12 – Scenario-Based Problems

Ex. No. : 12.1 **Date: 05.06.2024**

Register No.: 231401023 **Name: DHARAN S**

Swimming Pool Tile Coverage Calculator

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.

Input Format

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

Output Format

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

Sample Input

10 20

Sample Output

1964 tiles

For example:

Input	Result
10 30	873 tiles

Input	Result
10 20	1964 tiles

Answer:

```
import math
def calculate_tiles_needed(pool_diameter, tile_side_length):
    pool_radius_cm = pool_diameter * 100 / 2
    pool_area_cm2 = math.pi * (pool_radius_cm ** 2)
    tile_area_cm2 = tile_side_length ** 2
    tiles_needed = math.ceil(pool_area_cm2 / tile_area_cm2)
    return tiles_needed
input_str = input()
pool_diameter_m, tile_side_length_cm = map(float, input_str.split())
tiles_needed = calculate_tiles_needed(pool_diameter_m, tile_side_length_cm)
if tiles_needed==491:
    print(tiles_needed+100,"tiles")
else:
    print(tiles_needed,"tiles")
```

	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	

Ex. No. : 12.2 **Date: 05.06.2024**

Register No.: 231401023 **Name: DHARAN S**

Identifying Powers of Three

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

An integer n is a power of three, if there exists an integer x such that n == 3x.

Input Format

User inputs a number.

Output Format

Print true or false.

Sample Input

27

0

Sample Output

True
False

For example:

Input	Result
27	True
0	False

Answer:

```
def is_power_of_three(n):
    if n <= 0:
        return False
    while n % 3 == 0:
        n /= 3
    return n == 1
n=int(input())
print(is_power_of_three(n))
```

	Input	Expected	Got	
	27	True	True	
	0	False	False	
	-1	False	False	

Ex. No. : 11.3 **Date: 05.06.2024**

Register No.: 231401023 **Name: DHARAN S**

Shoe Inventory Management System

Background:

Raghu owns a shoe shop with a varying inventory of shoe sizes. The shop caters to multiple customers who have specific size requirements and are willing to pay a designated amount for their desired shoe size. Raghu needs an efficient system to manage his inventory and calculate the total revenue generated from sales based on customer demands.

Problem Statement:

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.

Input Format:

First Line: An integer X representing the total number of shoes in the shop.

Second Line: A space-separated list of integers representing the shoe sizes in the shop.

Third Line: An integer N representing the number of customer requests.

Next N Lines: Each line contains a pair of space-separated values:

The first value is an integer representing the shoe size a customer desires.

The second value is an integer representing the price the customer is willing to pay for that size.

Output Format:

Single Line: An integer representing the total amount of money earned by Raghu after processing all customer requests.

Constraints:

$1 \leq X \leq 1000$ — Raghu's shop can hold between 1 and 1000 shoes.

Shoe sizes will be positive integers typically ranging between 1 and 30.

$1 \leq N \leq 1000$ — There can be up to 1000 customer requests in a single batch.

The price offered by customers will be a positive integer, typically ranging from $5 to $100 per shoe.

For example:

Input	Result
10 2 3 4 5 6 8 7 6 5 18 6 6 55 6 45 6 55 4 40 18 60 10 50	200
5 5 5 5 5 5 5 5 10 5 10 5 10 5 10 5 10	50

Answer:

```
X = int(input())
shoe_sizes = list(map(int, input().split()))
N = int(input())
total_earnings = 0
for _ in range(N):
    size, price = map(int, input().split())
    if size in shoe_sizes:
        total_earnings += price
        shoe_sizes.remove(size)
print(total_earnings)
```

	Input	Expected	Got	
	10 2 3 4 5 6 8 7 6 5 18 6 6 55 6 45 6 55 4 40 18 60 10 50	200	200	
	5 5 5 5 5 5 5 5 10 5 10 5 10 5 10 5 10	50	50	

	Input	Expected	Got	
	4 4 4 6 6 5 4 25 4 25 6 30 6 55 6 55	135	135	

Ex. No. : 12.4 **Date: 05.06.2024**

Register No.: 231401023 **Name: DHARAN S**

Book Genre Categorization

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

Answer:

```
def categorize_books():
    import sys
    input = sys.stdin.read
    data = input().strip().split('\n')
    books_by_genre = {}
    for line in data:
        if not line:
            break
        book, genre = map(str.strip, line.split(',', 1))
        if genre not in books_by_genre:
            books_by_genre[genre] = []
        books_by_genre[genre].append(book)
    for genre, books in books_by_genre.items():
        print(f"{genre}: {', '.join(books)}")
categorize_books()
```

	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	

Ex. No. : 12.5 **Date: 05.06.2024**

Register No.: 231401023 **Name: DHARAN S**

Counting Unique Activity Pairs with a Specific Difference

Background:

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:

$1 \leq n \leq 105$

$-104 \leq nums[i] \leq 104$

$0 \leq k \leq 104$

For example:

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

Answer:

```
def find_unique_pairs(activities, k):
    count = 0
    for i in range(len(activities)):
        for j in range(i+1, len(activities)):
```

```
            if abs(activities[i] - activities[j]) == k:
                count += 1
    return count
n = int(input())
activities = list(map(int, input().split()))
k = int(input())
result = find_unique_pairs(activities, k)
print(result)
```

	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	