CS 61B Spring 2017

Arrays, Recursion, & Pointers

Week 3: January 30, 2017

1 Arrays

1.1 Define a function, `squareSum`, that squares every element in an `int[] values` and returns the sum of all the squared values.

```
public class FunWithArrays {
    public static int squareSum(int[] values) {
        int sum = 0;
        for (int i = 0; i < values.length; i++) {
            values[i] *= values[i];
            sum += values[i];
        }
        return sum;
    }
}
```

1.2 Provide a descriptive name for each of the following methods. Assume that `values` contains at least one element.

```
private static boolean ____________________________ (int[] values) {
    int k = 0;
    while (k < values.length-1) {
        if (values[k] > values[k+1]) {
            return false;
        }
        k = k+1;
    }
    return true;
}
```

`isIncreasing`

```
private static void _______________________________ (int[] values) {
    int k = 0;
    while (k < values.length/2) {
        int temp = values[k];
        values[k] = values[values.length-1-k];
        values[values.length-1-k] = temp;
        k = k+1;
    }
}
```

`reverse`

2 Recursion

2.1 A classic puzzle called the Towers of Hanoi is a game that consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with `n` disks in a neat stack in ascending order of size on a `start` rod with the smallest disk at the top.

The objective of the puzzle is to move the entire stack to an `end` rod, obeying the following rules:

- Only one disk may be moved at a time.
- Each move consists of taking the top (smallest) disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.
- No disk may be placed on top of a smaller disk.

Complete the definition of `moveStack`, which prints out the steps required to move `n` disks from the `start` rod to the `end` rod without violating the rules.

```
public class Towers {

    /** Print instructions to move a disk. */
    public static void printMove(int start, int end) {
        System.out.println("Move the top disk from rod " + start + " to rod " + end);
    }

    /** Print the moves required to move n disks on the start pole to the end
    pole without violating the rules of Towers of Hanoi. */
    public static void moveStack(int n, int start, int end) {
        assert 1 <= start && start <= 3 && 1 <= end && end <= 3 && start != end;
        if (n == 1) {
            printMove(start, end);
        } else {
            int other = 6 - start - end;
            moveStack(n - 1, start, other);
            printMove(start, end);
            moveStack(n - 1, other, end);
        }
    }
}
```

3 Pointers

3.1 Draw a box-and-pointer diagram for the following program.

```
public class IntList {
    int first;
    IntList rest;

    public IntList(int first, IntList rest) {
        this.first = first;
        this.rest = rest;
    }

    public static void main(String[] args) {
        IntList N = new IntList(1, null);
        IntList M = new IntList(2, N);
        N.rest = M;
        M.rest.rest.rest = new IntList(3, N);
        N = N.rest;
    }
}
```