first = new Node(value, first);
          else { // Fälle b und c
               Value v = first;
               while (v != null && value > v.value) // Suche Nachfolgerknoten
                    v = v.next;
               Value n = new Node(value, v); // Neuer Knoten mit diesem Nachfolger, v= null möglich
               // Einfügen dieses Knotens nach Vorgänger von v
          }
     }
}
```

Einfügen

```
public void Insert (int value)
{
	if (first == null) {
		first = new Node(value, null);
	}
	else {
		if (value <= first.value) // Fall a
			first = new Node(value, first);
		else { // Fälle b und c
			Node prev = first; // != null
			Node v = prev.next;
			while (v != null && value > v.value){ // Suche Nachfolgerknoten
				prev = v;
				v = v.next;
			}
			Value n = new Node(value, v); // Neuer Knoten mit diesem Nachfolger, v= null möglich
			prev.next = v;
		}
	}
}
```

Vereinfachen, Testen, fertig.

```
public void Insert (int value)
{
    if (first == null || value <= first.value)
        first = new Node(value, first);
    else {
        Node prev = first;
        Node v = prev.next;
        while (v != null && value > v.value){ // suche Nachfolger und Vorgänger
            prev = v;
            v = v.next;
        }
        prev.next = new Node(value, v); // Einfügen
    }
}
```

Alternative

```
public void Insert (int value)
{
    if (first == null || value <= first.value)
        first = new Node(value, first);
    else {
        Node v = first; // v spielt die Rolle des Vorgängers
        while (v.next != null && value > v.next.value){
            v = v.next;
        }
        v.next = new Node(value, v); // Einfügen
    }
}
```