**Name: Session:**

**Programming I**

**Final Challenge**

For your final challenge, here are 20 programs to work on. Each program has starter code which requires you only complete the function. The functions should return a value which is printed out by the main program. Most of these functions are fairly short. Before your panic, you do not have to complete all of these. You will have until Friday. 1/22/2021 to complete as many as you can. Your grade will be based on how many you complete. If you complete 15 – 20, you will get an A (100). If you complete 10 – 14, you will get a B (85). If you get 5 – 9, you will get a C (70). Partial credit will be awarded for partially completed functions. Each function has associated test code which will test your functions. You should print out your documented source code and attach it to this sheet in numerical order.

1. Create a function that takes in a date and returns the correct century.

Examples

century(1756) ➞ "18th century"

century(1555) ➞ "16th century"

century(1000) ➞ "10th century"

century(1001) ➞ "11th century"

century(2005) ➞ "21st century"

Notes

All dates will be between 1000 and 2010.

The 11th century is between 1001 and 1100.

The 18th century is between 1701-1800.

2. Create a function that takes a number as an argument and returns True or False depending on whether the number is symmetrical or not. A number is symmetrical when it is the same as its reverse.

Examples

is\_symmetrical(7227) ➞ True

is\_symmetrical(12567) ➞ False

is\_symmetrical(44444444) ➞ True

is\_symmetrical(9939) ➞ False

is\_symmetrical(1112111) ➞ True

3. A museum wants to get rid of some exhibitions. Katya, the interior architect, comes up with a plan to remove the most boring exhibitions. She gives them a rating, and removes the one with the lowest rating. Just as she finishes rating the exhibitions, she's called off to an important meeting. She asks you to write a program that tells her the ratings of the items after the lowest one is removed.

Create a function that takes a list of integers and removes the smallest value.

Examples

remove\_smallest([1, 2, 3, 4, 5] ) ➞ [2, 3, 4, 5]

remove\_smallest([5, 3, 2, 1, 4]) ➞ [5, 3, 2, 4]

remove\_smallest([2, 2, 1, 2, 1]) ➞ [2, 2, 2, 1]

Notes

Don't change the order of the left over items.

If you get an empty list, return an empty list: [] ➞ [].

If there are multiple items with the same value, remove item with lower index (3rd example).

4. An isogram is a word that has no repeating letters, consecutive or nonconsecutive. Create a function that takes a string and returns either True or False depending on whether or not it's an "isogram".

Examples

is\_isogram("Algorism") ➞ True

is\_isogram("PasSword") ➞ False # Not case sensitive.

is\_isogram("Consecutive") ➞ False

Notes

Ignore letter case (should not be case sensitive).

All test cases contain valid one word strings.

5. Steve and Maurice have racing snails. They each have 3, a slow (s), medium (m) and fast (f) one. Although Steve's snails are all a bit stronger than Maurice's, Maurice has a trick up his sleeve. His plan is:

Round 1: [s, f] Sacrifice his slowest snail against Steve's fastest.

Round 2: [m, s] Use his middle snail against Steve's slowest.

Round 3: [f, m] Use his fastest snail against Steve's middle.

Create a function that determines whether Maurice's plan will work by outputting true if Maurice wins 2/3 games.

The function inputs:

List 1: [s, m, f] for Maurice.

List 2: [s, m, f] for Steve.

Examples

maurice\_wins([3, 5, 10], [4, 7, 11]) ➞ True

# Since the matches are (3, 11), (5, 4) and (10, 7), Maurice wins 2 out of 3.

maurice\_wins([6, 8, 9], [7, 12, 14]) ➞ False

# Since the matches are (6, 14), (8, 7) and (9, 12), Steve wins 2 out of 3.

maurice\_wins([1, 8, 20], [2, 9, 100]) ➞ True

Notes

Maurice wins if his competing snail's speed strictly exceeds Steve's competing snail's speed.

Steve will always play in this order: [f, s, m].

6. ATM machines allow 4 or 6 digit PIN codes and PIN codes cannot contain anything but exactly 4 digits or exactly 6 digits. Your task is to create a function that takes a string and returns True if the PIN is valid and False if it's not.

