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)

```java
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (____________________________________) {
        __________________________________________
    }
    int[] a = new int[totalLength];
    int aIndex = 0;
    for (____________________________________) {
        __________________________________________
        __________________________________________
        __________________________________________
        __________________________________________
    }
    return a;
}
```
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.

```java
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)

```java
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 = __________________________;
            for (_______________________________________) {
                if (________________________________________) {
                    _____________________________
                }
            }
        }
    }
}
```
3 Even Odd

Implement the method evenOdd by destructively changing the ordering of a given IntList so that even indexed links precede odd indexed links.
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).

Hint: Make sure your solution works for lists of odd and even lengths.

```java
public class IntList {
    public int first;
    public IntList rest;
    public IntList (int f, IntList r) {
        this.first = f;
        this.rest = r;
    }

    public static void evenOdd(IntList lst) {
        if (________________________________________________) {
            return;
        }

        IntList second = __________________;
        int index = __________________;  
        while (____________________________________________) {
            ____________________________________________
            ____________________________________________
            ____________________________________________
            ____________________________________________
            ____________________________________________
        }
    }
}
```