﻿ ﻿CS 171

Lab Assignment 9

This lab assignment uses many elements provided in the main bibliographic reference for

these lectures:

Programming in Python 3

A Complete Introduction to the Python Language,

2nd Edition,

Mark Summerfield

﻿**Exercises**

**﻿Exercise 1** *Playing with digits*

*Some numbers have funny properties. For example:*

*89 --> 8¹ + 9² = 89 \* 1*

*695 --> 6² + 9³ + 5⁴= 1390 = 695 \* 2*

*46288 --> 4³ + 6⁴+ 2⁵ + 8⁶ + 8⁷ = 2360688 = 46288 \* 51*

*Given a positive integer n written as abcd... (a, b, c, d... being digits) and a positive integer p*

*we want to find a positive integer k, if it exists, such as the sum of the digits of n taken to the successive powers of p is equal to k \* n.*

*In other words:*

*Is there an integer k such as : (a ^ p + b ^ (p+1) + c ^(p+2) + d ^ (p+3) + ...) = n \* k*

*If it is the case we will return k, if not return -1.*

*Note: n and p will always be given as strictly positive integers.*

Examples:

dig\_pow(89, 1) should return 1 since 8¹ + 9² = 89 = 89 \* 1

dig\_pow(92, 1) should return -1 since there is no k such as 9¹ + 2² equals 92 \* k

dig\_pow(695, 2) should return 2 since 6² + 9³ + 5⁴= 1390 = 695 \* 2

dig\_pow(46288, 3) should return 51 since 4³ + 6⁴+ 2⁵ + 8⁶ + 8⁷ = 2360688 = 46288 \* 51

**Exercise 2** *Count the smiley faces!*

*Given an array (arr) as an argument complete the function countSmileys that should return the total number of smiling faces.*

*Rules for a smiling face:*

*Each smiley face must contain a valid pair of eyes. Eyes can be marked as : or ;*

*A smiley face can have a nose but it does not have to. Valid characters for a nose are - or ~*

*Every smiling face must have a smiling mouth that should be marked with either ) or D*

*No additional characters are allowed except for those mentioned.*

*Valid smiley face examples: :) :D ;-D :~)*

*Invalid smiley faces: ;( :> :} :]*

Examples:

countSmileys([':)', ';(', ';}', ':-D']); // should return 2;

countSmileys([';D', ':-(', ':-)', ';~)']); // should return 3;

countSmileys([';]', ':[', ';\*', ':$', ';-D']); // should return 1;

**Exercise 3** *The Supermarket Queue*

*There is a queue for the self-checkout tills at the supermarket. Your task is write a function to calculate the total time required for all the customers to check out!*

*input*

*customers: an array of positive integers representing the queue. Each integer represents a customer, and its value is the amount of time they require to check out.*

*n: a positive integer, the number of checkout tills.*

*output*

*The function should return an integer, the total time required.*

*Important*

*Please look at the examples and clarifications below, to ensure you understand the task correctly :)*

Examples:

queue\_time([5,3,4], 1)

# should return 12

# because when n=1, the total time is just the sum of the times

queue\_time([10,2,3,3], 2)

# should return 10

# because here n=2 and the 2nd, 3rd, and 4th people in the

# queue finish before the 1st person has finished.

queue\_time([2,3,10], 2)

# should return 12

**Exercise 4** *IP Validation*

*Write an algorithm that will identify valid IPv4 addresses in dot-decimal format. IPs should be considered valid if they consist of four octets, with values between 0 and 255, inclusive.*

*Input to the function is guaranteed to be a single string.*

Examples:

Valid inputs:

1.2.3.4

123.45.67.89

Invalid inputs:

1.2.3

1.2.3.4.5

123.456.78.90

123.045.067.089

Note that leading zeros (e.g. 01.02.03.04) are considered invalid.