```
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (int i = 0; i < x.length; i++) {
        totalLength += x[i].length;
    }
    int[] a = newint[totalLength];
    int aIndex = 0;
    for (int i = 0; i < x.length; i++) {
        for (int j = 0; j < x[i].length; j++) {
            a[aIndex] = x[i][j];
            aIndex++;
        }
    }
    return a;
}
```

CS 61B Spring 2019

Linked Lists, Arrays

Exam Prep 3: Feb 4, 2019

1 Flatten

Write a method `flatten` that takes in a 2-D array x and returns a 1-D array that contains all of the arrays in x concatenated together.

For example, `flatten({{1, 2, 3}, {}, {7, 8}})` should return `{1, 2, 3, 7, 8}`. (Summer 2016 MT1)

```
1  public static int[] flatten(int[][] x) {
2      int totalLength = 0;
3
4      for (_int i=0;i<x.length;i++__) {
5
6          totalLength+=x[i].length;
7      }
8
9      int[] a = new int[totalLength];
10     int aIndex = 0;
11
12     for (_int i=0;i<x.length;i++___) {
13
14         if(x[i].length==0)
15             continue;
16
17
18         for(int j=0;j<x[i].length;j++)
19
20             a[aIndex++] = x[i][j];
21     }
22
23     return a;
24 }
```

loop然後算出totalLength

超進去

沒必要

2 Skippify

Suppose we have the following `IntList` class, as defined in lecture and lab, with an added `skippify` function.
Suppose that we define two IntLists as follows.

```
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)
(Spring '17, MT1)

先空1格，在空2->3

destructive

```
public class IntList {
    public int first;
    public IntList rest;

    @Override
    public boolean equals(Object o) { ... }
    public static IntList list(int... args) { ... }

    public void skippify() {
        IntList p = this;
        int n = 1;
        while (p != null) {
            IntList next = p.rest;

            for (int i=0;;i++) {

                if (                                    ) {
                    next.res =
                }

                p=p.rest;
            }

                                                    
                rest = next;
                                                    
                n++
                                                    
        }
    }
}
```

8->7...->1

A數字小抓數字大

判斷兩objet相等

nstructor)

this=A=1

count

ptr

空格數=n

通過這個i

這裡還在while迴圈

上面while條件跟ptr的變動是一致的(如果是ptr=ptr.rest=>就要驗證ptr是否為null)

3 Sans

Fill in the blanks below to correctly implement `ilsans` and `dilsans`.
(Spring '18, MT1)

```
1  public class IntList {
2      public int first;
3      public IntList rest;
4      public IntList (int f, IntList r) {
5          this.first = f;
6          this.rest = r;
7      }
8
9      /** Non-destructively creates a copy of x that contains no occurences of y.*/
10     public static IntList ilsans(IntList x, int y) {
11       if (__x==null_____) {
12         return __null;
13       }
14       if (__x.first==y____) {
15         return _new IntList(x.rest.first,ilsans(x.rest.rest,i)
16       }
17       return new IntList(x.first,ilsans(x.rest,i))
18     }
19
20     /** Destructively modify and return x to contain no occurences of y,
21       without using the keyword "new". */
22     public static IntList dilsans(IntList x, int y) {
23       if (_x==null_____) {
24         ___return null_____;
25       }
         x.rest = dilsans(x.rest,y);
         if (x.first == y) {
           return __x.rest_____;
         }
         return ___x_____;
       }
```

所有node都要抄新的!!!

把y拔掉

empty IntList

recursive?就不用ptr loop thru

會近來這一定是x.first!=y所以不用再判斷

照抄一個新頭!

x.first跟人家有點不一樣

條件達成

所有人先走到最後面再去做檢查

不用再去後面了，只要牽上就好

牽上

條件不達成

這裡是x.first!=y，且是recursieve主要動力