Coders here is a simple task for you, Given string str. Your task is to check whether it is a binary string or not by using python set.

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

**For example:**

| **Input** | **Result** |
| --- | --- |
| 01010101010 | Yes |
| 010101 10101 | No |

def is\_binary\_string(s):

# Create a set of unique characters in the string

unique\_chars = set(s)

# If the set only contains '0' and '1', and the length is 2, it's a binary string

if unique\_chars == {'0', '1'} or unique\_chars == {'0'} or unique\_chars == {'1'}:

return "Yes"

else:

return "No"

str1 = input()

print(is\_binary\_string(str1))

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 01010101010 | Yes | Yes |  |
|  | REC123 | No | No |  |
|  | 010101 10101 | No | No |  |

Write a program to eliminate the common elements in the given 2 arrays and print only the non-repeating

elements and the total number of such non-repeating elements.

Input Format:

The first line contains space-separated values, denoting the size of the two arrays in integer format respectively.

The next two lines contain the space-separated integer arrays to be compared.

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Input:

5 4

1 2 8 6 5

2 6 8 10

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output:

1 5 10

3

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127)  Input:

5 5

1 2 3 4 5

1 2 3 4 5

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output:

NO SUCH ELEMENTS

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5 4  1 2 8 6 5  2 6 8 10 | 1 5 10  3 |

def find\_non\_repeating\_elements(arr1, arr2):

set1 = set(arr1)

set2 = set(arr2)

non\_repeating\_elements = (set1.symmetric\_difference(set2))

if non\_repeating\_elements:

print(" ".join(map(str, non\_repeating\_elements)))

print(len(non\_repeating\_elements))

else:

print("NO SUCH ELEMENTS")

# Reading input

n, m = map(int, input().split())

arr1 = list(map(int, input().split()))

arr2 = list(map(int, input().split()))

# Function call

find\_non\_repeating\_elements(arr1, arr2)

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5 4  1 2 8 6 5  2 6 8 10 | 1 5 10  3 | 1 5 10  3 |  |
|  | 3 3  10 10 10  10 11 12 | 11 12  2 | 11 12  2 |  |

Given a tuple and a positive integer k, the task is to find the count of distinct pairs in the tuple whose sum is equal to **K**.

**Examples:**

**Input:** t = (5, 6, 5, 7, 7, 8 ), K = 13   
**Output:** 2   
**Explanation:**   
Pairs with sum K( = 13) are  {(5, 8), (6, 7), (6, 7)}.   
Therefore, distinct pairs with sum K( = 13) are { (5, 8), (6, 7) }.   
Therefore, the required output is 2.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1,2,1,2,5  3 | 1 |
| 1,2  0 | 0 |

def countDistinctPairs(t, K):

pair\_set = set()

count = 0

for i in range(len(t)):

for j in range(i + 1, len(t)):

if t[i] + t[j] == K and (t[i], t[j]) not in pair\_set and (t[j], t[i]) not in pair\_set:

count += 1

pair\_set.add((t[i], t[j]))

return count

# Prompt the user to input the tuple as comma-separated values

t\_input = input()

# Prompt the user to input the value of K

K\_input = int(input())

# Convert the tuple elements to integers

t = tuple(map(int, t\_input.split(',')))

# Calculate the count of distinct pairs

result = countDistinctPairs(t, K\_input)

print(result)

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5,6,5,7,7,8  13 | 2 | 2 |  |
|  | 1,2,1,2,5  3 | 1 | 1 |  |
|  | 1,2  0 | 0 | 0 |  |

The **DNA sequence** is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.

* For example, "ACGAATTCCG" is a **DNA sequence**.

When studying **DNA**, it is useful to identify repeated sequences within the DNA.

Given a string s that represents a **DNA sequence**, return all the **10-letter-long** sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in **any order**.

