WEEK 9

Functions:Built-in functions,User-defined functions,Recursive functions

Top of Form

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 %)

```
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 i - coin >= 0:
```

```
                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 **2**
Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 %)

```
def checkUgly(n):
    if n <= 0:
        return "not ugly"
```

```
    while n % 2 == 0:
        n //= 2
    while n % 3 == 0:
        n //= 3
    while n % 5 == 0:
        n //= 5

    return "ugly" if n == 1 else "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 **3**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 %)

```
def automorphic(num):
    if num <= 0:
        return "Invalid input"

    square = num * num
    num_str = str(num)
    square_str = str(square)

    if square_str[-len(num_str):] == num_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**
Correct

Mark 1.00 out of 1.00

Flag question

Question text
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 %)

```
def abundant(n):
    if n <= 0:
        return "No"

    sum_divisors = 0
    for i in range(1, n):
        if n % i == 0:
```

```
            sum_divisors += i

    if sum_divisors > n:
        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.

Question **5**
Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 %)

```
def productDigits(num):
    if num <= 0:
        return "FALSE"

    num_str = str(num)
    even_product = 1
    odd_sum = 0

    for i in range(len(num_str)):
        digit = int(num_str[i])
        if (i + 1) % 2 == 0:
            even_product *= digit
        else:
            odd_sum += digit

    return "True" if even_product % odd_sum == 0 else "False"
```

	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.

Bottom of Form
Top of Form