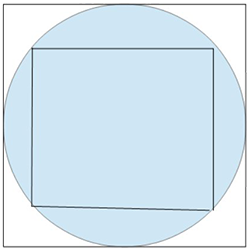
**Problem 1**

**A Circle and Two Squares**

Imagine a circle and two squares: a smaller and a bigger one. For the smaller one, the circle is a circumcircle and for the bigger one, an incircle.



Create a function, that takes an integer (radius of the circle) and returns the difference of the areas of the two squares.

**Examples**

square\_areas\_difference(5) ➞ 50

square\_areas\_difference(6) ➞ 72

square\_areas\_difference(7) ➞ 98

**Notes**

Use only positive integer parameters.

**Problem 2**

## Calculate the Profit

You work for a manufacturer, and have been asked to calculate the total profit made on the sales of a product. You are given a dictionary containing the cost price per unit (in dollars), sell price per unit (in dollars), and the starting inventory. Return the total **profit** made, rounded to the nearest dollar.

### Examples

profit({

"cost\_price": 32.67,

"sell\_price": 45.00,

"inventory": 1200

}) ➞ 14796

profit({

"cost\_price": 225.89,

"sell\_price": 550.00,

"inventory": 100

}) ➞ 32411

profit({

"cost\_price": 2.77,

"sell\_price": 7.95,

"inventory": 8500

}) ➞ 44030

### Notes

* Assume all inventory has been sold.
* Profit = Total Sales - Total Cost

**Problem 3**

**Check If Lines Are Parallel**

Given two lines, determine whether or not they are parallel.

Lines are represented by a list [a, b, c], which corresponds to the line ax+by=c.

**Examples**

lines\_are\_parallel([1, 2, 3], [1, 2, 4]) ➞ True

# x+2y=3 and x+2y=4 are parallel.

lines\_are\_parallel([2, 4, 1], [4, 2, 1]) ➞ False

# 2x+4y=1 and 4x+2y=1 are not parallel.

lines\_are\_parallel([0, 1, 5], [0, 1, 5]) ➞ True

# Lines are parallel to themselves.

**Notes**

* Two lines are parallels if they have the same slope and the y-intercepts are different. If the slopes are different, the lines are not parallel.
* All test cases use valid input (no lists of the wrong size, for example).
* All coefficients will be integers (whole numbers).

**Problem 4**

## Classes For Fetching Information on a Sports Player

Create a class that takes the following four arguments for a particular football player:

* name
* age
* height
* weight

Also, create three functions for the class that returns the following strings:

* get\_age() returns "name is age age"
* get\_height() returns "name is heightcm"
* get\_weight() returns "name weighs weightkg"

### Examples

p1 = Player("David Jones", 25, 175, 75)

p1.get\_age() ➞ "David Jones is 25 years"

p1.get\_height() ➞ "David Jones is 175 cm"

p1.get\_weight() ➞ "David Jones weighs 75 kg"

#### Notes

name will be passed in as a string and age, height, weight will be integers.

**Problem 5**

## Length of Number

Create a function that takes a number num and returns its length.

### Examples

number\_length(10) ➞ 2

number\_length(5000) ➞ 4

number\_length(0) ➞ 1

### Notes

**DO NOT USE LEN() FOR THIS CHALLENGE**

**Problem 6**

**Find the Vertex of a Quadratic**

Every quadratic curve y = a x² + b x + c has a **vertex** point: the turning point where the curve stops heading down and starts going up.

Given the values a, b and c, you need to return the coordinates of the vertex. Return your answers rounded to 2 decimal places.

**Examples**

find\_vertex(1, 0, 25) ➞ [0, 25]

# The vertex of y=x²+25 is at (0, 25).

find\_vertex(-1, 0, 25) ➞ [0, 25]

# The vertex of y=-x²+25 is at (0, 25).

find\_vertex(1, 10, 4) ➞ [-5, -21]

# The vertex of y=x²+10x+4 is at (-5, -21).

**Notes**

* See **Resources** if you're not sure how to find the x or y coordinates of the vertex.
* a will always be non-zero.

