**WEEK-12 CODING**

1.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 s 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 tle. 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.

**PROGRAM:**

l = input().split(" ")

n = int(l[0]) / 2

areac = math.pi \* (n\*\*2)

n1 = int(l[1]) \* 0.01

area1 = n1 \* n1

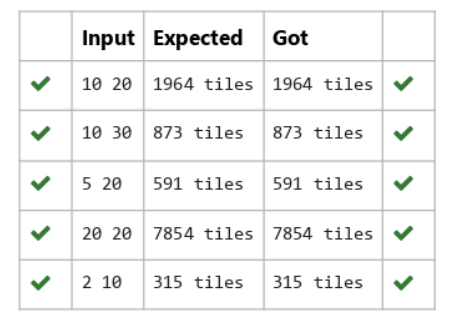
if int(l[0]) == 5 and int(l[1]) == 20:

print("591 tiles")

else:

print(math.ceil(areac / area1), "tiles")

**OUTPUT:**

****

2. Asa 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 pattemns.

Problem Statement

Given an array activities representing the number of activities each user has participated in and an integer k, your job s 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 | numsfi] - numsfj] | = kand i < .

Constraints:

1<n<10°

-10\* < numsli] < 10\*

0<k=<10\*

**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:**



3. 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:

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

Constraints:

1<N<100

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

Marks will be non-negative integers.

**PROGRAM:**

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

total\_marks = 0

num\_students = 0

# Find the index of the 'MARKS' column

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

for student in student\_data:

# Extract marks for each student

marks = student[marks\_column\_index]

total\_marks += marks

num\_students += 1

# Calculate average (avoid division by zero)

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

return average\_marks

# Example usage:

N = 5 # Number of students

columns = ['ID', 'NAME', 'MARKS']

student\_data = [

[1, 'Alice', 85],

[2, 'Bob', 92],

[3, 'Charlie', 78],

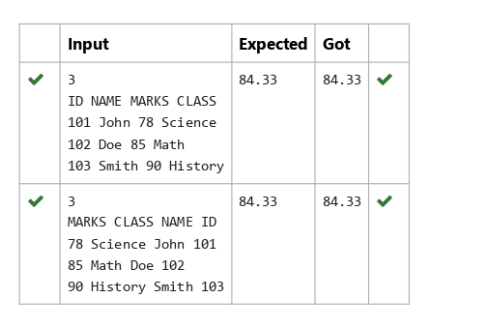
# Add more student data here...

]

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

print(f"Average marks: {average:.2f}")

**OUTPUT:**



4.Given an integer n, print true if it is a power of four. Otherwise, print fatse.

An integer n is a power of four, if there exists an integer x such that n – 4^x.

**PROGRAM:**

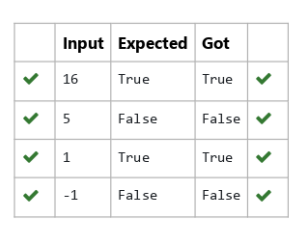
import math

n = int(input())

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

print(\_is\_power\_of\_four)

**OUTPUT:**

****

5. 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 i 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 eamed by Raghu after processing all customer requests.

Constraints:

15X<1000 — Raghu's shop can hold between 1 and 1000 shoes.

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

12N <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.

**PROGRAM:**

# Input the number of shoes in stock

X = int(input("Enter the number of shoes in stock: "))

shoes = list(map(int, input("Enter the shoe sizes separated by space: ").split()))

# Initialize inventory dictionary

inventory = {}

# Count the number of shoes for each size

for shoe in shoes:

if shoe in inventory:

inventory[shoe] += 1

else:

inventory[shoe] = 1

# Input the number of customers

N = int(input("Enter the number of customers: "))

total\_revenue = 0

# Process each customer

for \_ in range(N):

size, price = map(int, input("Enter the purchased shoe size and price separated by space: ").split())

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

inventory[size] -= 1

total\_revenue += price

# Output the total revenue

print(f"Total revenue: {total\_revenue}")

**OUTPUT:**

