CS 61B Fall 2020

Arrays, Linked Lists

Exam Prep Discussion 3: Sept 7, 2020

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[] xRows : x                    ) {
5          totalLength += xRows.length;
6      }
7
8      int[] a = new int[totalLength];
9      int aIndex = 0;
10     for (int[] xRows : x                    ) {
11         for (int xRowValue : xRows) {
12         a[aIndex] = xRowValue;
13         aIndex += 1;
14         }
15     }
16
17     return a;
18 }
```

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.

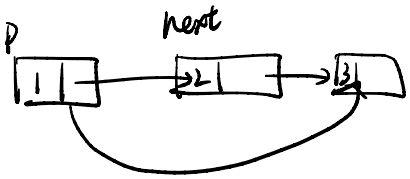
```
IntList A = IntList.list(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
IntList B = IntList.list(9, 8, 7, 6, 5, 4, 3, 2, 1);
```

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 index=0; index < n; index++) {

                if (next != null) {

                    p.rest = next.rest
                }

                next = next.rest
            }

            p = p.rest

            n++;
        }
    }
}
```



next == null

break;

p.rest = next

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.

```
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 (_lst.rest == null || lst==null_______) {
        return;
      }

      IntList second = _lst.rest_________;

      int index = ___0___________________;

      while (_____________________________________) {
          Intlist temp = lst.rest    //防止数据丢失
          lst.rest = lst.rest.rest
          lst = temp;
          index ++ ;
      }
      lst.rest = second
    }
}
```

lst==null 要最先判断，因为如果先判断 lst.rest 可能报错空指针，因为lst是空

|| lst.rest.rest == null)

0 used to help find the last even

在到 last even link 时结束循环

lst.rest 向后两位，跳过中间，直接destructively

lst 向后1位

index % 2 == 0 && (lst.rest == null || lst.rest.rest == null)

even index

last element

second to last