Om Shete - C3 - 2103163

Experiment No 4

Aim: Implementation of Euler's Totient Function.

Theory:

Euler's Totient function? (n) for an input n is the count of numbers in {1, 2, 3, …, n-1} that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1.

Example:

?(1) = 1
gcd(1, 1) is 1

?(2) = 1
gcd(1, 2) is 1, but gcd(2, 2) is 2.

?(3) = 2
gcd(1, 3) is 1 and gcd(2, 3) is 1

?(4) = 2
gcd(1, 4) is 1 and gcd(3, 4) is 1

?(5) = 4
gcd(1, 5) is 1, gcd(2, 5) is 1,
gcd(3, 5) is 1 and gcd(4, 5) is 1

?(6) = 2
gcd(1, 6) is 1 and gcd(5, 6) is 1,

Euler's totient function one may use to know how many prime numbers are coming up to the given integer 'n.' It is also called an arithmetic function. Two things are important for an application or use of Euler's totient function. One is that the gcd formed from the given integer 'n' should be multiplicative. The other is that the numbers of gcd should be the prime numbers only. The integer 'n' in this case should be more than 1. Calculating the Euler's totient function from a negative integer is impossible. The principle, in this case, is that for ϕ(n), the multiplicators called m and n should be greater than 1. Hence, denoted by $1<m<n$ and gcd (m, n) = 1. Sign ϕ is the sign used to denote the totient function.

Properties of Euler’s Totient Function

There are some different properties. Some of the properties of Euler’s totient function are as under:

- Φ is the symbol used to denote the function.
- The function deals with the prime numbers theory.
- The function is applicable only in the case of positive integers.
- For ϕ (n), one can find two multiplicative prime numbers to calculate the function.
- The function is a mathematical function and useful in many ways.
- If integer ‘n’ is a prime number, then gcd (m, n) = 1.
- The function works on the formula 1< m< n, where m and n are the prime and multiplicative numbers.
- In general, the equation is:

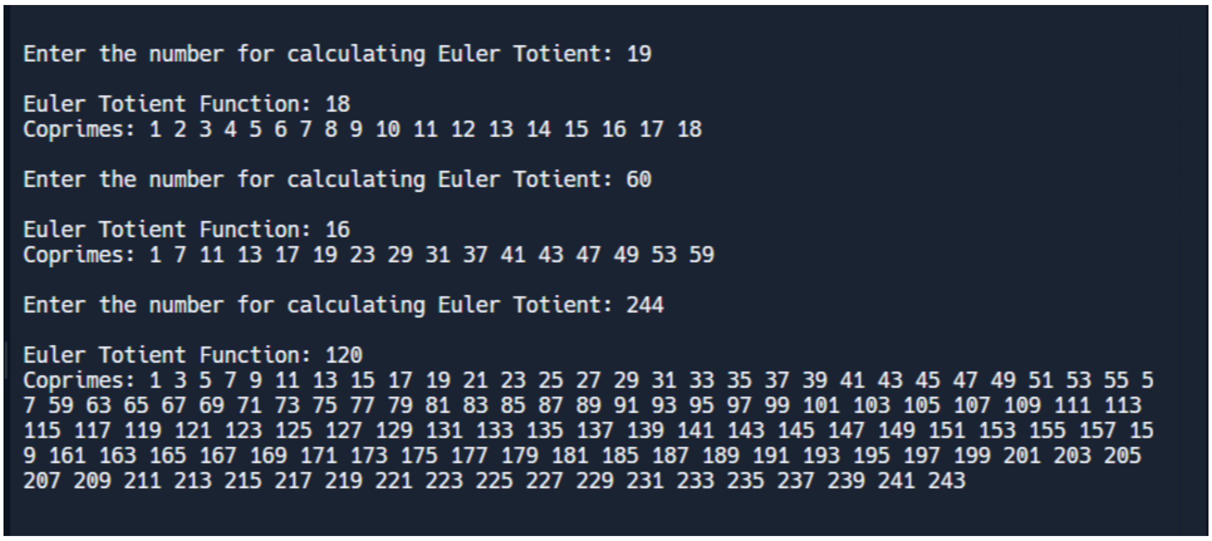
$\Phi (mn) = \phi (m) * \phi (n) (1- 1 / m) (1 – 1/ n)$

Euler’s totient function is useful in many ways. One may use it in the RSA encryption system for security purposes. The function deals with the prime number theory, and it is useful in the calculation of large calculations also. One may also use this function in algebraic calculations and elementary numbers. The symbol used to denote the function is ϕ, also called a phi function. The function consists of more theoretical use rather than practical use. The practical use of the function is limited. One can understand the function through various practical examples rather than theoretical explanations. There are various rules for calculating the Euler’s totient function, and different rules apply to different numbers. The function was first introduced in 1763. Due to some issues, it got recognition in 1784, and they modified the name in 1879. The function is universal and can be applied everywhere.

Code:

```
#include <bits/stdc++.h>
using namespace std;

unordered_map<int, int> primeFactors(int n) {
  unordered_map<int, int> factors;
  for (int i = 2; i * i <= n; ++i) {
    while (n % i == 0) {
      factors[i] = factors[i] + 1;
      n /= i;
    }
  }
  if (n > 1) {
    factors[n] = factors[n] + 1;
  }
  return factors;
}

int gcd(int a, int b) { return b == 0 ? a : gcd(b, a % b); }

void printCoprimes(int n) {
  for (int i = 1; i < n; ++i) {
    if (gcd(i, n) == 1) {
      cout << i << " ";
    }
  }
  cout << endl;
}

void eulerTotient(int n) {
  unordered_map<int, int> factors = primeFactors(n);
  if (factors.size() == 1 && factors[n] == 1) {
    cout << "Euler Totient Function: " << (n - 1) << endl;

    cout << "Coprimes: ";
    printCoprimes(n);
  } else {
    int size = factors.size();
    if (size == 2 && factors.begin()->second == 1 &&
        next(factors.begin())->second == 1) {
      int p = factors.begin()->first;
      int q = next(factors.begin())->first;
      int totient = (p - 1) * (q - 1);
      cout << "Euler Totient Function: " << totient << endl;
      cout << "Coprimes: ";
      printCoprimes(n);
    } else {
      int totient = 1;
      for (const auto &entry : factors) {
        int p = entry.first;
        int k = entry.second;
        totient *= (pow(p, k) - pow(p, k - 1));
      }
      cout << "Euler Totient Function: " << totient << endl;
      cout << "Coprimes: ";
      printCoprimes(n);
    }
  }
}

int main() {
  for (int i = 1; i <= 3; i++) {
    cout << "\nEnter the number for calculating Euler Totient: ";
    int n;
    cin >> n;
    cout << endl;
    eulerTotient(n);
  }
  return 0;
}
```

Output:



```
Enter the number for calculating Euler Totient: 19

Euler Totient Function: 18
Coprimes: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

Enter the number for calculating Euler Totient: 60

Euler Totient Function: 16
Coprimes: 1 7 11 13 17 19 23 29 31 37 41 43 47 49 53 59

Enter the number for calculating Euler Totient: 244

Euler Totient Function: 120
Coprimes: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 5
7 59 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113
115 117 119 121 123 125 127 129 131 133 135 137 139 141 143 145 147 149 151 153 155 157 15
9 161 163 165 167 169 171 173 175 177 179 181 185 187 189 191 193 195 197 199 201 203 205
207 209 211 213 215 217 219 221 223 225 227 229 231 233 235 237 239 241 243
```