

## 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 (-----) {
5
6         -----
7     }
8
9     int[] a = new int[totalLength];
10    int aIndex = 0;
11
12    for (-----) {
13
14        -----
15
16        -----
17
18        -----
19
20        -----
21    }
22
23    return a;
24 }
```

## 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) { ... }
7     public static IntList list(int... args) { ... }
8
9     public void skippify() {
10        IntList p = this;
11        int n = 1;
12        while (p != null) {
13
14            IntList next = _____;
15
16            for (_____ ) {
17
18                if (_____ ) {
19
20                    _____
21                }
22
23                _____
24            }
25
26            _____
27
28            _____
29
30            _____
31        }
32    }
33 }
```

### 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 occurrences of y.*/
10     public static IntList ilsans(IntList x, int y) {
11         if (_____ ) {
12             return _____;
13         }
14         if (_____ ) {
15             return _____;
16         }
17         return new _____;
18     }
19
20     /** Destructively modify and return x to contain no occurrences of y,
21         without using the keyword "new". */
22     public static IntList dilsans(IntList x, int y) {
23         if (_____ ) {
24             _____;
25         }
26         _____;
27         if (x.first == y) {
28             return _____;
29         }
30         return _____;
31     }
32 }

```