**Example 1:**

**Input:** s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

**Output:** ["AAAAACCCCC","CCCCCAAAAA"]

**Example 2:**

**Input:** s = "AAAAAAAAAAAAA"

**Output:** ["AAAAAAAAAA"]

**For example:**

| **Input** | **Result** |
| --- | --- |
| AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT | AAAAACCCCC  CCCCCAAAAA |

def findRepeatedDnaSequences(s):

if len(s) <= 10:

return []

sequences\_count = {}

result = set()

for i in range(len(s) - 9):

sequence = s[i:i+10]

sequences\_count[sequence] = sequences\_count.get(sequence, 0) + 1

if sequences\_count[sequence] > 1:

result.add(sequence)

return sorted(list(result)) # Sort and convert the set to list

# Example usage:

s1 = input()

result1 = findRepeatedDnaSequences(s1)

for sequence in result1:

print(sequence)

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT | AAAAACCCCC  CCCCCAAAAA | AAAAACCCCC  CCCCCAAAAA |  |
|  | AAAAAAAAAAAAA | AAAAAAAAAA | AAAAAAAAAA |  |

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.There is only **one repeated number** in nums, return *this repeated number*. Solve the problem using set.

**Example 1:**

**Input:** nums = [1,3,4,2,2]

**Output:** 2

**Example 2:**

**Input:** nums = [3,1,3,4,2]

**Output:** 3

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1 3 4 4 2 | 4 |

def findDuplicate(nums):

num\_set = set()

for num in nums:

if num in num\_set:

return num

num\_set.add(num)

# Get input from the user

nums = list(map(int, input().split()))

# Call the function and print the result

result = findDuplicate(nums)

print(result)

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1 3 4 4 2 | 4 | 4 |  |
|  | 1 2 2 3 4 5 6 7 | 2 | 2 |  |

Passed all tests!

Week 8

Given an array of names of candidates in an election. A candidate name in the array represents a vote cast to the candidate. Print the name of candidates received Max vote. If there is tie, print a lexicographically smaller name.

**Examples:**

Input :  votes[] = {"john", "johnny", "jackie",

                    "johnny", "john", "jackie",

                    "jamie", "jamie", "john",

                    "johnny", "jamie", "johnny",

                    "john"};

Output : John

We have four Candidates with name as 'John', 'Johnny', 'jamie', 'jackie'. The candidates John and Johny get maximum votes. Since John is alphabetically smaller, we print it. Use dictionary to solve the above problem

**Sample Input:**

10

John

John

Johny

Jamie

Jamie

Johny

Jack

Johny

Johny

Jackie

**Sample Output:**

Johny

def find\_max\_vote\_candidate(votes):

vote\_count = {}

for candidate in votes:

vote\_count[candidate] = vote\_count.get(candidate, 0) + 1

max\_votes = max(vote\_count.values())

max\_vote\_candidates = [candidate for candidate, count in vote\_count.items() if count == max\_votes]

max\_vote\_candidates.sort()

return max\_vote\_candidates[0]

def main():

n = int(input())

votes = []

for \_ in range(n):

votes.append(input())

print(find\_max\_vote\_candidate(votes))

if \_\_name\_\_ == "\_\_main\_\_":

main()

In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points. The points associated with each letter are shown below:

Points Letters

1 A, E, I, L, N, O, R, S, T and U

2 D and G

3 B, C, M and P

4 F, H, V, W and Y

5 K

8 J and X

10 Q and Z

Write a program that computes and displays the Scrabble™ score for a word. Create a dictionary that maps from letters to point values. Then use the dictionary to compute the score.

A Scrabble™ board includes some squares that multiply the value of a letter or the value of an entire word. We will ignore these squares in this exercise.

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Input

REC

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output

REC is worth 5 points.

**For example:**

| **Input** | **Result** |
| --- | --- |
| REC | REC is worth 5 points. |

letter\_values = {

'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T': 1, 'U': 1,

'D': 2, 'G': 2,

'B': 3, 'C': 3, 'M': 3, 'P': 3,

'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,

'K': 5,

'J': 8, 'X': 8,

'Q': 10, 'Z': 10

}

word = input().upper()

score = sum(letter\_values.get(letter, 0) for letter in word)

print(f"{word} is worth {score} points.")

A sentence is a string of single-space separated words where each word consists only of lowercase letters.A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return a list of all the uncommon words. You may return the answer in any order.

