Started on	Saturday, 8 June 2024, 5:28 AM
State	Finished
Completed on	Saturday, 8 June 2024, 5:36 AM
Time taken	8 mins 6 secs
Marks	4.00/5.00
Grade	**80.00** 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_digit(digit):
   #Check if the digit is a prime number (2, 3, 5, or 7)
   return digit in {'2', '3', '5', '7'}
def christmasDiscount(n):
    # Convert the total bill amount to a string
    orderValue_str=str(n)

    discount=0

    #Iterate through each digit of the total bill amount

    for digit in orderValue_str:
        #Check if the digit is a prime number
        if is_prime_digit(digit):
             discount+=int(digit)

    return 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

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

```
def coinChange(target):
    coins = [1, 2, 3, 4]
    dp = [float('inf')] * (target + 1)
    dp[0] = 0
    for i in range(1, target + 1):
        for coin in coins:
            if coin <= i:
                dp[i] = min(dp[i], dp[i - coin] + 1)
    return dp[target]


```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

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(number):
    if number < 0:
        return "Invalid input"
    square = number * number
    number_str = str(number)
    square_str = str(square)
    if square_str.endswith(number_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 **4**

Not answered

Mark 0.00 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):

```

Question **5**

Correct

Mark 1.00 out of 1.00

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`

Answer: (penalty regime: 0 %)

Reset answer

```
def abundant(number):
    divisor_sum = sum([divisor for divisor in range(1, number) if number % di
    if divisor_sum > number:
        return "Yes"
    else:
        return "No"
```

	Test	Expected	Got	
✓	`print(abundant(12))`	Yes	Yes	✓
✓	`print(abundant(13))`	No	No	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week9_MCQ

Jump to...

Searching ►