**07 - Functions**

**Ex. No. : 7.1 Date: 09.05.2024**

**Register No.: 231401122 Name:**VARSHA LEENA .I



**Abundant Number**

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):

l,s=[],0

for i in range(1,int(n//2)+1):

if(n%i==0):

l.append(i)

for i in l:

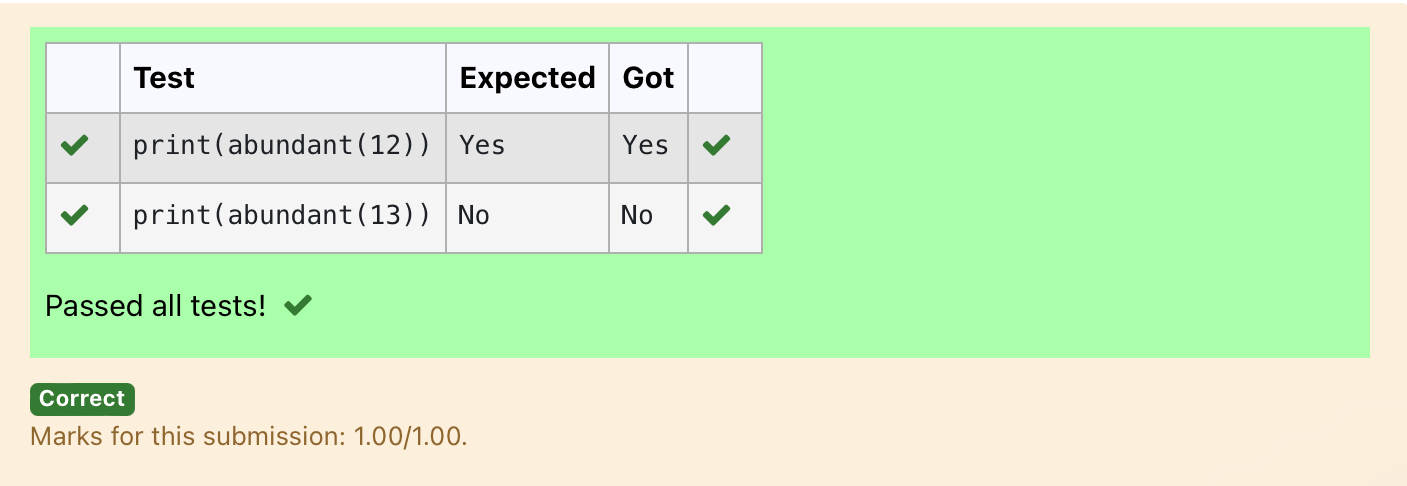
s+=i

if(s>n):

return("Yes")

else:

return("No")



**Ex. No. : 7.2 Date: 09.05.2024**

**Register No.: 231401122 Name:**VARSHA LEENA .I



**Automorphic number or not**

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):

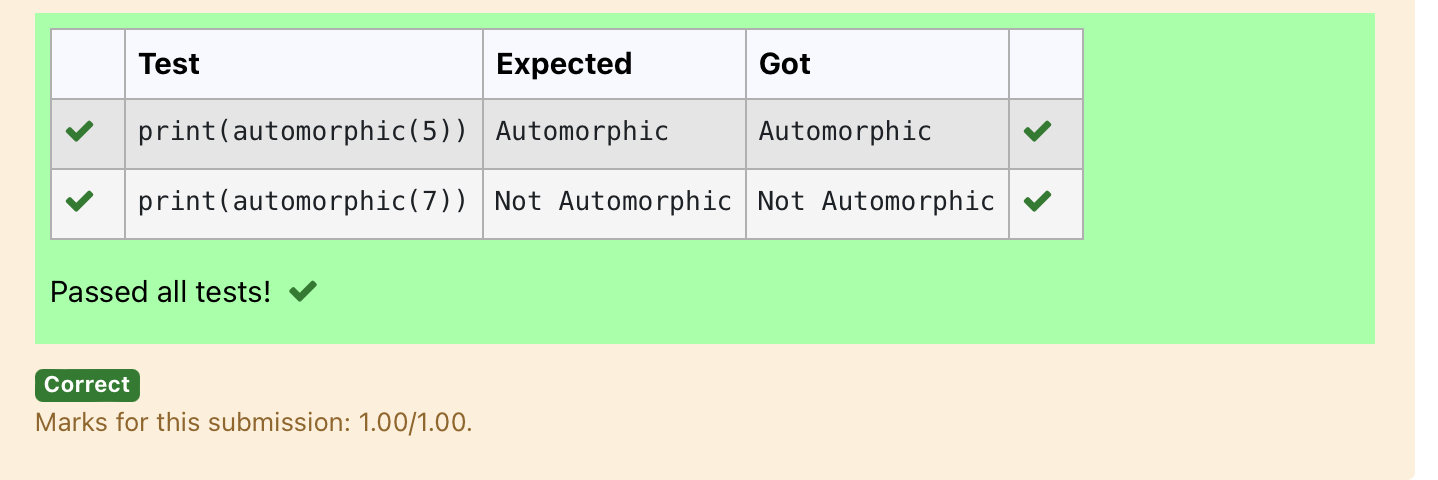
a=str(n\*n)

