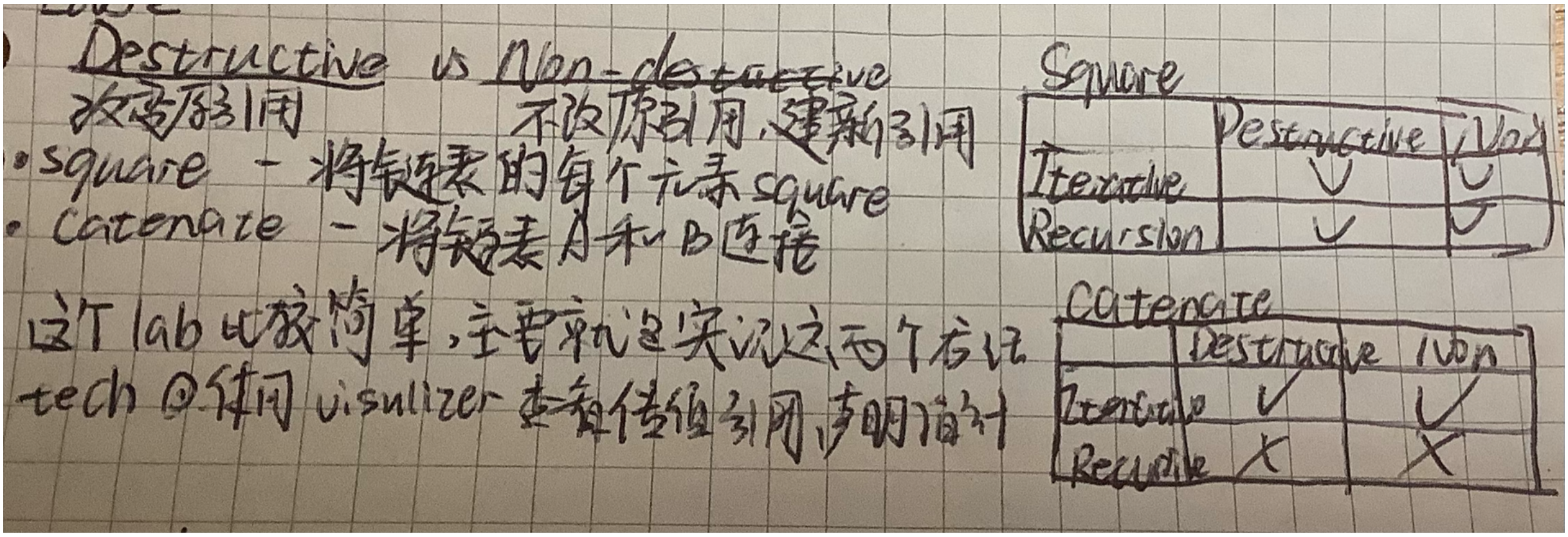
LinkedList

Destructive 改变原引用
Non destructive 不改变原引用



Square

```java
/**
 * Returns a list equal to L with all elements squared. Destructive.
 */
public static void dSquareList(IntList L) {

    while (L != null) {
        L.first = L.first * L.first;
        L = L.rest;
    }
}

public static IntList dSquareListRecursive(IntList L){
    if (L == null){
        return L;
    }else {
        L.first *= L.first;
        return dSquareListRecursive(L.rest);
    }

}
```

```java
/**
 * Returns a list equal to L with all elements squared. Non-destructive.
 */
public static IntList squareListIterative(IntList L) {
    if (L == null) {
        return null;
    }
    IntList res = new IntList(L.first * L.first, null);
    IntList ptr = res;
    L = L.rest;
    while (L != null) {
        ptr.rest = new IntList(L.first * L.first, null);
        L = L.rest;
        ptr = ptr.rest;
    }
    return res;
}

/**
 * Returns a list equal to L with all elements squared. Non-destructive.
 */
public static IntList squareListRecursive(IntList L) {
    if (L == null) {
        return null;
    }
    return new IntList(L.first * L.first, squareListRecursive(L.rest));
}
```

