WEEK-05-CODING-Functions-User Defined

1.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`

Coding:

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

    for p in [2,3,5]:
        while n % p ==0:
            n //= p
    return "ugly" if n==1 else "not ugly"
```

Output:

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

Passed all tests! ✓

2. Write a function that returns the value of a+aa+aaa+aaaa with a given digit as the value of a.

Suppose the following input is supplied to the program:

9

Then, the output should be:

9+99+999+9999=11106

Sample Input Format:

9

Sample Output format:

11106

For example:

Test	Result
`print(Summation(8))`	`9872`

Coding:

```
def Summation(n):
    a=str(n)
    t1=int(a)
    t2=int(a*2)
    t3=int(a*3)
    t4=int(a*4)

    total = t1+t2+t3+t4

    return total
```

Output:

	Test	Expected	Got	
✓	`print(Summation(8))`	`9872`	`9872`	✓
✓	`print(Summation(10))`	`10203040`	`10203040`	✓

Passed all tests! ✓

3. A strobogrammatic number is a number that looks the same when rotated 180 degrees (looked at upside down).

Write a program to determine if a number is strobogrammatic. The number is represented as a string.

Example 1:

Input:

69

Output:

true

Example 2:

Input:

88

Output:

true

Example 3:

Input:

962

Output:

false

Example 4:

Input:

1

Output:

true

For example:

Test	Result
`print(Strobogrammatic(69))`	`true`
`print(Strobogrammatic(962))`	`false`

Coding:

```
def Strobogrammatic(n: int) -> bool:
    s={'0':'0', '1':'1', '6':'9', '8':'8', '9':'6'}
```

```
n=str(n)
l,r = 0 , len(n) - 1
while l <= r:
    if n[l] not in s or n[r] not in s:
        return 'false'
    if s[n[l]] != n[r]:
        return 'false'
    l +=1
    r -=1
return 'true'
```

Output:

	Test	Expected	Got	
✓	print(Strobogrammatic(69))	true	true	✓
✓	print(Strobogrammatic(88))	true	true	✓
✓	print(Strobogrammatic(962))	false	false	✓

Passed all tests! ✓

4. 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

Coding:

```
def christmasDiscount(n):
    p={'2','3','5','7'}
    v=0
    for d in str(n):
        if d in p:
            v +=int(d)
    return v

#print(christmasDiscount(n))
```

Output:

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

Passed all tests! ✓

5. 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

Coding:

```
def coinChange(n):
    if n==0:
        return 0
    dp = [float('inf')]*(n+1)
    dp[0]=0

    c=[1,2,3,4]

    for a in range(1, n+1):
        for c1 in c:
            dp[a] = min(dp[a],dp[a-c1] + 1)
    return dp[n] if dp[n] != float('inf') else -1
```

Output:

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

Passed all tests! ✓

6. Complete a Recursive Function to find if a given number N can be expressed as a sum of two prime numbers.

Note: YOU MUST OPTIMIZE the logic to find whether a number is prime or not, as very large prime numbers are provided as input. If the logic is not optimized your program will NOT get executed within the given time limit.

Input Format:

First line contains number N.

Output Format:

Return either yes or no.

Boundary Conditions / Constraints:

3 <= N <= 10^9

Example Input/Output 1:

Input:

20

Output:

yes

Input:

23

Output:

no

Explanation:

20 can be expressed as 17+3

23 cannot be expressed as sum of two primes

For example:

Test	Result
`print(checkPrimeSum(20))`	`yes`
`print(checkPrimeSum(23))`	`no`

Coding:

```
import math
def ss(l):
    s=[True]*(l+1)
    s[0]=s[1]=False
    for st in range(2, int(math.sqrt(l))+1):
        if s[st]:
            for m in range(st*st,l+1,st):
                s[m]=False
    return {num for num,p in enumerate(s) if p}

def checkPrimeSum(n):
    if n<4:
        return"no"
    a=ss(n)
    for b in a:
        if(n-b) in a:
            return "yes"
```

Output:

	Test	Expected	Got	
✓	`print(checkPrimeSum(20))`	yes	yes	✓

Passed all tests! ✓

7. Complete the recursive function to return Binary Equivalent of an Integer using Recursion.

Sample Test Cases

Test Case 1

Input

10

Output

1010

Test Case 2

Input

257

Output

100000001

For example:

Test	Result
`print(binayNumber(10))`	`1010`
`print(binayNumber(257))`	`100000001`

Coding:

