GE19211 / GE23233 / GE23231 - PSPP/PUP

Quiz navigation

1 2 3 4 5

Show one page at a time

Finish review

Started on	Saturday, 25 May 2024, 8:34 AM
State	Finished
Completed on	Saturday, 25 May 2024, 9:10 AM
Time taken	36 mins 35 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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):
    A = sum([ i for i in range(1, n) if n % i == 0])
    return "Yes" if A > 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 **2**

Correct

Mark 1.00 out of 1.00

Flag question

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):
    A = n **2
    return "Automorphic" if str(A).endswith(str(n)) else "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 **3**

Correct

Mark 1.00 out of 1.00

Flag question

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(n):
    # Ensure the input is a string
    n = str(n)

    # Initialize sum of even and odd digits
    sum_even = 0
    sum_odd = 0

    # Iterate over the digits and calculate the sum
    for i, digit in enumerate(n):
        # Check if the position is even or odd
        if i % 2 == 0:
            sum_even += int(digit)
        else:
            sum_odd += int(digit)

    # Calculate and return the difference
    return abs(sum_even - sum_odd)

# Read input until EOF3 error
while True:
    try:
        number = input()
        print(differencesum(number))
    except EOFError:
        break
```

	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

Flag question

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(productOfDigits(1256))`	True
`print(productOfDigits(1595))`	False

Answer: (penalty regime: 0 %)

Reset answer

```
def productOfDigits(number):
    num_str = str(number)
    product_even = 1
    sum_odd = 0
    for i in range(len(num_str)):
        digit = int(num_str[i])
        if (i + 1) % 2 == 0:
            product_even *= digit
        else:
            sum_odd += digit
    if sum_odd == 0:
        return False
    return product_even % sum_odd == 0

if __name__ == "__main__":
    try:
        number = int(input())
        if productOfDigits(number):
            print("TRUE")
        else:
            print("FALSE")
    except EOFError:
        pass
    except ValueError:
        print("ValueError: Invalid input. Please enter a positive integer.")
```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

Flag question

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(n):
    coins = [1, 2, 3, 4]
    dp = [0] + [float('inf')] * n
    for amount in range(1, n + 1):
        dp[amount] = min(dp[amount - coin] + 1 for coin in coins if amount >= coin)
    return dp[n]
```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review