CS 61B Discussion 5 Spring 2015

1 Counting Stars

Given a `String[] args`, write a method that counts the number of appearances of `"star"`, case sensitive. If this number is even, simply return `args`. Otherwise, return a `String[]` containing all the non-"star" entries (including `null` entries). Your code shouldn't error for any input, you may not use the modulo (`%`) operator, and you are not allowed to take more than one pass through the input `array`.

A potentially helpful method: `Arrays.copyOf(String[] array, int newLength)` - Copies the specified array, truncating or padding with nulls as necessary so the returned copy has the specified length. For example, if `orig` is `{"a", "2", "3"}`, `Arrays.copyOf(orig, 2)` is `{"a", "2"}` and `Arrays.copyOf(orig, 4)` is `{"a", "2", "3", null}`.

```java
import java.util.Arrays;
public static String[] starCount(String[] args) {
        // Check for null and trivial case.
        if (args == null || args.length == 0) {
                return args;
        }

        int numStars = 0;
        String[] bigArr = new String[args.length];
        int nextSlot = 0;
        for (int loc = 0; loc < args.length; loc++) {
                // Account for null entries with versatile checks.
                if ("star".equals(args[loc])) {
                        numStars++;
                } else {
                        bigArr[nextSlot] = args[loc];
                        nextSlot++;
                }
        }
        if (isOdd(numStars)) {
                return Arrays.copyOf(bigArr, nextSlot);
        } else {
                return args;
        }

        // Returns if x is odd using bit ops
        private static boolean isOdd(int x) {
                return (x & 1) != 0;
        }
}
```

2 HugString: Part 1 of 2

You will be helping Josh convert an input `String` to a singly-linked list of `char`'s. You'll first need the building blocks: your linked nodes.

```java
class CNode {
        char head;
        CNode next;
        public CNode(char head, CNode next) {
                this.head = head;
                this.next = next;
        }
}
```

3 HugString: Part 2 of 2

Now implement the method that makes the HugString. You may want to use the `String.charAt(int loc)` method, which returns the character at position `loc`. For example, `"hey".charAt(0)` returns `'h'`.

```java
/** Converts the input String s into a linked list of CNodes
    and returns the head of the list. */
public static CNode makeHugString(String s) {
        // Check null FIRST, then check length.
        if (s == null || s.length() == 0) {
                return null;
        }

        CNode answer = new CNode(s.charAt(0), null);
        CNode curr = answer;

        for (int loc = 1; loc < s.length(); loc++) {
                curr.next = new CNode(s.charAt(loc), null);
                curr = curr.next;
        }

        return answer;
}
```

4 HugString: Part 3 of 2 (Additional for Aces)

Building off of your code base from above, write a method `swapSpace` to destructively replace every space (`" "`) in an input `CNode` linked-list with `"61B"`. The input is the first `CNode` node.

```java
public void swapSpace(CNode in) {
        while (in != null) {
                if (in.head == ' ') {
                        // Just swap the space for a '6'
                        in.head = '6';

                        // Insert a 'B' after '6'
                        in.next = new CNode('B', in.next);

                        // Insert a '1' after '6'
                        in.next = new CNode('1', in.next);

                        // Hop it forward
                        in = in.next.next;
                }
                in = in.next;
        }
        /** NOTE
          * It's okay to directly modify the in pointer because methods
          * always get passed a copy of its parameters. If you get passed
          * a primitive, it's a copy. If you get passed an Object pointer,
          * it's also a copy, except it also points to the original thing,
          * so edits to the Object's instance variables will last.
          */
}
```