**Problem 7**

## Two Distinct Elements

In each input list, every number **repeats at least once**, except for **two**. Write a function that returns the **two unique numbers**.

### Examples

return\_unique([1, 9, 8, 8, 7, 6, 1, 6]) ➞ [9, 7]

return\_unique([5, 5, 2, 4, 4, 4, 9, 9, 9, 1]) ➞ [2, 1]

return\_unique([9, 5, 6, 8, 7, 7, 1, 1, 1, 1, 1, 9, 8]) ➞ [5, 6]

### Notes

Keep the same ordering in the output.

**Problem 8**

**Neutralisation**

Given two strings comprised of + and -, return a new string which shows how the two strings interact in the following way:

* When positives and positives interact, they *remain positive*.
* When negatives and negatives interact, they *remain negative*.
* But when negatives and positives interact, they *become neutral*, and are shown as the number 0.

**Worked Example**

neutralise("+-+", "+--") ➞ "+-0"

# Compare the first characters of each string, then the next in turn.

# "+" against a "+" returns another "+".

# "-" against a "-" returns another "-".

# "+" against a "-" returns "0".

# Return the string of characters.

**Examples**

neutralise("--++--", "++--++") ➞ "000000"

neutralise("-+-+-+", "-+-+-+") ➞ "-+-+-+"

neutralise("-++-", "-+-+") ➞ "-+00"

**Notes**

The two strings will be the same length.

**Problem 9**

## Sum Fractions

Create a function that takes a list containing **nested lists** as an argument. Each sublist has 2 elements. The first element is the numerator and the second element is the denominator. Return the sum of the fractions rounded to the nearest whole number.

### Examples

sum\_fractions([[18, 13], [4, 5]]) ➞ 2

sum\_fractions([[36, 4], [22, 60]]) ➞ 9

sum\_fractions([[11, 2], [3, 4], [5, 4], [21, 11], [12, 6]]) ➞ 11

### Notes

Your result should be a number not string.

**Problem 10**

**Harshad Number**

A number is said to be **Harshad** if it's *exactly divisible* by the **sum** of its digits. Create a function that determines whether a number is a Harshad or not.

**Examples**

is\_harshad(75) ➞ False

# 7 + 5 = 12

# 75 is not exactly divisible by 12

is\_harshad(171) ➞ True

# 1 + 7 + 1 = 9

# 9 exactly divides 171

is\_harshad(481) ➞ True

is\_harshad(89) ➞ False

is\_harshad(516) ➞ True

is\_harshad(200) ➞ True

**Problem 11**

## Harmonic Series

In mathematics, the harmonic series is the divergent infinite series:

Its name derives from the concept of overtones, or harmonics in music.

Create a function that, given a precision parameter, returns the value of the harmonic series.

### Examples

harmonic(3) ➞ 1.833

harmonic(1) ➞ 1.0

harmonic(5) ➞ 2.283

### Notes

Round the result to the third decimal place.

**Problem 12**

## The Fizz Buzz Test

Write a program that returns a list of all the numbers from 1 to an integer argument. But for multiples of three use “Fizz” instead of the number and for the multiples of five use “Buzz”. For numbers which are multiples of both three and five use “FizzBuzz”.

### Example

fizz\_buzz(10) ➞ [1, 2, "Fizz", 4, "Buzz", "Fizz", 7, 8, "Fizz", "Buzz"]

fizz\_buzz(15) ➞ [1, 2, "Fizz", 4, "Buzz", "Fizz", 7, 8, "Fizz", "Buzz", 11, "Fizz", 13, 14, "FizzBuzz"]

### Notes

Make sure to return a list.

**Problem 13**

## Iterated Square Root

The iterated square root of a number is the number of times the square root function must be applied to bring the number **strictly under 2**.

