Lists (continue)

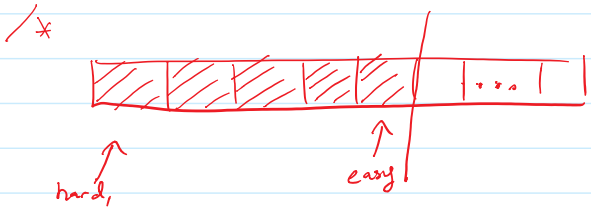
```
public class ArrayList<E> implements List<E> {

    public void insert (...) {
        :  // Exceptions
    }

    private void expandArray() {
        :
    }

    public void append (E elem) {
        insert(size, elem);
    }

    public E get(int index) throws ... // Compile time {
        if ( index < 0 || index >= size) {
            throw new ListIndexOutOfBoundsException(); //Implicitly
                                                       //a runtime exception
        }
```

```
int[] arr = ...;
for (int i=-1; i < arr.length; i++) {

    arr[i] = 0;  // ArrayIndexOutOfBounds...
}                    ↑
                     RuntimeException since no try/catch


try {
  Scanner scan = new Scanner(new File("someFile.txt"));
} catch (...) {  // FileNotFoundException

}

  for (int i=0; i< arr.size; i++) {
      try {
          arr[i] = 0;  // list.get(i);
      } catch (ArrayIndexOutOfBoundsException e) {
          :

      }
  }
```

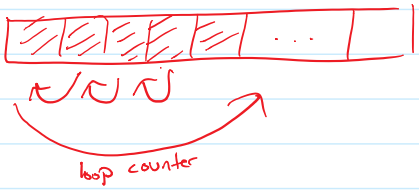
```
public E get(int index) {
    if (index < 0 || index >= size) {
        throw new ListIndexOutOfBoundsException();
    }
    return convertElement(arr[index]);
}
```

convertElement: f: Obj → E, arr[index]: Object

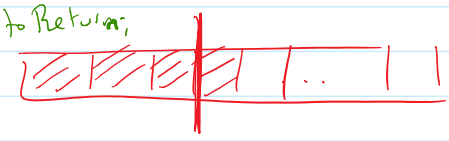
```
public E remove(int index) {
    E toReturn = get(index); //possible exception
```



loop

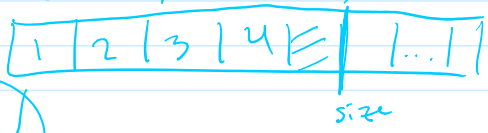



```
*/ size--
    for (int i=index; i<size; i++) {
        theArray[i] = theArray[i+1];
    }
    return toReturn;
}
```

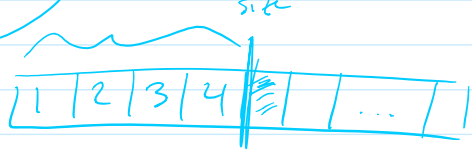


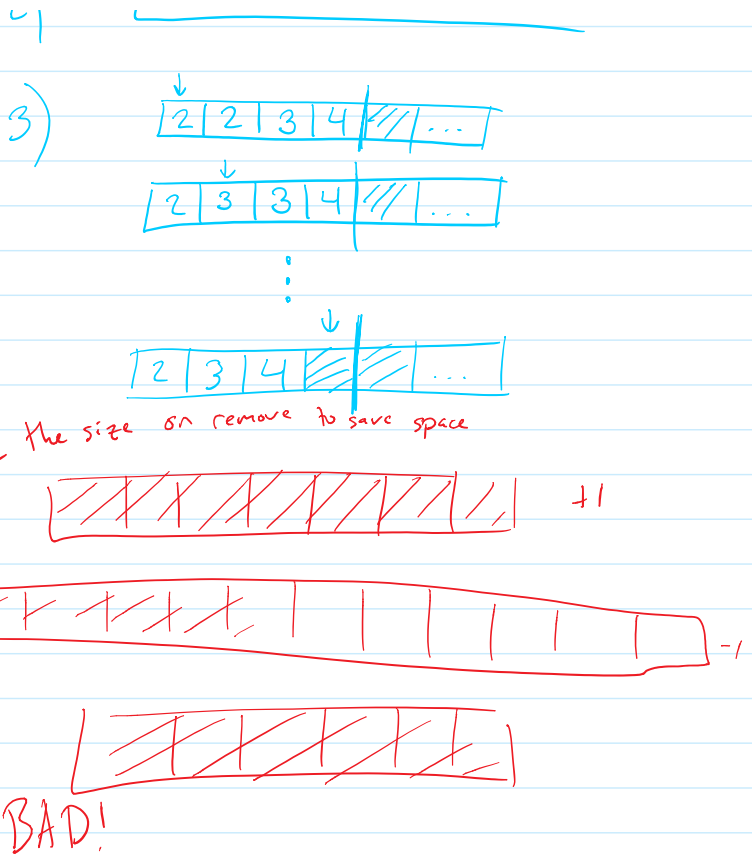
Example size is 5, remove(0)

1)



2)





Instead, choose 1/4 → Probably more efficient

```
public int size() {
        return size;
}
```

Pros & Cons:

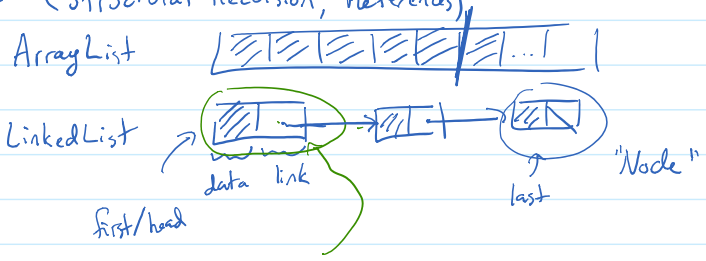
Pro:

Random Access

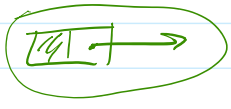
Con:

Too much space

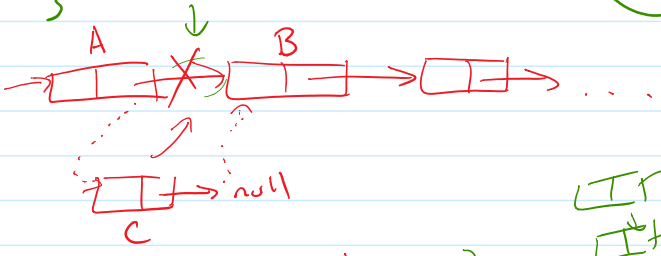
Linked List (Structural Recursion, References)




```
public class LinkedList<E> implements List<E> {
    private class Node<E> {
        public E element;
        public Node<E> next;
        : // Constructor ?
    }
```



```
: // Constructor :
}

private Node<E> head;
private int size;

public LinkedList() {
    head = null;
    size = 0;
}

public void insert(int index, E elem) {   // elem: null or otherwise
    if (head == null) {                   // Error checking
        head = new Node<E>();
        head.element = elem;
        head.next = null;
    }
    Node<E> iter = head;
    for (int i=0; i < index; i++);
        iter = iter.next;
    }
    Node<E> newNode = new Node<E>;
    newNode.element = elem;
    newNode.next = iter.next;
    iter.next = newNode;
}
```




```
C.next = A.next
A.next = C
```