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

 Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

Note:

Use dictionary to solve the problem

**For example:**

| **Input** | **Result** |
| --- | --- |
| this apple is sweet  this apple is sour | sweet sour |

def uncommon\_words(s1, s2):

def count\_words(sentence):

word\_count = {}

for word in sentence.split():

word\_count[word] = word\_count.get(word, 0) + 1

return word\_count

s1\_words = count\_words(s1)

s2\_words = count\_words(s2)

uncommon = []

for word, count in s1\_words.items():

if count == 1 and word not in s2\_words:

uncommon.append(word)

for word, count in s2\_words.items():

if count == 1 and word not in s1\_words:

uncommon.append(word)

return uncommon

s1 = input()

s2 = input()

result = uncommon\_words(s1, s2)

print(" ".join(result))

Give a dictionary with value lists, sort the keys by summation of values in value list.

**Input** : test\_dict = {‘Gfg’ : [6, 7, 4], ‘best’ : [7, 6, 5]}

**Output** : {‘Gfg’: 17, ‘best’: 18}

**Explanation** : Sorted by sum, and replaced.

**Input** : test\_dict = {‘Gfg’ : [8,8], ‘best’ : [5,5]}

**Output** : {‘best’: 10, ‘Gfg’: 16}

**Explanation** : Sorted by sum, and replaced.

 Sample Input:

2

Gfg 6 7 4

Best 7 6 5

Sample Output

Gfg 17

Best 18

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2  Gfg 6 7 4  Best 7 6 5 | Gfg 17  Best 18 |

def sort\_dict\_by\_sum(test\_dict):

sum\_dict = {key: sum(value) for key, value in test\_dict.items()}

sorted\_keys = sorted(test\_dict, key=lambda x: sum\_dict[x])

sorted\_dict = {key: sum(test\_dict[key]) for key in sorted\_keys}

return sorted\_dict

def main():

n = int(input())

test\_dict = {}

for \_ in range(n):

data = input().split()

key = data[0]

values = list(map(int, data[1:]))

test\_dict[key] = values

sorted\_dict = sort\_dict\_by\_sum(test\_dict)

for key, value in sorted\_dict.items():

print(key, value)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Create a student dictionary  for n students with the student name as key and their test mark assignment mark and lab mark as values. Do the following computations and display the result.

1.Identify the student with the  highest average score

2.Identify the student who as the highest Assignment marks

3.Identify the student with the Lowest lab marks

4.Identify the student with the lowest average score

Note:

If more than one student has the same score display all the student names

Sample input:

4

James 67 89 56

Lalith 89 45 45

Ram 89 89 89

Sita 70 70 70

Sample Output:

Ram

James Ram

Lalith

Lalith

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  James 67 89 56  Lalith 89 45 45  Ram 89 89 89  Sita 70 70 70 | Ram  James Ram  Lalith  Lalith |

def highest\_average\_score(students):

averages = {name: sum(marks) / len(marks) for name, marks in students.items()}

max\_average = max(averages.values())

return sorted([name for name, avg in averages.items() if avg == max\_average])

def highest\_assignment\_marks(students):

max\_assignment\_marks = max(student[1] for student in students.values())

return sorted([name for name, marks in students.items() if marks[1] == max\_assignment\_marks])

def lowest\_lab\_marks(students):

min\_lab\_marks = min(student[2] for student in students.values())

return sorted([name for name, marks in students.items() if marks[2] == min\_lab\_marks])

def lowest\_average\_score(students):

averages = {name: sum(marks) / len(marks) for name, marks in students.items()}

min\_average = min(averages.values())

return sorted([name for name, avg in averages.items() if avg == min\_average])

n = int(input())

students = {}

for \_ in range(n):

name, test, assignment, lab = input().split()

students[name] = [int(test), int(assignment), int(lab)]

highest\_avg = highest\_average\_score(students)

highest\_assignment = highest\_assignment\_marks(students)

lowest\_lab = lowest\_lab\_marks(students)

lowest\_avg = lowest\_average\_score(students)

print(\*highest\_avg)

print(\*highest\_assignment)

print(\*lowest\_lab)

print(\*lowest\_avg)

week 9

Write a code to check whether product of digits at even places is divisible by sum of digits

at odd place of a positive integer.