Given an integer, return its iterated square root. Return "invalid" if it is negative.

### Examples

i\_sqrt(1) ➞ 0

i\_sqrt(2) ➞ 1

i\_sqrt(7) ➞ 2

i\_sqrt(27) ➞ 3

i\_sqrt(256) ➞ 4

i\_sqrt(-1) ➞ "invalid"

**Problem 14**

**Wash Your Hands :)**

It takes **21 seconds** to wash your hands and help prevent the spread of COVID-19.

Create a function that takes the number of times a person washes their hands per day N and the number of months they follow this routine nM and calculates the duration in **minutes and seconds** that person spends washing their hands.

**Examples**

wash\_hands(8, 7) ➞ "588 minutes and 0 seconds"

wash\_hands(0, 0) ➞ "0 minutes and 0 seconds"

wash\_hands(7, 9) ➞ "661 minutes and 30 seconds"

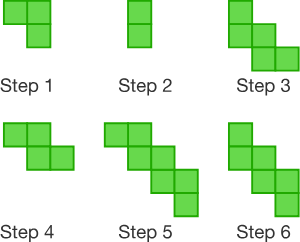
**Notes**

* Consider a month has 30 days.
* Wash your hands.

**Problem 15**

**Algebra Sequence — Boxes**

Create a function that takes a number (step) as an argument and returns the amount of boxes in that step of the sequence.



* Step 0: Start with 0
* Step 1: Add 3
* Step 2: Subtract 1
* Repeat Step 1 & 2 ...

**Examples**

box\_seq(0) ➞ 0

box\_seq(1) ➞ 3

box\_seq(2) ➞ 2

**Notes**

Step (the input) is always a positive integer (or zero).

**Problem 16**

## Last Digit Ultimate

Your job is to create a function, that takes 3 numbers: a, b, c and returns True if the last digit of a \* b = the last digit of c. Check the examples below for an explanation.

### Examples

last\_dig(25, 21, 125) ➞ True

# The last digit of 25 is 5, the last digit of 21 is 1, and the last

# digit of 125 is 5, and the last digit of 5\*1 = 5, which is equal

# to the last digit of 125(5).

last\_dig(55, 226, 5190) ➞ True

# The last digit of 55 is 5, the last digit of 226 is 6, and the last

# digit of 5190 is 0, and the last digit of 5\*6 = 30 is 0, which is

# equal to the last digit of 5190(0).

last\_dig(12, 215, 2142) ➞ False

# The last digit of 12 is 2, the last digit of 215 is 5, and the last

# digit of 2142 is 2, and the last digit of 2\*5 = 10 is 0, which is

# not equal to the last digit of 2142(2).

### Notes

Numbers can be negative.

**Problem 17**

## Expensive Orders

Write a function that has two parameters: orders and cost. Return any orders that are greater than the cost.

### Examples

expensive\_orders({ "a": 3000, "b": 200, "c": 1050 }, 1000)

➞ { "a": 3000, "c": 1050 }

expensive\_orders({ "Gucci Fur": 24600, "Teak Dining Table": 3200, "Louis Vutton Bag": 5550, "Dolce Gabana Heels": 4000 }, 20000)

➞ { "Gucci Fur": 24600 }

expensive\_orders({ "Deluxe Burger": 35, "Icecream Shake": 4, "Fries": 5 }, 40)

➞ {}

**Problem 18**

**Sastry Numbers**

In this challenge, you have to establish if a given integer n is a Sastry number. If the number resulting from the concatenation of an integer n with its successor is a perfect square, then n is a Sastry Number.

Given a positive integer n, implement a function that returns True if n is a Sastry number, or False if it's not.

