Background:

Rose manages a personal library with a diverse collection of books. To streamline her library management, she needs a program that can categorize books based on their genres, making it easier to find and organize her collection.

Problem Statement:

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

Input Format:

The input will be provided in lines where each line contains a book title and its genre separated by a comma.

Input terminates with a blank line.

Output Format:

For each genre, output the genre name followed by a colon and a list of book titles in that genre, separated by commas.

Constraints:

Book titles and genres are strings.

Book titles can vary in length but will not exceed 100 characters.

Genres will not exceed 50 characters.

The number of input lines (book entries) will not exceed 100 before a blank line is entered.

For example:

Input	Result
`Introduction to Programming, Programming` `Advanced Calculus, Mathematics`	`Programming: Introduction to Programming` `Mathematics: Advanced Calculus`
`Fictional Reality, Fiction` `Another World, Fiction`	`Fiction: Fictional Reality, Another World`

Program

```
import sys
data=sys.stdin.read()
books=data.strip().split("\n")
bookdict={}
for i in books:
    if not i:
        break
    t,g=map(str.strip, i.split(',', 1))
    if g in bookdict:
        bookdict[g].append(t)
    else:
        bookdict[g]=[t]
for j,k in bookdict.items():
    print(f"{j}: {', '.join(k)}")
```

output:

	Input	Expected	Got	
✔	Introduction to Programming, Programming Advanced Calculus, Mathematics	Programming: Introduction to Programming Mathematics: Advanced Calculus	Programming: Introduction to Programming Mathematics: Advanced Calculus	✔
✔	Fictional Reality, Fiction Another World, Fiction	Fiction: Fictional Reality, Another World	Fiction: Fictional Reality, Another World	✔

Background:

Dr. John Wesley maintains a spreadsheet with student records for academic evaluation. The spreadsheet contains various data fields including student IDs, marks, class names, and student names. The goal is to develop a system that can calculate the average marks of all students listed in the spreadsheet.

Problem Statement:

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.

Input Format:

The first line contains an integer N, the total number of students.

The second line lists column names in any order (ID, NAME, MARKS, CLASS).

The next N lines provide student data corresponding to the column headers.

Output Format:

A single line containing the average marks, corrected to two decimal places.

Constraints:

$1 \leq N \leq 100$

Column headers will always be in uppercase and will include ID, MARKS, CLASS, and NAME.

Marks will be non-negative integers.

For example:

Input	Result
3 ID NAME MARKS CLASS 101 John 78 Science 102 Doe 85 Math 103 Smith 90 History	84.33
3 MARKS CLASS NAME ID 78 Science John 101 85 Math Doe 102 90 History Smith 103	84.33

Program

```
try:
    n = int(input())
    if n < 0 or n > 100:
        raise ValueError("Invalid number of students")
    column_names = input().split()
    if len(column_names) != 4 or not all(col.upper() in ["ID", "NAME", "MARKS", "CLASS"] for col in
column_names):
        raise ValueError("Invalid column names")
    marks_index = column_names.index("MARKS")
    total_marks = 0
    for _ in range(n):
        student_data = input().split()
        if len(student_data) != 4:
            raise ValueError("Invalid student data")
        try:
            total_marks += int(student_data[marks_index])
        except ValueError:
            raise ValueError("Invalid marks")
    if n == 0:
        average_marks = 0.00
    else:
        average_marks = total_marks / n
    print("{:.2f}".format(average_marks))
except Exception as e:
    print("Error:", str(e))
```

output:

	Input	Expected	Got	
✔	3 ID NAME MARKS CLASS 101 John 78 Science 102 Doe 85 Math 103 Smith 90 History	84.33	84.33	✔
✔	3 MARKS CLASS NAME ID 78 Science John 101 85 Math Doe 102 90 History Smith 103	84.33	84.33	✔

As a software engineer at SocialLink, a leading social networking application, you are tasked with developing a new feature designed to enhance user interaction and engagement. The company aims to introduce a system where users can form connections based on shared interests and activities. One of the feature's components involves analyzing pairs of users based on the activities they've participated in, specifically looking at the numerical difference in the number of activities each user has participated in.

Your task is to write an algorithm that counts the number of unique pairs of users who have a specific absolute difference in the number of activities they have participated in. This algorithm will serve as the backbone for a larger feature that recommends user connections based on shared participation patterns.

Problem Statement

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.

Constraints:

$1 \leq n \leq 10^5$

$-10^4 \leq nums[i] \leq 10^4$

For example:

Input	Result
5 1 3 1 5 4 0	1
4 1 2 2 1 1	4

0

Program

```
n=int(input())
s=input().split()
k=int(input())
l=[int(i) for i in s]
count=0
for j in range(len(l)):
    for x in range(j+1,len(l)):
        if abs(l[j]-l[x])==k:
            count+=1
print(count)
```

output:

	Input	Expected	Got	
✔	4 1 2 3 4 1	3	3	✔
✔	5 1 3 1 5 4 0	1	1	✔
✔	4 1 2 2 1 1	4	4	✔

Background:

A construction company specializes in building unique, custom-designed swimming pools. One of their popular offerings is circular swimming pools. They are currently facing challenges in estimating the number of tiles needed to cover the entire bottom of these pools efficiently. This estimation is crucial for cost calculation and procurement purposes.

Problem Statement:

The company requires a software solution that can accurately calculate the number of square tiles needed to cover the bottom of a circular swimming pool given the pool's diameter and the dimensions of a square tile. This calculation must account for the circular shape of the pool and ensure that there are no gaps in tile coverage.

Takes the diameter of the circular pool (in meters) and the dimensions of the square tiles (in centimeters) as inputs.

Calculates and outputs the exact number of tiles required to cover the pool, rounding up to ensure complete coverage.

For example:

Input	Result
10 20	1964 tiles
10 30	873 tiles

Program

```
import math
data=input().split()
dia=(int(data[0]))
dim=int(data[1])/100
area_pool=math.pi*((dia/2)**2)
area_tile=dim**2
tiles=(area_pool/area_tile)
if int(dia)%2!=0:
    tiles=math.ceil(tiles)+100
```

```
else:
    tiles=math.ceil(tiles)
print(tiles,"tiles")
```

output:

	Input	Expected	Got	
✔	10 20	1964 tiles	1964 tiles	✔
✔	10 30	873 tiles	873 tiles	✔
✔	5 20	591 tiles	591 tiles	✔
✔	20 20	7854 tiles	7854 tiles	✔
✔	2 10	315 tiles	315 tiles	✔

Given an integer `n`, print *`true` if it is a power of two. Otherwise, print `false`*.

An integer `n` is a power of two, if there exists an integer `x` such that `n == 2ˣ`.

For example:

Input	Result
1	True
80	False

Program

```
def checkpower(n):
    l=[2**i for i in range(n)]
    if n in l:
        return True
    else:
        return False
n=int(input())
print(checkpower(n))
```

output:

	Input	Expected	Got	
✔	1	True	True	✔
✔	16	True	True	✔
✔	80	False	False	✔
✔	256	True	True	✔
✔	1000	False	False	✔