Hard

Complexity CodeEval.com Challenges - 2013

Completed by:

Kestutis IT Dev - July, 2013.

Data #4, (**/1-hardComplexityChallenges/telephoneWordsByKestutis.php**)

Telephone Words

Challenge Description:

Given a 7 digit telephone number, print out all the possible sequences of letters that can represent the given telephone number. Note that in a standard 12 key pad, 0 and 1 do not have any letters associated with them. They are to be treated as such, i.e. a 0/1 in the telephone number will be retained in the final word as well. You may use the following mapping between numbers and characters:

```
0 => 0
1 => 1
2 => abc
3 => def
4 => ghi
5 => jkl
6 => mno
7 => pqrs
8 => tuv
9 => wxyz
```

Input sample:

Your program should accept as its first argument a path to a filename.The input file contains 7 digit telephone numbers, one per line.

```
4155230
```

Output sample:

Print out the words that can produce the telephone number, alphabetically sorted and comma delimited. eg.

```
g1jjad0,g1jjae0,g1jjaf0,g1jjbd0,g1jjbe0,g1jjbf0,g1jjcd0,g1jjce0,g1jjcf0,g1jkad0,g1jkae
0,g1jkaf0,g1jkbd0,g1jkbe0,g1jkbf0,g1jkcd0,g1jkce0,g1jkcf0,g1jlad0,g1jlae0,g1jlaf0,g1jl
bd0,g1jlbe0,g1jlbf0,g1jlcd0,g1jlce0,g1jlcf0,g1kjad0,g1kjae0,g1kjaf0,g1kjbd0,g1kjbe0,g1
kjbf0,g1kjcd0,g1kjce0,g1kjcf0,g1kkad0,g1kkae0,g1kkaf0,g1kkbd0,g1kkbe0,g1kkbf0,g1kkcd0,
g1kkce0,g1kkcf0,g1klad0,g1klae0,g1klaf0,g1klbd0,g1klbe0,g1klbf0,g1klcd0,g1klce0,g1klcf
0,g1ljad0,g1ljae0,g1ljaf0,g1ljbd0,g1ljbe0,g1ljbf0,g1ljcd0,g1ljce0,g1ljcf0,g1lkad0,g1lk
ae0,g1lkaf0,g1lkbd0,g1lkbe0,g1lkbf0,g1lkcd0,g1lkce0,g1lkcf0,g1llad0,g1llae0,g1llaf0,g1
llbd0,g1llbe0,g1llbf0,g1llcd0,g1llce0,g1llcf0,h1jjad0,h1jjae0,h1jjaf0,h1jjbd0,h1jjbe0,
h1jjbf0,h1jjcd0,h1jjce0,h1jjcf0,h1jkad0,h1jkae0,h1jkaf0,h1jkbd0,h1jkbe0,h1jkbf0,h1jkcd
0,h1jkce0,h1jkcf0,h1jlad0,h1jlae0,h1jlaf0,h1jlbd0,h1jlbe0,h1jlbf0,h1jlcd0,h1jlce0,h1jl
cf0,h1kjad0,h1kjae0,h1kjaf0,h1kjbd0,h1kjbe0,h1kjbf0,h1kjcd0,h1kjce0,h1kjcf0,h1kkad0,h1
kkae0,h1kkaf0,h1kkbd0,h1kkbe0,h1kkbf0,h1kkcd0,h1kkce0,h1kkcf0,h1klad0,h1klae0,h1klaf0,
h1klbd0,h1klbe0,h1klbf0,h1klcd0,h1klce0,h1klcf0,h1ljad0,h1ljae0,h1ljaf0,h1ljbd0,h1ljbe
0,h1ljbf0,h1ljcd0,h1ljce0,h1ljcf0,h1lkad0,h1lkae0,h1lkaf0,h1lkbd0,h1lkbe0,h1lkbf0,h1lk
cd0,h1lkce0,h1lkcf0,h1llad0,h1llae0,h1llaf0,h1llbd0,h1llbe0,h1llbf0,h1llcd0,h1llce0,h1
llcf0,i1jjad0,i1jjae0,i1jjaf0,i1jjbd0,i1jjbe0,i1jjbf0,i1jjcd0,i1jjce0,i1jjcf0,i1jkad0,
i1jkae0,i1jkaf0,i1jkbd0,i1jkbe0,i1jkbf0,i1jkcd0,i1jkce0,i1jkcf0,i1jlad0,i1jlae0,i1jlaf
0,i1jlbd0,i1jlbe0,i1jlbf0,i1jlcd0,i1jlce0,i1jlcf0,i1kjad0,i1kjae0,i1kjaf0,i1kjbd0,i1kj
be0,i1kjbf0,i1kjcd0,i1kjce0,i1kjcf0,i1kkad0,i1kkae0,i1kkaf0,i1kkbd0,i1kkbe0,i1kkbf0,i1
kkcd0,i1kkce0,i1kkcf0,i1klad0,i1klae0,i1klaf0,i1klbd0,i1klbe0,i1klbf0,i1klcd0,i1klce0,
i1klcf0,i1ljad0,i1ljae0,i1ljaf0,i1ljbd0,i1ljbe0,i1ljbf0,i1ljcd0,i1ljce0,i1ljcf0,i1lkad
0,i1lkae0,i1lkaf0,i1lkbd0,i1lkbe0,i1lkbf0,i1lkcd0,i1lkce0,i1lkcf0,i1llad0,i1llae0,i1ll
af0,i1llbd0,i1llbe0,i1llbf0,i1llcd0,i1llce0,i1llcf0
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Moderate

Complexity CodeEval.com
Challenges - 2013

Completed by:

Kestutis IT Dev - July, 2013.

Data #2, (/2-moderateComplexityChallenges/emailValidationByKestutis.php)

Email Validation

Challenge Description:

You are given several strings that may/may not be valid emails. You should write a regular expression that determines if the email id is a valid email id or not. You may assume all characters are from the english language.

Input sample:

Your program should accept as its first argument a filename. This file will contain several text strings, one per line. Ignore all empty lines. E.g.

