Experiment No. 2

Aim: To implement Euler's Totient Function

Theory: Euler's totient function, written $\boldsymbol{\varphi(n)}$, and defined as the number of positive integers less than n and relatively prime to n. i.e. $|\boldsymbol{GCD(n,x) = 1}|$ $\boldsymbol{where\ x \in Z_n\ (Residues\ of\ n)}$.

$e.g.\ \varphi(25) = 20$. Listing all relatively prime to 25 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 p.

$\boldsymbol{\varphi(p) = p - 1}$. $e.g.\ \varphi(17) = 16$

It also have important properties for prime factorization:

1. $\boldsymbol{\varphi(n) = \varphi(p \times q) = \varphi(p) \times \varphi(q) = (p-1) \times (q-1)}$ ---- {where, $p\&q$ are distinct prime}. $e.g.\ \varphi(35) = \varphi(7 \times 5) = \varphi(7) \times \varphi(5) = (7-1) \times (5-1) = 24$.

2. $\boldsymbol{\varphi(n) = \varphi\left(p^k\right) = (p^k - p^{k-1})}$ --- {where, p is prime}.

$e.g.\ \varphi(441) = \varphi(7^2 \times 3^2) = (7^2 - 7^1) \times (3^2 - 3^1) = 252$.

Implementation:

```python
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 t
otient value: ")))

print(f"totient_value = {totient_value}")
```

Output:

```
Enter number to find its totient value: 441
totient_value = 252
```

```
Enter number to find its totient value: 67
totient_value = 66
```

```
Enter number to find its totient value: 999
totient_value = 648
```

```
Enter number to find its totient value: 89
totient_value = 88
```

```
Enter number to find its totient value: 44
totient_value = 20
```

```
Enter number to find its totient value: 848
totient_value = 416
```