Question1

Create a function that takes three integer arguments (a, b, c) and returns the amount of integers which are of equal value.

**Examples**

equal(3, 4, 3) ➞ 2

equal(1, 1, 1) ➞ 3

equal(3, 4, 1) ➞ 0

**Notes**

Your function must return 0, 2 or 3.

ANS: def equal(a, b, c):

if a == b == c:

return 3

elif a == b or b == c or a == c:

return 2

else:

return 0

# Test cases

print(equal(3, 4, 3)) # ➞ 2

print(equal(1, 1, 1)) # ➞ 3

print(equal(3, 4, 1)) # ➞ 0

Question2

Write a function that converts a **dictionary** into a **list** of keys-values **tuples**.

### Examples

dict\_to\_list({

"D": 1,

"B": 2,

"C": 3

}) ➞ [("B", 2), ("C", 3), ("D", 1)]

dict\_to\_list({

"likes": 2,

"dislikes": 3,

"followers": 10

}) ➞ [("dislikes", 3), ("followers", 10), ("likes", 2)]

### Notes

Return the elements in the list in alphabetical order.

ANS: To convert a dictionary into a list of key-value tuples, you can use the `items()` method of the dictionary, which returns a view object containing tuples of the dictionary's key-value pairs. You can then sort this list of tuples in alphabetical order based on the keys. Here's the implementation of the `dict\_to\_list` function:

def dict\_to\_list(input\_dict):

return sorted(input\_dict.items())

# Test cases

print(dict\_to\_list({"D": 1, "B": 2, "C": 3})) # ➞ [("B", 2), ("C", 3), ("D", 1)]

print(dict\_to\_list({"likes": 2, "dislikes": 3, "followers": 10})) # ➞ [("dislikes", 3), ("followers", 10), ("likes", 2)]

The function `dict\_to\_list` takes a dictionary `input\_dict` as input. It uses the `items()` method to retrieve the key-value pairs as tuples and then sorts them using the `sorted()` function. The result is a list of key-value tuples sorted in alphabetical order based on the keys.

Question3

Write a function that creates a dictionary with each **(key, value)** pair being the **(lower case, upper case)** versions of a letter, respectively.

### Examples

mapping(["p", "s"]) ➞ { "p": "P", "s": "S" }

mapping(["a", "b", "c"]) ➞ { "a": "A", "b": "B", "c": "C" }

mapping(["a", "v", "y", "z"]) ➞ { "a": "A", "v": "V", "y": "Y", "z": "Z" }

### Notes

All of the letters in the input list will always be lowercase.

Question4

Write a function, that replaces all vowels in a string with a specified vowel.

### Examples

vow\_replace("apples and bananas", "u") ➞ "upplus und bununus"

vow\_replace("cheese casserole", "o") ➞ "chooso cossorolo"

vow\_replace("stuffed jalapeno poppers", "e") ➞ "steffed jelepene peppers"

### Notes

All words will be lowercase. Y is not considered a vowel.

ANS: def mapping(letters):

return {letter: letter.upper() for letter in letters}

# Test cases

print(mapping(["p", "s"])) # ➞ { "p": "P", "s": "S" }

print(mapping(["a", "b", "c"])) # ➞ { "a": "A", "b": "B", "c": "C" }

print(mapping(["a", "v", "y", "z"])) # ➞ { "a": "A", "v": "V", "y": "Y", "z": "Z" }

Question5

Create a function that takes a string as input and capitalizes a letter if its ASCII code is even and returns its lower case version if its ASCII code is odd.

### Examples

ascii\_capitalize("to be or not to be!") ➞ "To Be oR NoT To Be!"

ascii\_capitalize("THE LITTLE MERMAID") ➞ "THe LiTTLe meRmaiD"

ascii\_capitalize("Oh what a beautiful morning.") ➞ "oH wHaT a BeauTiFuL moRNiNg."

ANS: To implement the `ascii\_capitalize` function, you can iterate through each character in the input string, check the ASCII code of the character, and capitalize it if the code is even, or convert it to lowercase if the code is odd. Here's the implementation:

def ascii\_capitalize(input\_str):

result = ""

for char in input\_str:

ascii\_code = ord(char)

if ascii\_code % 2 == 0:

result += char.upper()

else:

result += char.lower()

return result

# Test cases

print(ascii\_capitalize("to be or not to be!")) # ➞ "To Be oR NoT To Be!"

print(ascii\_capitalize("THE LITTLE MERMAID")) # ➞ "THe LiTTLe meRmaiD"

print(ascii\_capitalize("Oh what a beautiful morning.")) # ➞ "oH wHaT a BeauTiFuL moRNiNg."

The function `ascii\_capitalize` takes a string `input\_str` as input. It initializes an empty string `result` to store the modified string. It iterates through each character in the input string, calculates its ASCII code using `ord(char)`, and then checks if the code is even or odd using the modulo operator `%`. If the code is even, it capitalizes the character using `char.upper()` and appends it to the `result`. If the code is odd, it converts the character to lowercase using `char.lower()` and appends it to the `result`. The function returns the modified string as the output.