Input Format:

Take an input integer from stdin.

Output Format:

Print TRUE or FALSE.

Example Input:

1256

Output:

TRUE

Example Input:

1595

Output:

FALSE

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(productDigits(1256)) | True |
| print(productDigits(1595)) | False |

def productDigits(n):

digits = str(n)

product\_even = 1

sum\_odd = 0

has\_even = False

has\_odd = False

for i in range(len(digits)):

digit = int(digits[i])

if (i + 1) % 2 == 0:

product\_even \*= digit

has\_even = True

else:

sum\_odd += digit

has\_odd = True

if not has\_odd or sum\_odd == 0:

return False

if not has\_even:

return False

return product\_even % sum\_odd == 0

Given a number with maximum of 100 digits as input, find the difference between the sum

of odd and even position digits.

Input Format:

Take a number in the form of String from stdin.

Output Format:

Print the difference between sum of even and odd digits

Example input:

1453

Output:

1

Explanation:

Here, sum of even digits is 4 + 3 = 7

sum of odd digits is 1 + 5 = 6.

Difference is 1.

Note that we are always taking absolute difference

def differenceSum(n):

sum\_odd = 0

sum\_even = 0

n = str(n)

for i, digit in enumerate(n):

digit = int(digit)

if (i + 1) % 2 == 0:

sum\_even += digit

else:

sum\_odd += digit

difference = abs(sum\_even - sum\_odd)

return difference

An abundant number is a number for which the sum of its proper divisors is greater than

the number itself. Proper divisors of the number are those that are strictly lesser than the number.

Input Format:

Take input an integer from stdin

Output Format:

Return Yes if given number is Abundant. Otherwise, print No

Example input:

12

Output:

Yes

Explanation

The proper divisors of 12 are: 1, 2, 3, 4, 6, whose sum is 1 + 2 + 3 + 4 + 6 = 16. Since sum of

proper divisors is greater than the given number, 12 is an abundant number.

Example input:

13

Output:

No

Explanation

The proper divisors of 13 is: 1, whose sum is 1. Since sum of proper divisors is not greater

than the given number, 13 is not an abundant number.

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(abundant(12)) | Yes |
| print(abundant(13)) | No |

def abundant(n):

divisors = [i for i in range(1, n) if n % i == 0]

if sum(divisors) > n:

return "Yes"

else:

return "No"

An automorphic number is a number whose square ends with the number itself.

For example, 5 is an automorphic number because 5\*5 =25. The last digit is 5 which same

as the given number.

If the number is not valid, it should display “Invalid input”.

If it is an automorphic number display “Automorphic” else display “Not Automorphic”.

Input Format:

Take a Integer from Stdin Output Format: Print Automorphic if given number is Automorphic number,otherwise Not Automorphic Example input: 5 Output: Automorphic Example input: 25 Output: Automorphic Example input: 7 Output: Not Automorphic

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(automorphic(5)) | Automorphic |

def automorphic(n):

# Check if the input number is valid

if n <= 0:

return "Invalid input"

# Calculate the square of the number

square = n \*\* 2

# Check if the square ends with the number itself

if str(square).endswith(str(n)):

return "Automorphic"

else:

return "Not Automorphic"

n = int(input())

print(automorphic(n))

An e-commerce company plans to give their customers a special discount for Christmas.

They are planning to offer a flat discount. The discount value is calculated as the sum of all

the prime digits in the total bill amount.

Write an algorithm to find the discount value for the given total bill amount.

Constraints

1 <= orderValue< 10e100000

Input

The input consists of an integer orderValue, representing the total bill amount.

Output

Print an integer representing the discount value for the given total bill amount.

Example Input

578

Output

12

**For example:**

| **Test** | **Result** |
| --- | --- |
| print(christmasDiscount(578)) | 12 |

def is\_prime(n):

"""Function to check if a number is prime."""

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

i = 5

while i \* i <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

def christmasDiscount(n):

"""Function to calculate the discount value for the given total bill amount."""

total\_discount = 0

for digit in str(n):

if int(digit) > 1 and is\_prime(int(digit)):

total\_discount += int(digit)

return total\_discount