Quiz 25 SOLUTIONS

CSCI 110 Section 1

Wednesday, November 9, 2016

1) Using a HashMap, write a function to determine whether a String has balanced parentheses. For example, the String “((he)llo)” does while the String “(ok))” does not. Hint: also consider that the String “))test((“ is not balanced since there are insufficient opening parentheses before the closing parentheses. [60 points]

Psuedocode:

- Create an HashMap with two entries, one for opening parenthesis and one for closing
- Go through the string, incrementing the count for any opening and closing parenthesis we see
    - If we see that the number of opening parentheses isn't enough for a closing parenthesis to be there, we can immediately return false
- After going through the String, if the counts are equal, then the parentheses must have been balanced

```
boolean isBalanced(String s) {
    HashMap<Character, Integer> counts =
        new HashMap<Character, Integer>();
    counts.put('(', 0);
    counts.put(')', 0);
    for (int i = 0; i < s.length(); i++) {
        char c = s.getChar(i);
        if (c == ')') {
            if (counts.get(')') >= counts.get('(')) {
                // There weren't enough opening parentheses before
                // this closing one
                return false;
            }
            counts.put(c, counts.get(c) + 1);
        } else if (c == '(') {
            counts.put(c, counts.get(c) + 1);
        }
    }
    return counts.get('(') == counts.get(')');
}
```

2) Without using a HashMap or any other data structure, write a function to determine whether a String has balanced parentheses. [40 points]

This solution adds one to a variable for every opening parenthesis it sees, and subtracts one for every closing one it sees. If the counter is 0 at the end, that means the parenthesis must have been balanced. However, the counter must never dip below 0.

```
boolean isBalanced(String s) {
    int count = 0;
    for (int i = 0; i < s.length(); i++) {
        char c = s.getChar(i);
        if (c == ')') {
            if (count <= 0) {
                // There weren't enough opening parentheses before
                // this closing one
                return false;
            }
            count--;
        } else if (c == '(') {
            count++;
        }
    }
    return count == 0;
}
```

3) Using a Stack, write a function to determine whether a String has balanced parentheses. [extra credit, 40 points]

Space complexity can also be described using Big-O notation. Unlike the first two solutions which each take O(1) space - the HashMap always has two entries, and an int is always an int - this one is less efficient and takes O(n) space. For example, a String like “((((((“ will get all of its characters pushed onto the stack.

All three solutions take O(n) time since each has to iterate through the entire string.

```
boolean isBalanced(String s) {
    Stack<Character> parens = new Stack<Character>();
    for (int i = 0; i < s.length(); i++) {
        char c = s.getChar(i);
        if (c == ')') {
            if (parens.empty()) {
                // There weren't enough opening parentheses before
                // this closing one
                return false;
            }
            parens.pop(c);
        } else if (c == '(') {
            parens.push(c);
        }
    }
    return parens.empty();
}
```