A World Without Strings is Chaos

John Earnest

June 2, 2017

1. **Multiplicity**

   Characters are expensive, and the accountants tell me we can't hand them out willy-nilly anymore. Given a string `x` and a character `y`, how many times does `y` occur in `x`?

```
  f["fhqwhgads";"h"]
2
  f["mississippi";"s"]
4
  f["life";"."]
0
```

2. **Trapeze Part**

   Sometimes I try reading sentences right-to-left to make my life more exciting. Results have been mixed. Given a string `x`, is it identical when read forwards and backwards?

```
  f "racecar"
1
  f "wasitaratisaw"
1
  f "palindrome"
0
```

3. **Duplicity**

   One is the loneliest number. Given a string `x`, produce a list of characters which appear more than once in `x`:

```
  f "applause"
"ap"
  f "foo"
,"o"
  f "baz"
""
```

4. **Sort Yourself Out**

   Alphabetical filing systems are passé. It's far more Zen to organize words by their histograms! Given strings `x` and `y`, do both strings contain the same letters, possibly in a different order?

```
  f["teapot";"toptea"]
1
  f["apple";"elap"]
0
  f["listen";"silent"]
1
```

5. **Precious Snowflakes**

   It's virtuous to be unique, just like everyone else. Given a string `x`, find all the characters which occur exactly once, in the order they appear.

```
  f "somewhat heterogenous"
"mwa rgnu"
  f "aaabccddefffgg"
"be"
```

6. **Musical Chars**

   Imagine four chars on the edge of a cliff. Say a direct copy of the char nearest the cliff is sent to the back of the line of char and takes the place of the first char. The formerly first char becomes the second, the second becomes the third, and the fourth falls off the cliff. Strings work the same way. Given strings `x` and `y`, is `x` a *rotation* of the characters in `y`?

```
  f["foobar";"barfoo"]
1
  f["fboaro";"foobar"]
0
  f["abcde";"deabc"]
1
```

7. **Size Matters**

   Sometimes small things come first. Given a list of strings `x`, sort the strings by length, ascending:

```
  f ("books";"apple";"peanut";"aardvark";"melon";"pie")
("pie"
 "books"
 "apple"
 "melon"
 "peanut"
 "aardvark")
```

8. **Popularity Contest**

   Sixty-two thousand four hundred repetitions make one truth. Given a string `x`, identify the character which occurs most frequently. If more than one character occurs the same number of times, you may choose arbitrarily. Is it harder to find all such characters?

```
  f "abdbbac"
"b"
  f "CCCBBBAA"
"C"
  f "CCCBBBBAA"
"B"
```

9. **esreveR A ecnetneS**

   Little-endian encoding is such a brilliant idea I want to try applying it to English. Given a string `x` consisting of words (one or more non-space characters) which are separated by spaces, reverse the order of the characters in each word.

```
  f "a few words in a sentence"
"a wef sdrow ni a ecnetnes"
  f "zoop"
"pooz"
  f "one two three four"
"eno owt eerht ruof"
```

10. **Compression Session**

    Let's cut some text down to size. Given a string `x` and a boolean vector `y` of the same length, extract the characters of `x` corresponding to a 1 in `y`.

```
  f["foobar";1 0 0 1 0 1]
"fbr"
  f["embiggener";0 0 1 1 1 1 0 0 1 1]
"bigger"
```

11. **Expansion Mansion**

    Wait, strike that- reverse it. Given a string `x` and a boolean vector `y`, spread the characters of `x` to the positions of 1s in `y`, filling intervening characters with underscores.

```
  f["fbr";1 0 0 1 0 1]
"f__b_r"
  f["bigger";0 0 1 1 1 1 0 0 1 1]
"__bigg__er"
```

12. **C_ns_n_nts**

    Vowels make prose far too... pronounceable. Given a string `x`, replace all the vowels (a, e, i, o, u, or y) with underscores.

```
  f "FLAPJACKS"
"FL_PJ_CKS"
  f "Several normal words"
"S_v_r_l n_rm_l w_rds"
```

13. **Cnsnnts Rdx**

    On second thought, I've deemed vowels too vile for placeholders. Given a string `x`, remove all the vowels entirely.

```
  f "Several normal words"
"Svrl nrml wrds"
  f "FLAPJACKS"
"FLPJCKS"
```

14. **TITLE REDACTED**

    Given a string `x` consisting of words separated by spaces (as above), and a string `y`, replace all words in `x` which are the same as `y` with a series of `X`es:

```
  f["a few words in a sentence";"words"]
"a few XXXXX in a sentence"
  f["one fish two fish";"fish"]
"one XXXX two XXXX"
  f["I don't give a care";"care"]
"I don't give a XXXX"
```

15. **It's More Fun To Permute**

    My ingenious histogram-based filing system has a tiny flaw: some people insist that the order of letters in their names is significant, and now I need to re-file everything. Given a string `x`, generate a list of all possible reorderings of the characters in `x`. Can you do this non-recursively?

```
  f "xyz"
("xyz"
 "xzy"
 "yzx"
 "yxz"
 "zxy"
 "zyx")
```