Started on	Friday, 24 May 2024, 9:26 AM
State	Finished
Completed on	Friday, 24 May 2024, 9:51 AM
Time taken	25 mins 16 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 1.00 out of 1.00

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 algorithm 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

Answer: (penalty regime: 0 %)

Reset answer

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def christmasDiscount(orderValue):
    prime_digits = [2, 3, 5, 7]  # List of prime digits
    total_discount = 0

    for digit in str(orderValue):
        if int(digit) in prime_digits:
            total_discount += int(digit)

    return total_discount
```

	Test	Expected	Got	
✓	print(christmasDiscount(578))	12	12	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

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`

Answer: (penalty regime: 0 %)

Reset answer

```
def productDigits(n):
    num_str = str(n)
    even_product = 1
    odd_sum = 0
    for i in range(len(num_str)):
        digit = int(num_str[i])
        if i % 2 != 0:  # Even index
            even_product *= digit
        else:  # Odd index
            odd_sum += digit
    return even_product % odd_sum == 0
```

	Test	Expected	Got	
✓	`print(productDigits(1256))`	`True`	`True`	✓
✓	`print(productDigits(1595))`	`False`	`False`	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

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.

For example:

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

Answer: (penalty regime: 0 %)

Reset answer

```
def checkUgly(n):
    if n <= 0:
        raise ValueError("Input number must be a positive integer.")
    for prime in [2, 3, 5]:
        while n % prime == 0:
            n /= prime
    if n == 1:
        return "ugly"
    else:
        return "not ugly"
```

	Test	Expected	Got	
✓	print(checkUgly(6))	ugly	ugly	✓
✓	print(checkUgly(21))	not ugly	not ugly	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

Reset answer

```
def automorphic(n):
    if not isinstance(n, int) or n < 0:
        return "Invalid input"

    # Calculate the square of the number
    square = n ** 2

    # Convert 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"
```

	Test	Expected	Got	
✓	print(automorphic(5))	Automorphic	Automorphic	✓
✓	print(automorphic(7))	Not Automorphic	Not Automorphic	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

Reset answer

```python
def coinChange(target):
    """
    Finds the minimum number of coins required to meet the given target amount.
    """
    # Create a list to store the minimum number of coins for each amount from 0 to target
    coins = [float('inf')] * (target + 1)
    coins[0] = 0  # 0 coins are needed to make 0 amount

    # Iterate over each amount from 1 to target
    for amount in range(1, target + 1):
        # Iterate over each available coin denomination
        for coin in [1, 2, 3, 4]:
            # If the coin is not more than the current amount
            if coin <= amount:
                # Update the minimum number of coins for the current amount
                coins[amount] = min(coins[amount], coins[amount - coin] + 1)

    # If it's not possible to make the target amount, return -1
    if coins[target] == float('inf'):
        return -1
    else:
        return coins[target]
```

	Test	Expected	Got	
✓	`print(coinChange(16))`	4	4	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week9_MCQ

Jump to...

Searching ►