07 - Functions

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

<u>Abundant Number</u>

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

```python
def abundant(n):

    if n <= 0:

        return "No"

    divisors_sum = sum(i for i in range(1, n) if n % i == 0)

    if divisors_sum > n:

        return "Yes"

    else:

        return "No"
```

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

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".

```python
def automorphic(n):
    # Check if the input is a valid positive integer
    if not isinstance(n, int) or n < 0:
        return "Invalid input"


    # Calculate the square of the number
    square = n * n


    # Convert both the number and its square to strings
    n_str = str(n)
    square_str = str(square)


    # Check if the square ends with the number
    if square_str.endswith(n_str):
        return "Automorphic"
    else:
        return "Not Automorphic"
```

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

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.

```python
def productDigits(n):
    dup=str(n)
    pro,sum2=1,0
    l1=[int(dup[i]) for i in range(0,len(dup),3)]
    l2=[int(i) for i in dup]
    for i in l1:
        pro*=i
    #print(pro,sum(l2))
    if sum(l2)%pro!=0:
        return True
    else:
        return False
```

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

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

$1 <= orderValue < 10e^{100000}$

```python
def christmasDiscount(n):
    prime= {'2', '3', '5', '7'}
    discount = 0

    for char in str(n):
        if char in prime:
            discount += int(char)

    return discount
```

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

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

```python
def coinChange(n):
    f=n//4
    t=(n%4)//3
    tw=((n%4)%3)//2
    o=((n%4)%3)%2
    return(f+t+tw+o)
```

Input Format:

Take a number in the form of String from stdin.

Output Format:

Print the difference between sum of even and odd digits

Example input:

1453

Output:

1

Explanation:

Here, sum of even digits is 4 + 3 = 7

sum of odd digits is 1 + 5 = 6.

Difference is 1.

Note that we are always taking absolute difference

Difference Sum

Given a number with maximum of 100 digits as input, find the difference between the sum of odd and even position digits.

```python
def differenceSum(n):
    odd=[]
    even=[]
    num=str(n)
    for i in range(len(num)):
        if(i%2==0):
            even.append(int(num[i]))
        else:
            odd.append(int(num[i]))
    o=sum(odd)
    e=sum(even)
    ans=o-e
    return ans
```

For example:

Test	Result
print(checkUgly(6))	ugly
print(checkUgly(21))	not ugly

Ugly number

A number is considered to be ugly if its only prime factors are 2, 3 or 5.
[1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, …] is the sequence of ugly numbers.
Task:
complete the function which takes a number n as input and checks if it's an ugly number.
return ugly if it is ugly, else return not ugly
Hint:
An ugly number U can be expressed as: U = 2^a * 3^b * 5^c, where a, b and c are nonnegative integers.

```
def checkUgly(n):

    if n <= 0:

        return "not ugly"


    for factor in [2, 3, 5]:

        while n % factor == 0:

            n //= factor


    return "ugly" if n == 1 else "not ugly"
```