Catenate

```java
/**
 * Returns a list consisting of the elements of A followed by the
 * *  elements of B.  May modify items of A. Don't use 'new'.
 */

public static IntList dcatenate(IntList A, IntList B) {
    //TODO:  fill in method


    IntList ptra = A;
    while (ptra.rest != null){
        ptra = ptra.rest;
    }
    ptra.rest = B;
    return A;
```

```java
/**
 * Returns a list consisting of the elements of A followed by the
 * * elements of B.  May NOT modify items of A.  Use 'new'.
 */
public static IntList catenate(IntList A, IntList B) {
    //TODO:  fill in method

    /*
   if (A.rest == null) {
       return new IntList(A.first, B);
   }else {
       return new IntList(A.first, catenate(A.rest,B));
   }
     */

    IntList res = new IntList(A.first, null);
    IntList ptr = res;
    A = A.rest;

    while (A != null){
        ptr.rest = new IntList(A.first, null);
        A = A.rest;
        ptr = ptr.rest;
    }

    ptr.rest = B;
    return res;



}
```

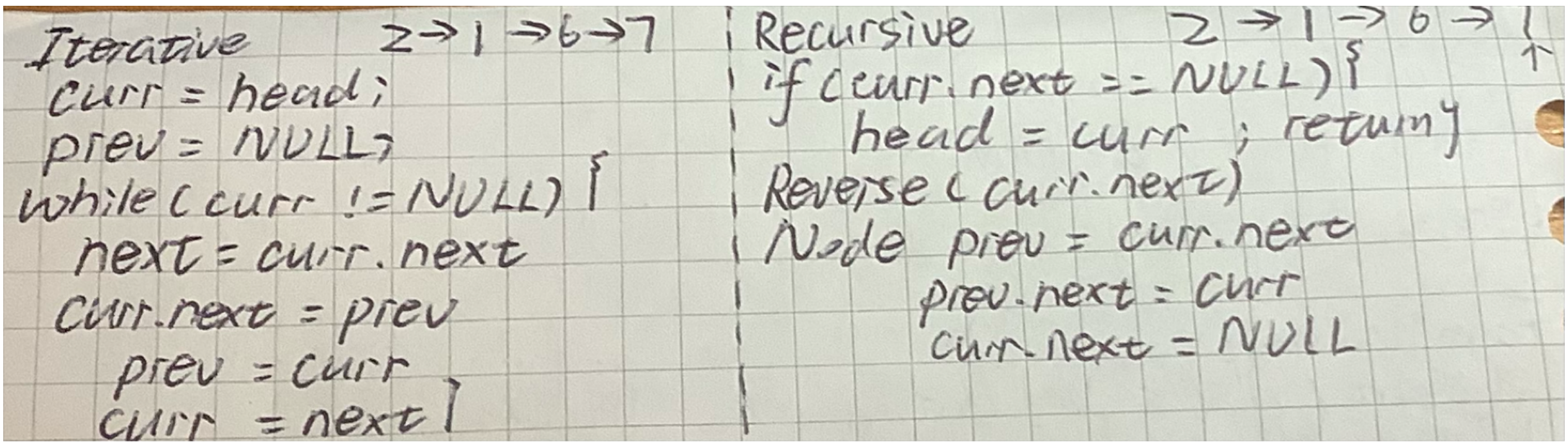
Sp18 dis03
Insert

For example, if the `SLList` is $5 \to 6 \to 2$, `insert(10, 1)` results in $5 \to 10 \to 6 \to 2$.

```
public void insert(int item, int position) {

    if (first == null || position == 0) {
        addFirst(item);
        return;
    }
    IntNode currentNode = first;
    while (position > 1 && currentNode.next != null) {
        position--;
        currentNode = currentNode.next;
    }
    IntNode newNode = new IntNode(item, currentNode.next);
    currentNode.next = newNode;
```

