WEEK 1

2# Write a Python program to detect whether given integer is perfect or not ?

a=int(input("enter your number: "))

sum=0

for i in range(1,a):

    if(a % i == 0):

        sum = sum + i

if(sum==a):

    print(a,"is perfect number")

else:

    print(a,"is not a perfect number")

3# Write a Python program to detect whether given integer is Armstrong Integer or not?

num = int(input("Enter a number: "))

sum = 0

temp = num

while temp > 0:

   digit = temp % 10

   sum += digit \*\* 3

   temp //= 10

if num == sum:

   print(num,"is an Armstrong number")

else:

print(num,"is not an Armstrong number")

WEEK 2

1# Write a program to detect whether a number is prime or not.

num = int(input("Enter a number: "))

if num <= 1:

    print(f"{num} is not a prime number.")

else:

    is\_prime = True

    for i in range(2, int(num \*\* 0.5) + 1):

        if num % i == 0:

            is\_prime = False

            break

    if is\_prime:

        print(f"{num} is a prime number.")

    else:

        print(f"{num} is not a prime number.")

2# Write a function to compute GCD of two integers.

a = int(input("Enter the first number: "))

b = int(input("Enter the second number: "))

while b != 0:

    a, b = b, a % b

print(f"The GCD of the two numbers is: {a}")

3# Write a program to detect whether two numbers are relatively prime or not?

num\_1=int(input("Enter first number: "))

num\_2=int(input("Enter secoond number: "))

a, b= num\_1, num\_2

while b!=0:

    a, b =b, a % b

if a == 1:

    print(f"{num\_1} and {num\_2} are relatively prime")

else:

    print(f"{num\_1} and {num\_2} are not relatively prime")

4# Write a function to display all ‘n’ narcissistic number.

def is\_narcissistic(num):

    digits = [int(digit) for digit in str(num)]  #digits of the number

    num\_digits = len(digits)

    return num == sum(digit \*\* num\_digits for digit in digits)

def display\_narcissistic\_numbers(n):

    print(f"Narcissistic numbers up to {n}:")

    for num in range(1, n + 1):

        if is\_narcissistic(num):

            print(num)

n = int(input("Enter a number n: "))

display\_narcissistic\_numbers(n)

WEEK 3

1# Write a function to check whether a number is in the form of 2k or not.

def is\_power\_of\_two(n):

    return n > 0 and (n & (n - 1)) == 0

a=int(input("Enter a number to check if it's in the form of 2^k or not : "))

if(is\_power\_of\_two(a)):

    print(f"{a} is in the form of 2^k")

else:

    print(f"{a} is not in the form of 2^k")

2# Write a function to check whether a prime number is Mersenne prime or not?

def mersenne\_prime(n):

    # Check if n + 1 is a power of 2

    if (n + 1) & n == 0:

        for i in range(2, int(n/2) + 1):

            if n % i == 0:

                return False

        return True

    return False

a=int(input("Enter a number to check whether it's mersenne prime or not: "))

if(mersenne\_prime(a)):

    print(f"{a} is mersenne prime!")

else:

    print(f"{a} is not mersenne prime")

3# Write a program to evaluate Euler Totients (Totatives) of a given integer.

import math

def euler\_totient(n):

    count = 0

    for i in range(1, n + 1):

        if math.gcd(i, n) == 1:

            count += 1

    return count

n = int(input("Enter a number: "))

print(f"The Euler's Totient of {n} is: {euler\_totient(n)}")

4# Implement a function that takes two arguments (r, n) and computer order of r under modulo(n) operation.

def order(r, n):

    if r % n == 0:

        return -1  # r is divisible by n, so the order doesn't exist.

    k = 1

    power = r % n  # Start with r^1 % n

    while power != 1:

        k += 1

        power = (power \* r) % n

    return k

r = 2

n = 7

print("Order of", r, "modulo", n, "is:", order(r, n))