Euler function

Group

A ***group*** G = <S, º > is a pair, where

- S is a finite or infinite set of elements;
- º is a binary operation (called the group operation) that together satisfy the four fundamental properties of *closure*, *associativity*, *the identity property*, and *the inverse property*.

1. **Closure**: If a and b are two elements in G, then $a \circ b$ is also in G.
2. **Associativity**: The defined operation º is associative, i.e., for all $a, b, c \in G$ we have: $(a \circ b) \circ c = a \circ (b \circ c)$.
3. **Identity**: There is an identity element I (a.k.a. 1, E or e) such that $I \circ a = a \circ I = a$ for every element $a \in G$.
4. **Inverse**: There must be an *inverse* (a.k.a. *reciprocal*) of each element. Therefore, for each element a of G, the set contains an element $b = a^{-1}$ such that

$a \circ a^{-1} = a^{-1} \circ a = I$

Let N be a set of positive integers. Then:

- <N, +> is **not** a group, there is no *identity* element.
- <N ∪ {0}, +> is **not** a group, *identity* = 0, but there is no *inverse* element.

Let Z be a set of integers. Then:

- <Z, +> is a group, *identity* = 0, $3^{-1} = -3$, $-3^{-1} = 3$.
- <Z, *> is **not** a group, *identity* = 1, but there is no *inverse* element.

Let Q be a set of fractions. Then:

- < Q, *> is a group, *identity* = 1, $3^{-1} = 1/3$, $2/7^{-1} = 7/2$.

Let M be a set of matrices. Then:

- < M \ (0), *> is a group, *identity* = E, each matrix has an *inverse*. Matrix multiplication is *associative*, but not *commutative*.

Complete residue system

A subset S of the set of integers is called a ***complete residue system*** modulo n if

- no two elements of S are congruent modulo n;
- S contains n elements;

For example, a complete residue system modulo 5 is {3, 4, 5, 6, 7}, which is equivalent to {0, 1, 2, 3, 4}.

$Z_n = \{0, 1, 2, \ldots, n - 1\}$ is a complete residue system consisting of minimal nonnegative residues.

< Z_n, $+_{\text{mod } n}$> is a group. For example, $Z_5 = \{0, 1, 2, 3, 4\}$.

Closure: 3 + 4 = 2 because (3 + 4) mod 5 = 2.

Associativity: (3 + 4) + 2 = 3 + (4 + 2) = 4.

Identity: I = 0.

Inverse: $3^{-1} = 2$ because $3 + 2 = 0$ (mod 5), $4^{-1} = 1$.

Reduced residue system

A subset Z_n^* of the set of integers is called a ***reduced residue system*** modulo n if

- Each element in Z_n^* is no more than n;
- Each element in Z_n^* is coprime with n;

$< Z_n^*, *_{\text{mod } n} >$ is a group.

For example, $Z_{10}^* = \{1, 3, 7, 9\}$, $Z_{12}^* = \{1, 5, 7, 11\}$. Product of any numbers from the set modulo n belongs to the same set:

i / j	1	3	7	9
1	1	3	7	9
3	3	9	1	7
7	7	1	9	3
9	9	7	3	1

(i * j) mod 10

i / j	1	5	7	11
1	1	5	7	11
5	5	1	11	7
7	7	11	1	5
11	11	7	5	1

(i * j) mod 12

If p is prime, then $Z_p^* = \{1, 2, 3, \ldots, p - 1\}$. All positive integers less than p belong to Z_p^* because they are coprime with p. For example, $Z_7^* = \{1, 2, 3, 4, 5, 6\}$.

The cardinality of the set Z_n^* equals to **Euler function** $\varphi(n)$:

$$| Z_n^* | = \varphi(n)$$

Below the **properties** of the Euler function are given:

- if p is prime, then $\varphi(p) = p - 1$ and $\varphi(p^a) = p^a * (1 - 1/p)$ for any a.
- if m and n are coprime, then $\varphi(m * n) = \varphi(m) * \varphi(n)$.
- if $n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$, the Euler function is calculated using the next formula:

$$\varphi(n) = n * (1 - 1/p_1) * (1 - 1/p_2) * \ldots * (1 - 1/p_k)$$

For example,

$$\varphi(20) = \varphi(2^2 * 5) = 20 * (1 - 1/2) * (1 - 1/5) = 20 * 1/2 * 4/5 = 8,$$
$$\varphi(12) = \varphi(2^2 * 3) = 12 * (1 - 1/2) * (1 - 1/3) = 12 * 1/2 * 2/3 = 4,$$
$$\varphi(10) = \varphi(2 * 5) = 10 * (1 - 1/2) * (1 - 1/5) = 10 * 1/2 * 4/5 = 4$$

