

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     for (int i = 0; i < x.length; i++) {
4         totalLength += x[i].length;
5     }
6     int[] a = new int[totalLength];
7     int aIndex = 0;
8     for (int i = 0; i < x.length; i++) {
9         for (int j = 0; j < x[i].length; j++) {
10             a[aIndex] = x[i][j];
11             aIndex++;
12         }
13     }
14     return a;
15 }
```

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.

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

```
1 public class IntList {
2     public int first;
3     public IntList rest;
4
5     @Override
6     public boolean equals(Object o) { ... } 这是一个重写的 equals 方法，用于比较两个对象是否相等。
7     public static IntList list(int... args) { ... }
8                                     这是一个静态方法，用于创建并返回一个 IntList 对象。
9     public void skippify() {
10                                     它接受可变数量的整数参数，并将它们转换为一个链表。
11         IntList p = this;
12         int n = 1; 是一个计数器，表示当前要跳过的元素数目
13         while (p != null) {
14             IntList next = p.rest;
15             for (int i = 0; i < n; i += 1) {
16                 if (next == null) {
17                     break;
18                 }
19                 next = next.rest; 将 next 移动到下一个要保留的节点
20             }
21             p.rest = next;
22             p = p.rest; 将 p 移动到下一个节点，以继续下一轮循环；p就是我们要的
23             n++;
24         }
25     }
26 }
```

3 Remove Duplicates

Fill in the blanks below to correctly implement `removeDuplicates`.
(Spring '17, 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      /**
10     * Given a sorted linked list of items - remove duplicates.
11     * For example given 1 -> 2 -> 2 -> 2 -> 3,
12     * Mutate it to become 1 -> 2 -> 3 (destructively)
13     */
14     public static void removeDuplicates(IntList p) {
15         if (p == null) {
16             return;
17         }
18
19         IntList current = p.rest; current 初始化为链表的第二个节点，即跳过头节点
20         IntList previous = p; previous 初始化为链表的头节点，用于跟踪前一个不重复的节点
21         while (current != null) {
22             if (current.first == previous.first) {
23                 previous.rest = current.rest;
24             } else {
25                 previous = current;
26             }
27             current = current.rest;
28         }
29     }
30 }
```