```
foo@bar.com
this is not an email id
admin#codeeval.com
good123@bad.com
```

Output sample:

Print out 'true' (all lowercase) if the string is a valid email. Else print out 'false' (all lowercase). E.g.

```
true
false
false
true
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Data #5, (**/2-moderateComplexityChallenges/primeNumbersByKestutis.php**)

Prime Numbers

Challenge Description:

Print out the prime numbers less than a given number N. For bonus points your solution should run in N*(log(N)) time or better. You may assume that N is always a positive integer.

Input sample:

Your program should accept as its first argument a path to a filename. Each line in this file is one test case. Each test case will contain an integer n < 4,294,967,295. E.g.

```
10
20
100
```

Output sample:

For each line of input, print out the prime numbers less than N, in ascending order, comma delimited. (There should not be any spaces between the comma and numbers) E.g.

```
2,3,5,7
2,3,5,7,11,13,17,19
2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

This is a Sponsored Challenge

Completed by:

Kestutis IT Dev - July, 2013.

Data #21, (**/2-moderateComplexityChallenges/longestLinesByKestutis.php**)

Longest Lines

Challenge Description:

Write a program to read a multiple line text file and write the 'N' longest lines to stdout. Where the file to be read is specified on the command line.

Input sample:

Your program should read an input file (the first argument to your program). The first line contains the value of the number 'N' followed by multiple lines. You may assume that the input file is formatted correctly and the number on the first line i.e. 'N' is a valid positive integer.e.g.

```
2
Hello World

CodeEval
Quick Fox
A
San Francisco
```

NOTE: For solutions in JavaScript, assume that there are 8 lines of input (i.e. line 1 will be N and the next 7 lines will be the input lines

Output sample:

The 'N' longest lines, newline delimited. Do NOT print out empty lines. Ignore all empty lines in the input. Ensure that there are no trailing empty spaces on each line you print. Also ensure that the lines are printed out in decreasing order of length i.e. the output should be sorted based on their length e.g.

