Started on	Saturday, 25 May 2024, 9:28 AM
State	Finished
Completed on	Saturday, 25 May 2024, 9:34 AM
Time taken	5 mins 54 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

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):
    # Available coin denominations
    coins = [1, 2, 3, 4]

    # Create a list to store the minimum coins needed for each amount up to the target
    dp = [float('inf')] * (target + 1)

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

    # Fill the dp array
    for i in range(1, target + 1):
        for coin in coins:
            if i - coin >= 0:
                dp[i] = min(dp[i], dp[i - coin] + 1)

    # The answer will be in dp[target]
    return dp[target]


```

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

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(number):
    # Convert the number to a string to easily access individual digits
    num_str = str(number)

    # Initialize product and sum
    product_even = 1
    sum_odd = 0

    # Iterate through the digits with their positions
    for i, digit in enumerate(num_str):
        digit = int(digit)
        if (i + 1) % 2 == 0:  # Even place (1-indexed)
            product_even *= digit
        else:  # Odd place (1-indexed)
            sum_odd += digit

    # Check if product of even place digits is divisible by sum of odd place digits
    if sum_odd == 0:
        return False  # Avoid division by zero
    return product_even % sum_odd == 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

Given a number with maximum of 100 digits as input, find the difference between the sum

of odd and even position digits.

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

Answer: (penalty regime: 0 %)

Reset answer

```
def differenceSum(num):
    num_str = str(num)  # Convert the integer to a string
    # Initialize sums for even and odd position digits
    even_sum = 0
    odd_sum = 0

    # Iterate through each digit in the number
    for i in range(len(num_str)):
        digit = int(num_str[i])

        # Check if the position is even or odd
        if (i + 1) % 2 == 0:
            even_sum += digit
        else:
            odd_sum += digit

    # Calculate the absolute difference between even and odd sums
    difference = abs(even_sum - odd_sum)

    return difference
```

	Test	Expected	Got	
✓	`print(differenceSum(1453))`	1	1	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

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(n):
    if n < 1:
        return "No"

    proper_divisors_sum = sum(divisor for divisor in range(1, n) if n % divisor == 0)

    return "Yes" if proper_divisors_sum > n else "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.

Question **5**

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

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

    # Convert both numbers to strings to compare the ending
    str_n = str(n)
    str_square = str(square)

    # Check if the square ends with the number
    if str_square.endswith(str_n):
        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.

◄ Week9_MCQ

Jump to...

Searching ►