**Experiment No. 2**

**Aim:** To implement Euler’s Totient Function

**Theory:** Euler’s totient function, written **,** and defined as the number of positive integers less than n and relatively prime to . i.e. .

. Listing all relatively prime to between [1, 24]: {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24}.

It can be easily inferred that, for any prime .

.

It also have important properties for prime factorization:

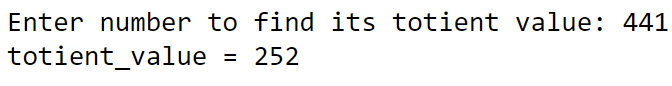
1. ---- {where, are distinct prime}.

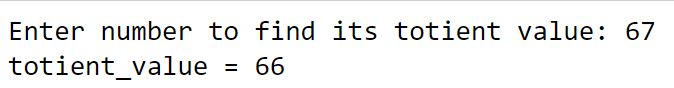
2. --- {where, is prime}.

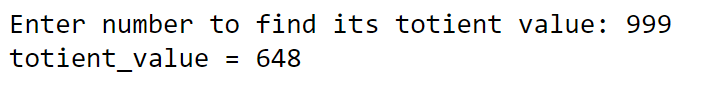
**Implementation:**

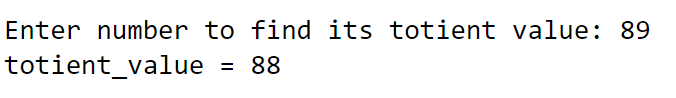
def gcd(a,b):  
 if not b:  
 return a  
 return gcd(b,a%b)  
  
def coprime(a,b):  
 return gcd(a,b) == 1  
  
def totient(n):  
 assert n>0, "Number should be greater than 0"  
 totient\_set ={1}  
 i = n-1  
 while(i>1):  
 if coprime(n,i):  
 totient\_set.add(i)  
 i -= 1  
   
 return totient\_set, len(totient\_set)  
   
totient\_set, totient\_value = totient(int(input("Enter number to find its totient value: ")))  
  
print(f"totient\_value = {totient\_value}")

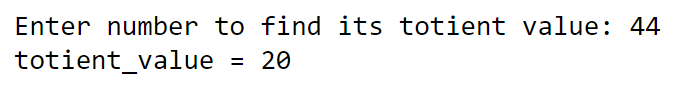
**Output:**

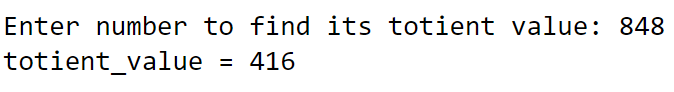
****

****

****

****

****

****