Function ***euler*** finds the value of $\varphi(n)$.

```
int euler(int n)
{
```

Initialize *result* with n.

```
  int i, result = n;
```

Iterate over all prime divisors *i* of *n*.

```
  for(i = 2; i * i <= n; i++)
  {
```

If *i* is a prime divisor of *n*, calculate

$$result = result * (1 - 1 / i) = result - result / i$$

```
    if (n % i == 0) result -= result / i;
```

Remove all divisors *i* from *n*.

```
    while (n % i == 0) n /= i;
  }
```

If $n > 1$, then initially *n* contained a prime divisor greater than $\sqrt{n}$. For example, number 10 = 2 * 5 contains prime divisor 5, greater than $\sqrt{10}$. Take this divisor into account when calculating the result.

```
  if (n > 1) result -= result / n;
  return result;
}
```

E-OLYMP [339. Again irreducible] The fraction *m* / *n* is called regular irreducible, if $0 < m < n$ and GCD(*m*, *n*) = 1. Find the number of regular irreducible fractions with denominator *n*.

► The number of regular irreducible fractions with denominator *n* equals to Euler's function φ(*n*). For *n* = 12 we have the following regular irreducible fractions:

$$\frac{1}{12}, \frac{5}{12}, \frac{7}{12}, \frac{11}{12}$$

Consider the set of all regular fractions with the denominator 12:

$$\frac{0}{12}, \frac{1}{12}, \frac{2}{12}, \frac{3}{12}, \frac{4}{12}, \frac{5}{12}, \frac{6}{12}, \frac{7}{12}, \frac{8}{12}, \frac{9}{12}, \frac{10}{12}, \frac{11}{12}$$

After simplifying, they will look like:

$$\frac{0}{1}, \frac{1}{12}, \frac{1}{6}, \frac{1}{4}, \frac{1}{3}, \frac{5}{12}, \frac{1}{2}, \frac{7}{12}, \frac{2}{3}, \frac{3}{4}, \frac{5}{6}, \frac{11}{12}$$

Let's group the fractions by their denominators:

$$\frac{0}{1}, \quad \frac{1}{2}, \quad \frac{1}{3}, \frac{2}{3}, \quad \frac{1}{4}, \frac{3}{4}, \quad \frac{1}{6}, \frac{5}{6}, \quad \frac{1}{12}, \frac{5}{12}, \frac{7}{12}, \frac{11}{12}$$

Among the denominators, every divisor *d* of 12 occurs along with all φ(*d*) of its numerators. All denominators are divisors of 12. Hence

$$\varphi(1) + \varphi(2) + \varphi(3) + \varphi(4) + \varphi(6) + \varphi(12) = 12$$

If we start with a series of irreducible fractions 0 / *m*, 1 / *m*, …, (*m* – 1) / *m*, we can get the equality:

$$n = \sum_{d|n} \varphi(d)$$

E-OLYMP 1563. Send a table Jimmy have to calculate a function $f(x, y)$ where x and y are both integers in the range $[1, n]$. When he knows $f(x, y)$, he can easily derive $f(k*x, k*y)$, where k is any integer from it by applying some simple calculations involving $f(x, y)$ and k.

Note that the function f is not symmetric, so $f(x, y)$ can not be derived from $f(y, x)$.

For example if $n = 4$, he only needs to know the answers for 11 out of the 16 possible input value combinations:

f(1,1)	f(1,2)	f(1,3)	f(1,4)
f(2,1)		f(2,3)	
f(3,1)	f(3,2)		f(3,4)
f(4,1)		f(4,3)	

The other 5 can be derived from them:

- f(2, 2), f(3, 3) and f(4, 4) from f(1, 1);
- f(2, 4) from f(1, 2);
- f(4, 2) from f(2, 1);

For the given value of n find the minimum number of function values Jimmy needs to know to compute all n^2 values $f(x, y)$.

