CS 61B Spring 2018

Linked Lists, Arrays

Exam Prep 3: January 22, 2018

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)

```
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 Remove Duplicates

Fill in the blanks below to correctly implement `removeDuplicates`.
(Spring '17, MT1)

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

    /**
    * 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 = _________________________;

        IntList previous = ________________________;

        while (__________________________________) {

            if (_____________________________________) {

                ________________________________________
            } else {

                ________________________________________
            }

            ____________________________________________
        }
    }
}
```