Reverse



Sp18 preExam

Remove duplicate

```
/**
* Given a sorted linked list of items - remove duplicates.
* For example given 1 -> 2 -> 2 -> 2 -> 3,
* Mutate it to become 1 -> 2 -> 3 (destructively)
*/
public static void removeDuplicates(IntList p) {
    if (p == null) {
        return;
    }

    IntList current = p.rest;
    IntList previous = p;
    while (current != null) {
        if (current.first == previous.first) {
            previous.rest = current.rest;
        } else {
            previous = current;
        }
        current = current.rest;
    }
}
```

Skippify

```
IntList A = IntList.list(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
IntList B = IntList.list(9, 8, 7, 6, 5, 4, 3, 2, 1);
```

Fill in the method `skippify` such that the result of calling `skippify` on A and B are as below:

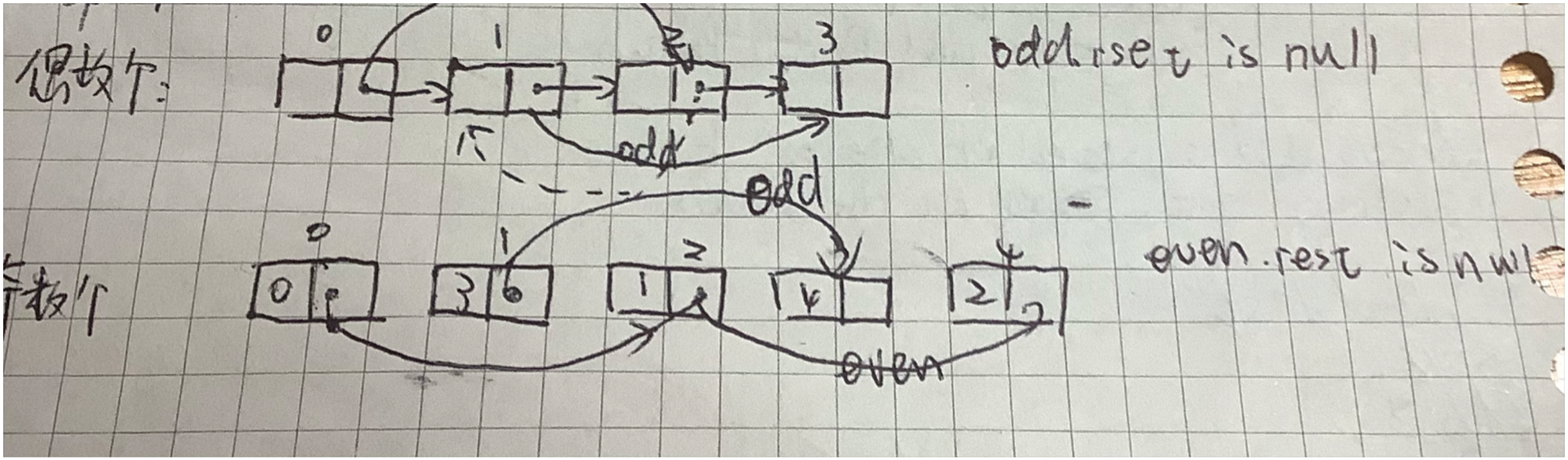
- After calling `A.skippify()`, A: (1, 3, 6, 10)
- After calling `B.skippify()`, B: (9, 7, 4)

```
public void skippify() {
    IntList p = this;
    int n = 1;
    while (p != null) {
        IntList next = p.rest;
        for (int i = 0; i < n; i += 1) {
            if (next == null) {
                break;
            }
            next = next.rest;
        }
        p.rest = next;
        p = p.rest;
        n++;
    }
}
```

Sp21 preExam

Even Odd

For instance, if `lst` is defined as `IntList.list(0, 3, 1, 4, 2, 5)`, `evenOdd(lst)` would modify `lst` to be `IntList.list(0, 1, 2, 3, 4, 5)`.




```
public static void evenOdd(IntList lst) {
    if (lst == null || lst.rest == null) {
        return;
    }
    IntList oddList = lst.rest;
    IntList second = lst.rest;
    while (lst.rest != null && oddList.rest != null) {
        lst.rest = lst.rest.rest;
        oddList.rest = oddList.rest.rest;
        lst = lst.rest;
        oddList = oddList.rest;
    }
    lst.rest = second;
}
```