► Let res(i) be the minimum required number of known values of $f(x, y)$, where $x, y \in \{1, \ldots, i\}$. Obviously, res(1) = 1, since for $n = 1$ it is enough to know $f(1, 1)$.

Let the value of res(i) is known. For $n = i + 1$ we must find the values

				f(1,i+1)
				f(2,i+1)
				...
				f(i,i+1)
f(i+1,1)	f(i+1,2)	...	f(i+1,i)	f(i+1,i+1)

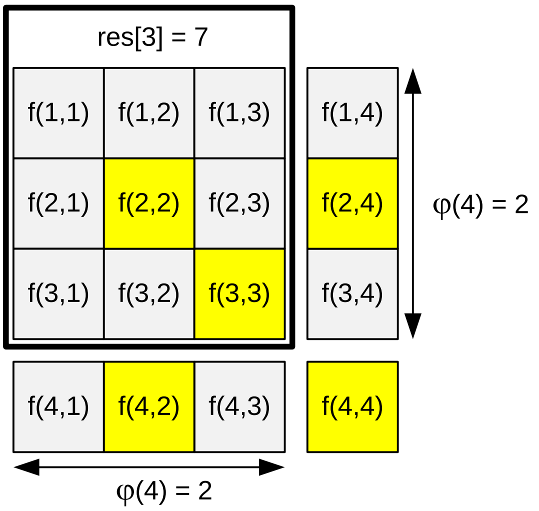
The values $f(j, i + 1)$ and $f(i + 1, j)$, $j \in \{1, \ldots, i + 1\}$ can be derived from the known values if GCD(j, i + 1) > 1, that is, if the numbers j and $i + 1$ are not coprime. Therefore, it is necessary to know all such $f(j, i + 1)$ and $f(i + 1, j)$, for which j and $i + 1$ are coprime. The number of such values is $2 * \varphi(i + 1)$, where φ is Euler's function. Thus

$$\text{res}(1) = 1,$$
$$\text{res}(i + 1) = \text{res}(i) + 2 * \varphi(i + 1),\ i > 1$$

Let's find the values of res(i) for some values of i:

$$\text{res}(1) = 1,$$
$$\text{res}(2) = \text{res}(1) + 2 * \varphi(2) = 1 + 2 * 1 = 3,$$

$$\text{res}(3) = \text{res}(2) + 2 * \varphi(3) = 3 + 2 * 2 = 7,$$



$$\text{res}(4) = \text{res}(3) + 2 * \varphi(4) = 7 + 2 * 2 = 11$$

Euler's theorem. If a and n are coprime, then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

$$|\, Z_n^* \,| = \varphi(n)$$

Proof. Let $Z_n^* = \{ r_1, ..., r_k\}$, where $k = \varphi(n)$. Then if we take any $a \in Z_n^*$ and find all possible products $a * r_i$, we get a set $\{ r_1', ..., r_k'\}$ that is just a permutation of $\{ r_1, ..., r_k\}$. Consider the system of congruence equations:

$$ar_1 \equiv r_1' \pmod{n},$$
$$ar_2 \equiv r_2' \pmod{n},$$
$$\ldots ,$$
$$ar_k \equiv r_k' \pmod{n}$$

Multiply the equations:

$$a^k * r_1 * \ldots * r_k \equiv r_1' * \ldots * r_k' \pmod{n}$$

Since the products $r_1 * \ldots * r_k$ and $r_1' * \ldots * r_k'$ are equal and coprime modulo n, we'll divide the equality by this product. We get

$$a^k \equiv 1 \pmod{n}$$

Since $k = \varphi(n)$, we have

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Fermat's theorem (a special case of Euler's theorem).
If p is prime, $a \in Z_p^*$, then

$$a^{p-1} \equiv 1 \pmod{p}$$

Corollary. If we multiply both sides of $a^{p-1} \equiv 1 \pmod{p}$ by a, we obtain

$$a^p \equiv a \pmod{p}$$

Corollary. $a^b (\text{mod}\, c) = a^{b'} (\text{mod}\, c)$, where $b' = b \bmod \varphi(c)$.
Proof. Let $b = k\varphi(c) + b'$.