```
San Francisco
Hello World
```

Completed by:

Kestutis IT Dev - July, 2013.

Data #6, (**/2-moderateComplexityChallenges/reverseAndAddPalindromeMakerByKestutis.php**)

Reverse and Add

Challenge Description:

Credits: Programming Challenges by Steven S. Skiena and Miguel A. Revilla

The problem is as follows: choose a number, reverse its digits and add it to the original. If the sum is not a palindrome (which means, it is not the same number from left to right and right to left), repeat this procedure. E.g.

```
195 (initial number) + 591 (reverse of initial number) = 786

786 + 687 = 1473

1473 + 3741 = 5214

5214 + 4125 = 9339 (palindrome)
```

In this particular case the palindrome 9339 appeared after the 4th addition. This method leads to palindromes in a few step for almost all of the integers. But there are interesting exceptions. 196 is the first number for which no palindrome has been found. It is not proven though, that there is no such a palindrome.

Input sample:

Your program should accept as its first argument a path to a filename. Each line in this file is one test case. Each test case will contain an integer n < 10,000. Assume each test case will always have an answer and that it is computable with less than 100 iterations (additions).

Output sample:

For each line of input, generate a line of output which is the number of iterations (additions) to compute the palindrome and the resulting palindrome. (they should be on one line and separated by a single space character). E.g.

```
4 9339
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Easy

Complexity CodeEval.com Challenges - 2013

Completed by:

Kestutis IT Dev - July, 2013.

Data #3, (**/3-easyComplexityChallenges/easy_hexToDecByKestutis.php**)

Hex to Decimal

Challenge Description:

You will be given a hexadecimal (base 16) number. Convert it into decimal (base 10)

Input sample:

Your program should accept as its first argument a path to a filename. Each line in this file contains a hex number. You may assume that the hex number does not have the leading '0x'. Also all alpha characters (e.g. a through f) in the input will be in lowercase e.g.

```
9f
11
```

Output sample:

Print out the equivalent decimal number e.g.

```
159
17
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Data #9, (**/3-easyComplexityChallenges/easy_penultimateWordByKestutis.php**)

Penultimate Word

Challenge Description:

Write a program which finds the next-to-last word in a string.

Input sample:

Your program should accept as its first argument a path to a filename. Input example is the following

```
some line with text

another line
```

Each line has more than one word.

Output sample:

Print the next-to-last word in the following way.

```
with

another
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Data #16, (**/3-easyComplexityChallenges/fibonacciSeriesByKestutis.php**)

Fibonacci Series

Challenge Description:

The Fibonacci series is defined as: F(0) = 0; F(1) = 1; F(n) = F(n-1) + F(n-2) when n>1;. Given a positive integer 'n', print out the F(n).

Input sample:

The first argument will be a text file containing a positive integer, one per line. e.g.

```
5
12
```

Output sample:

Print to stdout, the fibonacci number, F(n).
e.g.

```
5
144
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Data #14, (**/3-easyComplexityChallenges/reverseWordsByKestutis.php**)

Reverse words

Challenge Description:

Write a program to reverse the words of an input sentence.

Input sample:

The first argument will be a text file containing multiple sentences, one per line. Possibly empty lines too. e.g.

```
Hello World
Hello CodeEval
```

Output sample:

Print to stdout, each line with its words reversed, one per line. Empty lines in the input should be ignored. Ensure that there are no trailing empty spaces on each line you print.
e.g.

```
World Hello
CodeEval Hello
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Data #20, (**/3-easyComplexityChallenges/rightmostCharByKestutis.php**)

Rightmost Char

Challenge Description:

You are given a string 'S' and a character 't'. Print out the position of the rightmost occurrence of 't'(case matters) in 'S' or -1 if there is none. The position to be printed out is zero based.

Input sample:

The first argument is a file, containing a string and a character, comma delimited, one per line. Ignore all empty lines in the input file.e.g.

```
Hello World,r
Hello CodeEval,E
```

Output sample:

Print out the zero based position of the character 't' in string 'S', one per line. Do NOT print out empty lines between your output.
e.g.

```
8
10
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Order #20, (**/3-easyComplexityChallenges/rightmostOccurenceByKestutis.php**)

Self-Describing Numbers

Challenge Description:

A number is a self-describing number when (assuming digit positions are labeled 0 to N-1), the digit in each position is equal to the number of times that that digit appears in the number.

Input sample:

The first argument is the pathname to a file which contains test data, one test case per line. Each line contains a positive integer. Each line is in the format: N i.e. a positive integer eg.