if(int(a[-1])==n):

return("Automorphic")

else:

return("Not Automorphic")



**Ex. No. : 7.3 Date: 09.05.2024**

**Register No.: 231401122 Name:**VARSHA LEENA .I



**Check Product of Digits**

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 |

defproductDigits(n):

a=str(n)

s,p=0,1

for i in range(0,len(a),2):

s+=int(a[i])

for i in range(1,len(a),2):

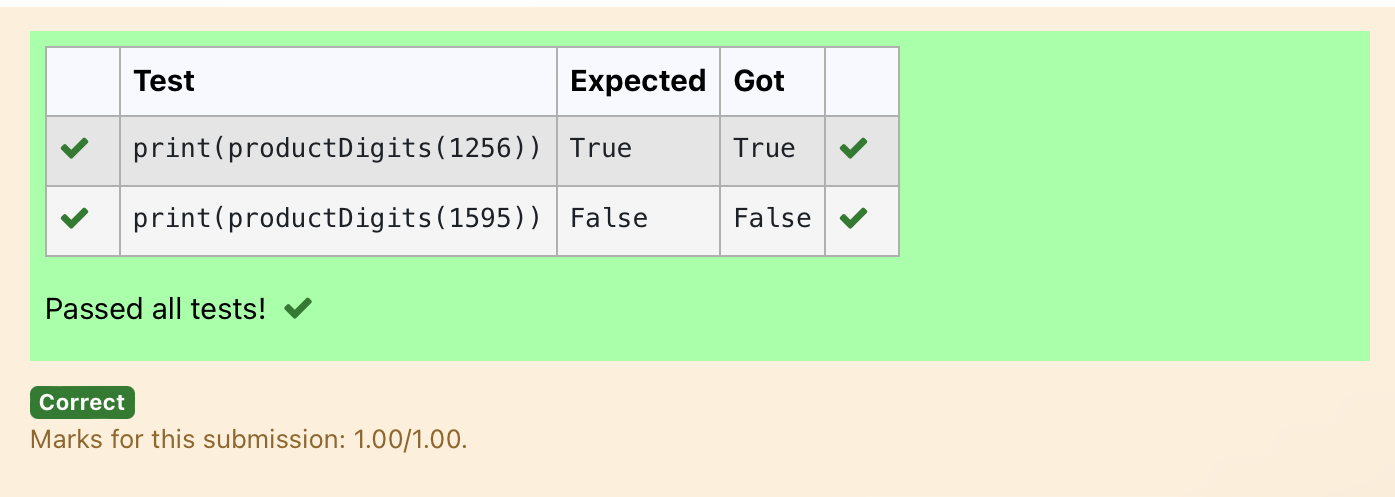
p\*=int(a[i])

if(p%s==0):

return("True")

else:

return("False")