**Examples**

is\_sastry(183) ➞ True

# Concatenation of n and its successor = 183184

# 183184 is a perfect square (428 ^ 2)

is\_sastry(184) ➞ False

# Concatenation of n and its successor = 184185

# 184185 is not a perfect square

is\_sastry(106755) ➞ True

# Concatenation of n and its successor = 106755106756

# 106755106756 is a perfect square (326734 ^ 2)

**Notes**

* A perfect square is a number with a square root equals to a whole integer.
* You can expect only valid positive integers greater than 0 as input, without exceptions to handle. Zero is a perfect square, but the concatenation 00 isn't considered as a valid result to check.

**Problem 19**

**Squares and Cubes**

Create a function that takes a list of two numbers and checks if the **square root** of the first number is equal to the **cube root** of the second number.

**Examples**

check\_square\_and\_cube([4, 8]) ➞ True

check\_square\_and\_cube([16, 48]) ➞ False

check\_square\_and\_cube([9, 27]) ➞ True

**Notes**

* Remember to return either True or False.
* All lists contain *two positive numbers*.

**Problem 20**

**Free Throw Probability**

What's the probability of someone making a certain amount of free throws in a row given their free throw success percentage? If Sally makes 50% of her free shot throws. Then Sally's probability of making 5 in a row would be 3%.

**Examples**

free\_throws("75%", 5) ➞ "24%"

free\_throws("25%", 3) ➞ "2%"

free\_throws("90%", 30) ➞ "4%"

**Notes**

* The success rate is a string.
* The function should return a string with the percent sign.
* Round your answer to the nearest whole number.

**Problem 21**

**Censor Words Longer Than Four Characters**

Create a function that takes a string and censors words **over four characters** with \*.

**Examples**

censor("The code is fourty") ➞ "The code is \*\*\*\*\*\*"

censor("Two plus three is five") ➞ "Two plus \*\*\*\*\* is five"

censor("aaaa aaaaa 1234 12345") ➞ "aaaa \*\*\*\*\* 1234 \*\*\*\*\*"

**Notes**

* Don't censor words with exactly four characters.
* If all words have four characters or less, return the original string.
* The amount of \* is the same as the length of the word.

**Problem 22**

**Don't Roll Doubles!**

John is playing a dice game. The rules are as follows.

1. Roll two dice.
2. Add the numbers on the dice together.
3. Add the total to your overall score.
4. Repeat this for three rounds.

**But if you roll DOUBLES, your score is instantly wiped to 0 and your game ends immediately!**

Create a function that takes in a list of tuples as input, and return John's score after his game has ended.

**Examples**

dice\_game([(1, 2), (3, 4), (5, 6)]) ➞ 21

dice\_game([(1, 1), (5, 6), (6, 4)]) ➞ 0

dice\_game([(4, 5), (4, 5), (4, 5)]) ➞ 27

**Notes**

* Ignore all other tuples in the list if a throw happens to be doubles and go straight to returning 0.
* John only has two dice and will always give you outcomes for three rounds.

**Problem 23**

## Pandigital Numbers

A **pandigital** number contains all digits (0-9) at least once. Write a function that takes an integer, returning True if the integer is pandigital, and False otherwise.

### Examples

is\_pandigital(98140723568910) ➞ True

is\_pandigital(90864523148909) ➞ False

# 7 is missing.

is\_pandigital(112233445566778899) ➞ False

### Notes

Think about the properties of a pandigital number when all duplicates are removed.

**Problem 24**

**From A to Z**

Given a string indicating a range of letters, return a string which includes all the letters in that range, *including* the last letter. Note that if the range is given in *capital letters*, return the string in capitals also!

**Examples**

gimme\_the\_letters("a-z") ➞ "abcdefghijklmnopqrstuvwxyz"

gimme\_the\_letters("h-o") ➞ "hijklmno"

gimme\_the\_letters("Q-Z") ➞ "QRSTUVWXYZ"

gimme\_the\_letters("J-J") ➞ J"

**Notes**

* A *hyphen* will separate the two letters in the string.
* You don't need to worry about error handling in this one (i.e. both letters will be the same case and the second letter will always be after the first alphabetically).