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)

```
1  public static int[] flatten(int[][] x) {
2      int totalLength = 0;
3
4      for (int[] row : x                    ) {
5
6          totalLength += row.length;
7      }
8
9      int[] a = new int[totalLength];
10     int aIndex = 0;
11
12     for (int[] row : x                    ) {
13
14         for (int i : row){
15
16             a[aIndex] = i;
17
18             aIndex += 1;
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.

```
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)

```
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<n, i++__) {

                if (____next == null____) {

                    ____break;____

                }

                ____next = next.rest____

            }

            ____p.rest = next;____

            ____p = next;____

            ____n += 1;____

        }
    }
}
```

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____;
20
21         IntList previous = ____p____;
22
23         while (current != null) {
24
25             if (current.first == current.first) {
26
27                 previous.rest = current.rest
28             } else {
29
30                 previous = current
31             }
32
33             current = current.rest
34         }
35     }
36 }
```