**Ex. No. : 7.4 Date: 09.05.2024**

**Register No.: 231401122 Name:**VARSHA LEENA .I



**Christmas Discount**

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 python code 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 |

defchristmasDiscount(n):

res=0

while n!=0:

rem=n%10

flag=0

for i in range(1,rem+1):

if rem%i==0:

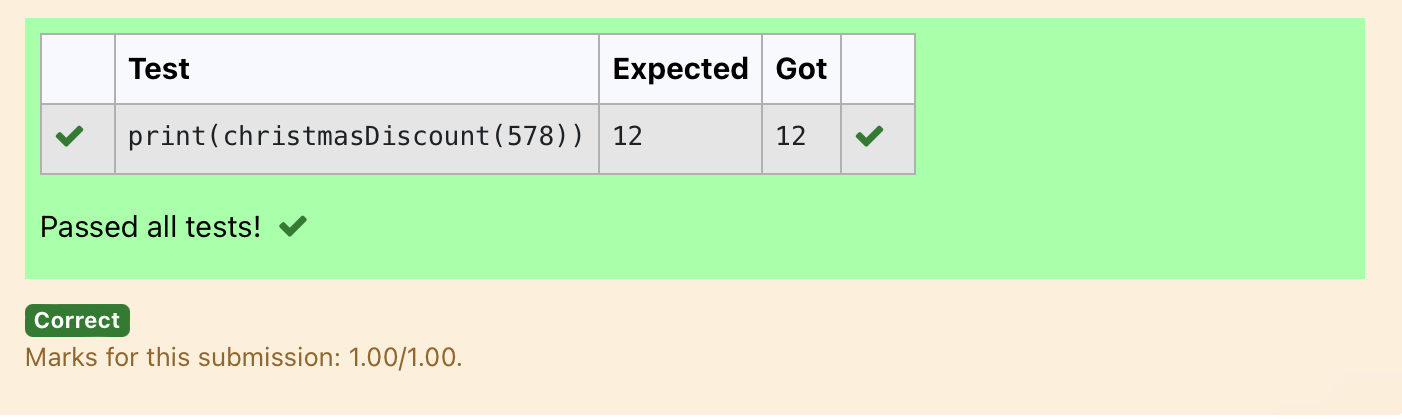
flag+=1

if flag==2:

res=res+rem

n=n//10

return res



**Ex. No. : 7.5 Date: 09.05.2024**

**Register No.: 231401122 Name:**VARSHA LEENA .I



**Coin Change**

complete function to implement coin change making problem i.e. finding the minimum

number of coins of certain denominations that add up to given amount of money.

The only available coins are of values 1, 2, 3, 4

Input Format:

Integer input from stdin.

Output Format:

return the minimum number of coins required to meet the given target.

Example Input:

16

Output:

4

Explanation:

We need only 4 coins of value 4 each

Example Input:

25

Output:

7

Explanation:

We need 6 coins of 4 value, and 1 coin of 1 value

**defcoinChange(amount):**

**# Available coin denominations**

**coins = [1, 2, 3, 4]**

**# Initialize a list to store the minimum number of coins for each amount from 0 to the target amount**

**dp = [float('inf')] \* (amount + 1)**

**dp[0] = 0 # Base case: 0 coins needed to make amount 0**

**# Iterate through all amounts from 1 to the target amount**

**for i in range(1, amount + 1):**

**# Iterate through all available coin denominations**

**for coin in coins:**

**# If the current coin denomination is less than or equal to the current amount**

**if coin <= i:**

**# Update dp[i] to be the minimum between its current value and dp[i - coin] + 1**

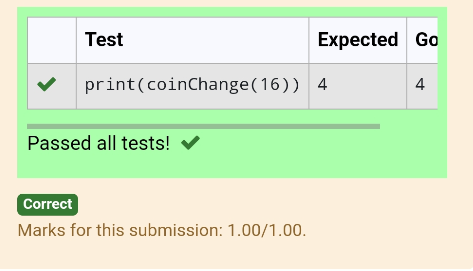
**dp[i] = min(dp[i], dp[i - coin] + 1)**

**# The result is stored at dp[amount]**

**return dp[amount]**

**amount = int(input())**

**print(coinChange(amount))**

****