**12 - Modules**

**Ex. No. : 12.1 Date:**

**Register No.: 231801024 Name: Boopesh.R**

**MODULE-POWER OF FOUR**

Given an integer n, print *true* if it is a power of four. Otherwise, print *false*. An integer n is a power of four, if there exists an integer x such that n == 4x.

**PROGRAM:**

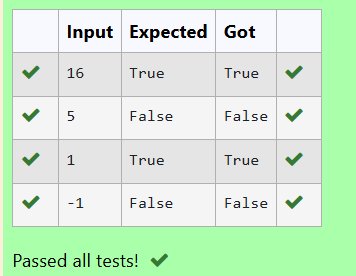
import math

n = int(input())

is\_power\_of\_four = n > 0 and math.log(n, 4).is\_integer()

print(is\_power\_of\_four)

**OUTPUT:**



**Ex. No. : 12.2 Date:**

**Register No.: 231801024 Name: Boopesh.R**

**MODULES- REPRESENTING UNIQUE PAIRS**

Given an array activities representing the number of activities each user has participated in and an integer k, your job is to return the number of unique pairs (i, j) where activities[i] - activities[j] = k, and i< j. The absolute difference between the activities should be exactly k.

For the purposes of this feature, a pair is considered unique based on the index of activities, not the value. That is, if there are two users with the same number of activities, they are considered distinct entities.

Input Format

The first line contains an integer, n, the size of the array nums.

The second line contains n space-separated integers, nums[i].

The third line contains an integer, k.

Output Format

Return a single integer representing the number of unique pairs (i, j)

where | nums[i] - nums[j] | = k and i< j.

**PROGRAM:**

def count\_unique\_pairs(n,activities,k):

unique\_pairs=set()

for i in range(n):

for j in range(i+1,n):

if abs(activities[i]-activities[j])==k:

unique\_pairs.add((i,j))

return len(unique\_pairs)

def main():

n=int(input())

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

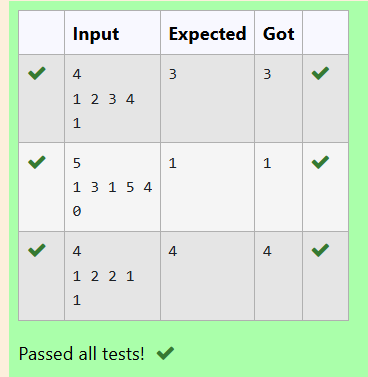
k=int(input())

result=count\_unique\_pairs(n,activities,k)

print(result)

main()

**OUTPUT:**



**Ex. No. : 12.3 Date:**

**Register No.: 231801024 Name: Boopesh.R**

**MODULES-CALCULATING AVERAGE**

Create a Python-based solution that can parse input data representing a list of students with their respective marks and other details, and compute the average marks. The input may present these details in any order, so the solution must be adaptable to this variability.

**PROGRAM:**

def calculate\_average\_marks(N, columns, student\_data):

total\_marks = 0

num\_students = 0

marks\_index = columns.index('MARKS')

for student in student\_data:

marks = int(student[marks\_index])

total\_marks += marks

num\_students += 1

average\_marks = total\_marks / num\_students if num\_students > 0 else 0

return average\_marks

N = int(input())

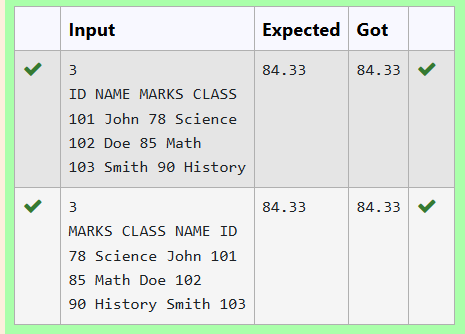
columns = input().split()

student\_data = [input().split() for \_ in range(N)]

average\_marks = calculate\_average\_marks(N, columns, student\_data)

print("{:.2f}".format(average\_marks))

**OUTPUT:**



**Ex. No. : 12.4 Date:**

**Register No.: 231801024 Name: Boopesh.R**

**MODULES-USING DICTIONARY**

Develop a Python program that reads a series of book titles and their correspondinggenres from user input, categorizes the books by genre using a dictionary, and outputs the list of books under each genre in a formatted manner.

**PROGRAM:**

d = {}

while True:

try:

book = input().split(',')

if len(book) < 2:

continue

book\_name = book[0].strip()

category = book[1].strip()

if category in d:

d[category].append(book\_name)

else:

d[category] = [book\_name]

except EOFError:

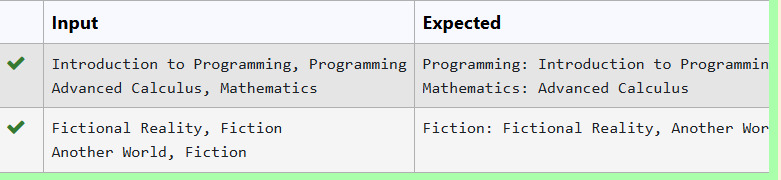
break

for k, v in d.items():

print(f"{k}: ", end='')

print(', '.join(v))

**OUTPUT:**

****

**Ex. No. : 12.5 Date:**

**Register No.: 231801024 Name: Boopesh.R**

**MODULES-DETERMINING THE TOTAL REVENUE**

Develop a Python program that manages shoe inventory and processes sales transactions to determine the total revenue generated. The program should handle inputs of shoe sizes available in the shop, track the number of each size, and match these with customer purchase requests. Each transaction should only proceed if the desired shoe size is in stock, and the inventory should update accordingly after each sale.

**PROGRAM:**

X = int(input())

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

inventory = {}

for shoe in shoes:

if shoe in inventory:

inventory[shoe] += 1

else:

inventory[shoe] = 1

N = int(input())

total\_revenue = 0

for \_ in range(N):

size, price = map(int, input().split())

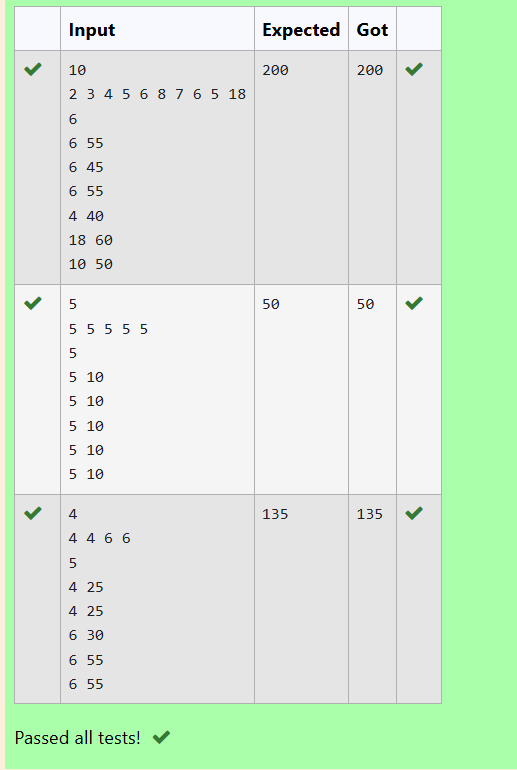
if size in inventory and inventory[size] > 0:

inventory[size] -= 1

total\_revenue += price

print(total\_revenue)

**OUTPUT:**

****