```
2020
22
1210
```

Output sample:

If the number is a self-describing number, print out a 1. If not, print out a 0 eg.

```
1
0
1
```

For the curious, here's how 2020 is a self-describing number: Position '0' has value 2 and there is two 0 in the number. Position '1' has value 0 because there are not 1's in the number. Position '2' has value 2 and there is two 2. And the position '3' has value 0 and there are zero 3's.

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Order #17, (**/3-easyComplexityChallenges/uniqueElementsByKestutis.php**)

Unique Elements

Challenge Description:

You are given a sorted list of numbers with duplicates. Print out the sorted list with duplicates removed.

Input sample:

File containing a list of sorted integers, comma delimited, one per line. e.g.

```
1,1,1,2,2,3,3,4,4
2,3,4,5,5
```

Output sample:

Print out the sorted list with duplicates removed, one per line
e.g.

```
1,2,3,4
2,3,4,5
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Order #25, (**/3-easyComplexityChallenges/happyNumberByKestutis.php**)

Happy Numbers

Challenge Description:

A happy number is defined by the following process. Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers.

Input sample:

The first argument is the pathname to a file which contains test data, one test case per line. Each line contains a positive integer. Each line is in the format: N i.e. a positive integer eg.

```
1
7
22
```

Output sample:

If the number is a happy number, print out a 1. If not, print out a 0 eg.

```
1
1
0
```

For the curious, here's why 7 is a happy number: 7->49->97->130->10->1. Here's why 22 is NOT a happy number: 22->8->64->52->29->85->89->145->42->20->4->16->37->58->89 ...

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Order #26, (**/3-easyComplexityChallenges/swapCaseByKestutis.php**)

Swap Case

Challenge Description:

Write a program which swaps letters' case in a sentence. All non-letter characters should remain the same.

Input sample:

Your program should accept as its first argument a path to a filename. Input example is the following

```
Hello world!
JavaScript language 1.8
A letter
```

Output sample:

Print results in the following way.

```
hELLO WORLD!
jAVAsCRIPT LANGUAGE 1.8
a LETTER
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Order #27, (**/3-easyComplexityChallenges/NModMByKestutis.php**)

N Mod M

Challenge Description:

Given two integers N and M, calculate N Mod M (without using any inbuilt modulus operator).

Input sample:

Your program should accept as its first argument a path to a filename. Each line in this file contains two comma separated positive integers. e.g.

```
20,6
2,3
```

You may assume M will never be zero.

Output sample:

Print out the value of N Mod M

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Order #28, (**/3-easyComplexityChallenges/setIntersectionByKestutis.php**)

Set Intersection

Challenge Description:

You are given two sorted list of numbers(ascending order). The lists themselves are comma delimited and the two lists are semicolon delimited. Print out the intersection of these two sets.

Input sample:

File containing two lists of ascending order sorted integers, comma delimited, one per line. e.g.

```
1,2,3,4;4,5,6
7,8,9;8,9,10,11,12
```

Output sample:

Print out the ascending order sorted intersection of the two lists, one per line
e.g.

```
4
8,9
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.

Completed by:

Kestutis IT Dev - July, 2013.

Order #29, (**/3-easyComplexityChallenges/armstrongNumbersByKestutis.php**)

Armstrong Numbers

Challenge Description:

An Armstrong number is an n-digit number that is equal to the sum of the n'th powers of its digits. Determine if the input numbers are Armstrong numbers.

Input sample:

Your program should accept as its first argument a path to a filename. Each line in this file has a positive integer. e.g.

```
6
153
351
```

Output sample:

Print out True/False if the number is an Armstrong number or not e.g.

```
True
True
False
```

Submit your solution in a file **(some file name).(py2| c| cpp| java| rb| pl| php| tcl| clj| js| scala| cs| m| py3| hs| go| bash| lua)** or use the online editor.