**Totient Function**

* **TUTORIAL**
* [**PROBLEMS**](https://www.hackerearth.com/practice/math/number-theory/totient-function/practice-problems/)

**What is Euler's Totient Function?**

Number theory is one of the most important topics in the field of Math and can be used to solve a variety of problems. Many times one might have come across problems that relate to the prime factorization of a number, to the divisors of a number, to the multiples of a number and so on.

Euler's Totient function is a function that is related to getting the number of numbers that are coprime to a certain number XX that are less than or equal to it. In short , for a certain number XX we need to find the count of all numbers YY where gcd(X,Y)=1gcd(X,Y)=1 and 1≤Y≤X1≤Y≤X.

A naive method to do so would be to **Brute-Force** the answer by checking the gcd of XX and every number less than or equal to XX and then incrementing the count whenever a GCDGCD of 11 is obtained. However, this can be done in a much faster way using Euler's Totient Function.

According to Euler's product formula, the value of the Totient function is below the product over all prime factors of a number. This formula simply states that the value of the Totient function is the product after multiplying the number NN by the product of (1−(1/p))(1−(1/p)) for each prime factor of NN.

So,  
ϕ(n)=n∏p prime p|n(1−1p)ϕ(n)=n∏p prime p|n(1−1p)

**Algorithm steps:**

* Generate a list of primes.
* While dealing with a certain NN, check and store all the primes that perfectly divide NN.
* Now, it is just needed to use these primes and the above formula to get the result.

**Implementation:**

set<> primes;

static void mark(int num,int max,int[] arr)

{

int i=2,elem;

while((elem=(num\*i))<=max)

{

arr[elem-1]=1;

i++;

}

}

GeneratePrimes()

{

int arr[max\_prime];

for(int i=1;i<arr.length;i++)

{

if(arr[i]==0)

{

list.add(i+1);

mark(i+1,arr.length-1,arr);

}

}

}

main()

{

GeneratePrimes();

int N=nextInt();

int ans=N;

for(int k:set)

{

if(N%k==0)

{

ans\*=(1-1/k);

}

}

print(ans);

}

There are a few subtle observations that one can make about Euler's Totient Function.

* The sum of all values of Totient Function of all divisors of NN is equal to NN.
* The value of Totient function for a certain prime PP will always be P−1P−1 as the number PP will always have a GCDGCD of 11 with all numbers less than or equal to it except itself.
* For 2 number A and B, if GCD(A,B)==1GCD(A,B)==1 then Totient(A)×Totient(B)Totient(A)×Totient(B) = Totient(A⋅B)Totient(A⋅B).

*Contributed by: [Anand Jaisingh](https://www.hackerearth.com/@anand.jaisingh" \t "blank)*

**Did you find this tutorial helpful?**

 YES

 NO

**TEST YOUR UNDERSTANDING**

**Euler Totient Function**

In number theory, the totient ΦΦ of a positive integer NN is defined as the number of positive integers less than or equal to NN that are co-prime to NN.

Given an integer NN. Compute the value of the totient ΦΦ.

**Input:**  
First and the only line of input contains single integer NN.

**Output:**   
Print the Φ(N)Φ(N) in a single line.

**Constraints:**  
1≤N≤10000001≤N≤1000000

**SAMPLE INPUT**

5

**SAMPLE OUTPUT**

4

<https://www.hackerearth.com/practice/math/number-theory/totient-function/tutorial/>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static int gcd(int a, int b)

{

if (a == 0)

return b;

return gcd(b % a, a);

}

static void Main(string[] args)

{

int n = int.Parse(Console.ReadLine());

int ans = 0;

for (int i = 1; i <= n; i++)

{

if (gcd(i, n) == 1)

{

ans++;

}

}

Console.WriteLine(ans);

Console.ReadLine();

}

}

}