```
def binayNumber(n):
    if n==0:
        return "0"
    elif n==1:
        return "1"
    return binayNumber(n//2)+str(n%2)
```

Output:

	Test	Expected	Got	
✓	print(binayNumber(10))	1010	1010	✓
✓	print(binayNumber(257))	100000001	100000001	✓

Passed all tests! ✓

8. The notion of a palindrome was introduced previously. In this exercise you will write a recursive function that determines whether or not a string is a palindrome. The empty string is a palindrome, as is any string containing only one character. Any longer string is a palindrome if its first and last characters match, and if the string formed by removing the first and last characters is also a palindrome.

Write a program that reads a string from the user and uses your recursive function to determine whether or not it is a palindrome. Then your program should display an appropriate message for the user.

Sample Input

malayalam

Sample Output

That was a palindrome!

Sample Input

madan

Sample Output

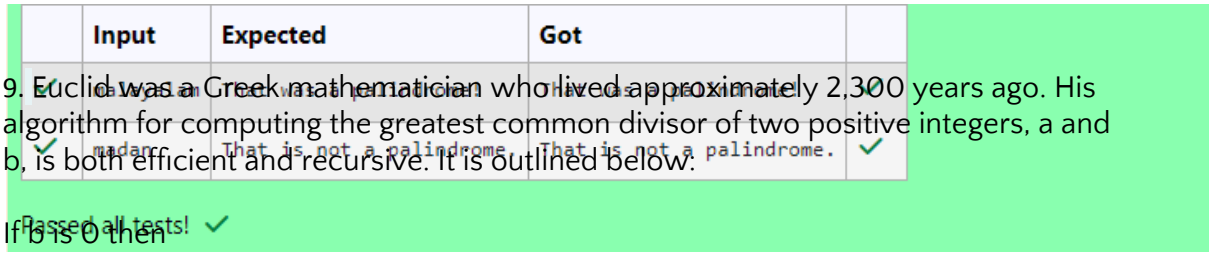
That is not a palindrome.

Coding:

```
def isPalindrome(s):
    # Base case: The empty string is a palindrome. So is a string containing only 1 character.
    if len(s) <= 1:
       return True
    if s[0] ==s[-1]:
        return isPalindrome(s[1:-1])
    else:
        return False

    # Recursive case: The string is a palindrome only if the first and last characters match,
and
    # the rest of the string is a palindrome


# Check whether or not a string entered by the user is a palindrome
# Read the string from the user
```

```
line = input().strip()
# Check its status and display the result
if isPalindrome(line):
    print("That was a palindrome!")

else:
    print("That is not a palindrome.")
```

Output:



	Input	Expected	Got	
✓	madam	That is a palindrome.	That is a palindrome.	✓
✓	madamc	That is not a palindrome.	That is not a palindrome.	✓

Passed all tests! ✓

9. Euclid was a Greek mathematician who lived approximately 2,300 years ago. His algorithm for computing the greatest common divisor of two positive integers, a and b, is both efficient and recursive. It is outlined below:

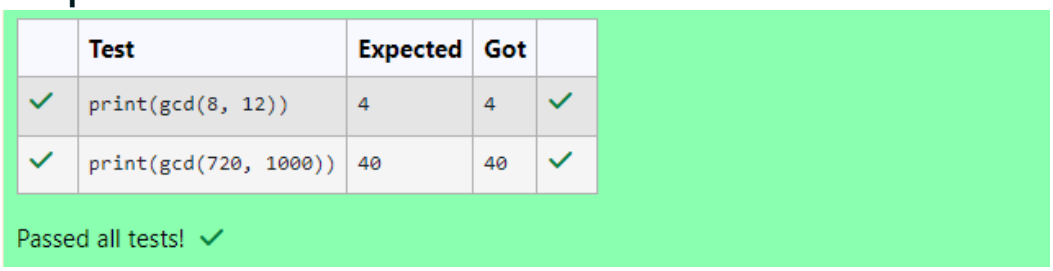
If b is 0 then
    return a
Else
    Set c equal to the remainder when a is divided by b
    Return the greatest common divisor of b and c

Write a Recursive funtion that implements Euclid's algorithm and uses it to determine the greatest common divisor of two integers entered by the user. Test your program with some very large integers. The result will be computed quickly, even for huge numbers consisting of hundreds of digits, because Euclid's algorithm is extremely efficient.

Coding:

```
def gcd(a,b):
    if b == 0:
        return a
    else:
        return gcd(b,a%b)
```

Output:



	Test	Expected	Got	
✓	print(gcd(8, 12))	4	4	✓
✓	print(gcd(720, 1000))	40	40	✓

Passed all tests! ✓

10. Given an integer number and you have to count the digits using recursion using Python program. In
this program, you will be reading an integer number and counting the total digits, using a function countDigits() which will take a number as an argument and return the count after recursion process.
Input Format: The first and only line of the input contains a single integer n
Output Format: Output a single line denoting the number of digits in n.

For example:

Test	Result
`print(countDigits(800))`	3

Coding:

```
def countDigits(n):
    if n<0:
        n = -n
    if n< 10:
        return 1
    else:
        return 1+ countDigits(n//10)
```

Output:

	Test	Expected	Got	
✓	`print(countDigits(12345))`	5	5	✓
✓	`print(countDigits(800))`	3	3	✓

Passed all tests! ✓