Then $a^b \pmod c = a^{k\varphi(c)+b'} \pmod c = \left(a^{\varphi(c)}\right)^k \cdot a^{b'} \pmod c = a^{b'} \pmod c$.

Example. Find the value of 2^{100} mod 17.
Since φ(17) = 16, 2^{100} mod 17 = $2^{100 \bmod 16}$ mod 17 = 2^4 mod 17 = 16.

Find the value of 2^{1000} mod 100. Since

$$\varphi(100) = \varphi(2^2 * 5^2) = 100 * (1 – 1/2) * (1 – 1/5) = 100 * 1/2 * 4/5 = 40,$$

2^{1000} mod 100 = $2^{100 \bmod 40}$ mod 100 = 2^{20} mod 100 = 1048576 mod 100 = 76.

Example. Let's find an inverse for each element from Z_{10}^* = {1, 3, 7, 9}. From the Euler theorem we have $a^{\varphi(10)} \equiv 1 \pmod{10}$ or $a^4 \equiv 1 \pmod{10}$, $a * a^3 \equiv 1 \pmod{10}$, so

$$a^{-1} = a^3 \pmod{10}$$

	a	1	3	7	9	
	a³	1	27	343	729	
a³ mod 10		1	7	3	9	a⁻¹

So $1^{-1} = 1$, $3^{-1} = 7$, $7^{-1} = 3$, $9^{-1} = 9$.

E-OLYMP 5213. Inverse Prime number n is given. The **inverse** number to i ($1 \le i < n$) is such number j that $i * j = 1 \pmod n$. Its possible to prove that for each i exists only one inverse. For all possible values of i find the inverse numbers.

► Since the number n is prime, then by Fermat's theorem i^{n-1} mod n = 1 for every $1 \le i < n$. This equality can be rewritten in the form $(i * i^{n-2})$ mod n = 1, whence the inverse of i equals to $j = i^{n-2}$ mod n.

Let n = 5. Consider the table:

i	1	2	3	4
i³ mod 5	1³ mod 5 1 mod 5 1	2³ mod 5 8 mod 5 3	3³ mod 5 27 mod 5 2	4³ mod 5 64 mod 5 4

E-OLYMP 9606. Modular division Three positive integers a, b and n are given. Find the value of a / b mod n. You must fund such x that $b * x = a$ mod n.

► Since number n is prime, then by Fermat's theorem b^{n-1} mod n = 1 for every $1 \le b < n$. This equality can be rewritten in the form $(b * b^{n-2})$ mod n = 1, whence the inverse of b equals to $y = b^{n-2}$ mod n.

Hence a / b mod $n = a * b^{-1}$ mod $n = a * y$ mod n.

Consider the sample: compute 4 / 8 mod 13. To do this, solve the equation $8 * x = 4$ mod 13, wherefrom $x = (4 * 8^{-1})$ mod 13.

Number 13 is prime, Fermat's theorem implies that 8^{12} mod 13 = 1 or $(8 * 8^{11})$ mod 13 = 1. Therefore 8^{-1} mod 13 = 8^{11} mod 13 = 5.

Compute the answer: $x = (4 * 8^{-1})$ mod 13 = (4 * 5) mod 13 = 20 mod 13 = 7.

E-OLYMP [9627. a^b^c](#) Find the value of

$$a^{b^c} mod(10^9 + 7)$$

► By Fermat's little theorem $a^{p-1} = 1 \pmod p$, where p is prime. The number $p = 10^9 + 7$ is prime. Hence, for example, it follows that $a^{(p-1)*l} = 1 \pmod p$ for any number l.

To evaluate the expression a^b^c first find $k = b$^c, then calculate a^k. However, the number b^c is large, we represent it in the form b^$c = (p-1) * l + s$ for some l and $s < p-1$. Then

$$a\text{^}(b\text{^}c) \bmod p = a^{(p-1)*l+s} \bmod p = (a^{(p-1)*l} * a^s) \bmod p = a^s \bmod p$$

It's obvious that $s = b$^$c \bmod (p-1)$. Hence

$$a\text{^}(b\text{^}c) \bmod p = a\text{^}(b\text{^}c \bmod (p-1)) \bmod p$$

Let's calculate the value of 3^2^3 mod 7. Module 7 is chosen to be prime. The value of expression is

$$3\text{^}(2\text{^}3) \bmod 7 = 3^8 \bmod 7 = 6561 \bmod 7 = (937 * 7 + 2) \bmod 7 = 2$$

