Mock Interviews 1A

DCSC

November 4 2016

1 ANAGRAM

```
    Solution #1: Sort the strings

boolean anagram(String s, String t) {
return sort(s) == sort(t);
}
Solution #2: Check if the two strings have identical counts for each unique char.

public static boolean anagram(String s, String t)
{

    if (s.length() != t.length()) return false;

    int[] letters = new int[256];

    int num_unique_chars = 0;

    int num_completed_t = 0;

    char[] s_array = s.toCharArray();

    for (char c : s_array)
    { // count number of each char in s.

        if (letters[c] == 0) ++num_unique_chars;

    ++letters[c];

    }

    for (int i = 0; i < t.length(); ++i)
    {

        int c = (int) t.charAt(i);
```

```
        if (letters[c] == 0)
        { // Found more of char c in t than in s.

            return false;

        }

        letters[c];

        if (letters[c] == 0)
        {

            ++num_completed_t;

            if (num_completed_t == num_unique_chars)
            {

                // it's a match if t has been processed completely

                return i == t.length()  1;

            }

        }

    }

    return false;
}
```

2 REVERSE SUM

```
LinkedListNode addLists(LinkedListNode 11, LinkedListNode 12, int carry)
{
    /* We're done if both lists are null AND the carry value is 0 */

    if (11 == null && 12 == null && carry == 0)
        return null;

    LinkedListNode result = new LinkedListNode(carry, null, null);

    /* Add value, and the data from 11 and 12 */
    int value = carry;
```

```
    if (l1 != null)
        value += l1.data;
    if (l2 != null)
        value += l2.data;


    result.data = value % 10; /* Second digit of number */


    /* Recurse */
    if (l1 != null || l2 != null)
    {
        LinkedListNode more = addLists(l1 == null ? null : l1.next,
        l2 == null ? null : l2.next,
        value >= 10 ? 1 : 8);

        result.setNext(more);
    }
    return result;
}
```