**Examples**

is\_valid\_PIN("1234") ➞ True

is\_valid\_PIN("12345") ➞ False

is\_valid\_PIN("a234") ➞ False

is\_valid\_PIN("") ➞ False

**Notes**

Some test cases contain special characters.

Empty strings must return False.

7. Given a list of boxes, create a function that returns the total volume of all those boxes combined together. A box is represented by a list with three elements: length, width and height.

For instance, total\_volume([2, 3, 2], [6, 6, 7], [1, 2, 1]) should return 266 since (2 x 3 x 2) + (6 x 6 x 7) + (1 x 2 x 1) = 12 + 252 + 2 = 266.

Examples

total\_volume([[4, 2, 4], [3, 3, 3], [1, 1, 2], [2, 1, 1]]) ➞ 63

total\_volume([[2, 2, 2], [2, 1, 1]]) ➞ 10

total\_volume([[1, 1, 1]]) ➞ 1

Notes

You will be given at least one box.

Each box will always have three dimensions included.

8. Create a function that takes a list of positive and negative numbers. Return a list where the first element is the count of positive numbers and the second element is the sum of negative numbers.

Examples

sum\_neg([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -11, -12, -13, -14, -15]) ➞ [10, -65]

# There are a total of 10 positive numbers.

# The sum of all negative numbers equals -65.

sum\_neg([92, 6, 73, -77, 81, -90, 99, 8, -85, 34]) ➞ [7, -252]

sum\_neg([91, -4, 80, -73, -28]) ➞ [2, -105]

sum\_neg([]) ➞ []

Notes

If given an empty list, return an empty list: []

9. Create a function that accepts a list of two strings and checks if the letters in the second string are present in the first string.

Examples

letter\_check("trances", "nectar") ➞ True

letter\_check("compadres", "DRAPES") ➞ True

letter\_check("parses", "parsecs") ➞ False

Notes

Function should not be case sensitive (as indicated in the second example).

Both strings are presented as a single argument in the form of a list.

10. Create a function that takes a string as an argument and returns a coded (h4ck3r 5p34k) version of the string.

Examples

hacker\_speak("javascript is cool") ➞ "j4v45cr1pt 15 c00l"

hacker\_speak("programming is fun") ➞ "pr0gr4mm1ng 15 fun"

hacker\_speak("become a coder") ➞ "b3c0m3 4 c0d3r"

Notes

In order to work properly, the function should replace all 'a's with 4, 'e's with 3, 'i's with 1, 'o's with 0, and 's's with 5.

11. Throughout the 12 days of Christmas, my true love gave me in total 364 items.

Create a function where given n days as an argument, return the total amount of items received throughout those days as an integer.

Examples

xmasItems(12) ➞ 364

xmasItems(3) ➞ 10

xmasItems(31) ➞ 5456

Notes

You will only be given valid inputs.

Remember to return as an integer.

0 as input should return 0.

12. Create a function that takes a list of 10 numbers (between 0 and 9) and returns a string of those numbers formatted as a phone number (e.g. (555) 555-5555).

Examples

format\_phone\_number([1, 2, 3, 4, 5, 6, 7, 8, 9, 0]) ➞ "(123) 456-7890"

format\_phone\_number([5, 1, 9, 5, 5, 5, 4, 4, 6, 8]) ➞ "(519) 555-4468"

format\_phone\_number([3, 4, 5, 5, 0, 1, 2, 5, 2, 7]) ➞ "(345) 501-2527"

Notes

Don't forget the space after the closing parenthesis.

13. Create a function that takes any non-negative number as an argument and returns it with its digits in descending order. Descending order is when you sort from highest to lowest.

Examples

sort\_descending(123) ➞ 321

sort\_descending(1254859723) ➞ 9875543221

sort\_descending(73065) ➞ 76530

Notes

You can expect non-negative numbers for all test cases.

14. A number is narcissistic when the sum of its digits, with each digit raised to the power of digits quantity, is equal to the number itself.

153 ➞ 3 digits ➞ 1³ + 5³ + 3³ = 1 + 125 + 27 = 153 ➞ Narcissistic

84 ➞ 2 digits ➞ 8² + 4² = 64 + 16 = 80 ➞ Not narcissistic

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

Examples

is\_narcissistic(8208) ➞ True # 8⁴ + 2⁴ + 0⁴ + 8⁴ = 8208

is\_narcissistic(22) ➞ False #2² + 2² = 8

is\_narcissistic(9) ➞ True # 9¹ = 9

Notes

Trivially, any number in the 1-9 range is narcissistic and any two-digit number is not.

Curious fact: Only 88 numbers are narcissistic.

15. A pair of strings form a strange pair if both of the following are true:

The 1st string's first letter = 2nd string's last letter.

The 1st string's last letter = 2nd string's first letter.

Create a function that returns True if a pair of strings constitutes a strange pair, and False otherwise.

Examples

is\_strange\_pair("ratio", "orator") ➞ True

# "ratio" ends with "o" and "orator" starts with "o".

# "ratio" starts with "r" and "orator" ends with "r".

is\_strange\_pair("sparkling", "groups") ➞ True

is\_strange\_pair("bush", "hubris") ➞ False

is\_strange\_pair("", "") ➞ True

Notes

It should work on a pair of empty strings (they trivially share nothing).

16. Create a function that takes a number a and finds the missing exponent x so that a when raised to the power of x is equal to b.

Examples

solve\_for\_exp(4, 1024) ➞ 5

solve\_for\_exp(2, 1024) ➞ 10

solve\_for\_exp(9, 3486784401) ➞ 10

Notes

a is raised to the power of what in order to equal b?

17. Create a function that takes a number as an argument and returns "Fizz", "Buzz" or "FizzBuzz".

If the number is a multiple of 3 the output should be "Fizz".

If the number given is a multiple of 5, the output should be "Buzz".

If the number given is a multiple of both 3 and 5, the output should be "FizzBuzz".

If the number is not a multiple of either 3 or 5, the number should be output on its own as shown in the examples below.

Examples

fizz\_buzz(3) ➞ "Fizz"

fizz\_buzz(5) ➞ "Buzz"

fizz\_buzz(15) ➞ "FizzBuzz"

fizz\_buzz(4) ➞ "4"

18. Create a function that takes a single word string and does the following:

* Concatenates inator to the end if the word ends with a consonant otherwise,
* concatenate -inator instead.

Adds the word length of the original word to the end, supplied with '000'.

The examples should make this clear.

Examples

inatorInator('Shrink') ➞ 'Shrinkinator 6000'

inatorInator('Doom') ➞ 'Doominator 4000'

inatorInator('EvilClone') ➞ 'EvilClone-inator 9000'

19. Create a function that takes a Present Value of Cash pv, an Investment Rate ir and the Number of Years years to be Invested and returns the Net Present Value.

In the world of finance, the time value of money must be taken into account. In simple terms, this is because $100 now would buy more than $100 a year from now. Therefore, if we receive $100 in one years time, it will not be worth as much to us today.

Assuming we received the Present Value of Cash at the end of each year over a period of time, this can be calculated by multiplying the Present Value of Cash by the cumulative present value interest rate.

The cumulative present value interest rate can be calculated by:



The result should always be rounded to the nearest whole dollar.

It is not possible to receive a negative amount of money, use a negative investment rate or invest for a negative number of years. These should return False.

Examples

net\_present\_value(100, 0.1, 1) ➞ "$91"

net\_present\_value(100, 0.2, 1) ➞ "$83"

net\_present\_value(100, 0.1, 20) ➞ "$851"

20. Write a function that takes a string of one or more words as an argument and returns the same string, but with all five or more letter words reversed. Strings passed in will consist of only letters and spaces. Spaces will be included only when more than one word is present.

Examples

reverse("Reverse") ➞ "esreveR"

reverse("This is a typical sentence.") ➞ "This is a lacipyt .ecnetnes"

reverse("The dog is big.") ➞ "The dog is big."

Notes

You can expect a valid string to be provided for each test case.