Fermat's theorem implies that $3^6 \bmod 7 = 1$. Therefore, for any positive integer k

$$(3^6 \bmod 7)^k = 3^{6k} \bmod 7 = 1$$

Since $2\text{^}3 = 2^3 = 8$, then $3^8 \bmod 7 = 3^{6*1+2} \bmod 7 = 3^2 \bmod 7 = 9 \bmod 7 = 2$

The original expression can also be evaluated as

$$3\text{^}(2\text{^}3) \bmod 7 = 3^8 \bmod 7 = 3^{8 \bmod 6} \bmod 7 = 3^2 \bmod 7 = 9 \bmod 7 = 2$$

E-OLYMP [1083. Sequence](#) In a sequence of numbers $a_1, a_2, a_3, \ldots$ the first term is given, and the other terms are calculated using the formula:

$$a_i = (a_{i-1} * a_{i-1}) \bmod 10000$$

Find the n-th term of the sequence.

► Let us express the first terms of the sequence in terms of a_1:

- $a_2 = a_1^2 \bmod 10000$,
- $a_3 = a_2^2 \bmod 10000 = a_1^4 \bmod 10000$,
- $a_4 = a_3^2 \bmod 10000 = a_2^4 \bmod 10000 = a_1^8 \bmod 10000$

The formula can be rewritten as $a_i = a_{i-1}^2 \bmod 10000$, whence it follows that to calculate a_n, the number a_1 should be raised to the power 2^{n-1}:

$$a_n = a_1^{2^{n-1}}$$

Considering that $a^b \bmod n = a^{b \bmod \varphi(n)} \bmod n$, to find the result res, the following calculations should be performed:

$$x = 2^{n-1} \bmod \varphi(10000) = 2^{n-1} \bmod 4000,$$

$$res = a_1^x \bmod 10000$$

E-OLYMP [7807. Happy sum](#) It is known that the number is happy, if its decimal notation contains only fours and sevens. For example, the numbers 4, 7, 47, 7777 and 4744474 are happy.

Let S be the set of happy numbers, no less than *a* and no more than *b*:

$$S = \{n : a \le n \le b, n \text{ is happy}\}$$

Calculate the remainder of dividing by 1234567891 the next sum:

$$\sum_{n \in S} n^n$$

► The modulus p = 1234567891 is primt. So $n^{p-1} = 1 \pmod p$. We have

$$n^n \pmod p = (n \bmod p)^{(p-1) + \ldots + (p-1) + (n \bmod (p-1))} \pmod p =$$

$$(n \bmod p)^{n \bmod (p-1)} \pmod p$$

For example $23^{23} \pmod 5 = (23 \bmod 5)^{4+4+4+4+4+3} \pmod 5 = 3^3 \pmod 5$, because $3^4 \pmod 5 = 1$.

Let modPow(a, n) = a^n mod p. Since $n \le 10^{18}$, then the arguments of modPow(n, n) wikk have the type *long long* and when multiplying we get overflow. From the above equality we have:

$$\text{modPow}(n, n) = \text{modPow}(n \bmod p, n \bmod (p-1))$$

Now we can pass *int* arguments to the function ***modPow***.

To generate happy numbers, it should be noted that if *n* is happy, then numbers 10**n* + 4 and 10**n* + 7 will be also happy.

Recursive generation of happy numbers.

```
void f(long long n)
{
```

As soon as the next generated number *n* becomes greater than *b*, we stop to generate the numbers.

```
  if (n > b) return;
```

Sum up the values n^n only for those happy numbers *n*, for which $a \le n \le b$.

```
  if (n >= a) res = (res + modPow(n % MOD, n % (MOD - 1))) % MOD;
```

In *n* is a happy number, then numbers 10**n* + 4 and 10**n* + 7 will be also happy.

```
  f(n * 10 + 4);
  f(n * 10 + 7);
}
```

Generate the happy numbers starting from 0. Calculate the required sum in the *res* variable.

```
f(0);
```

E-OLYMP [4742. Number of divisors](#) The integer *n* is given. Find the number of its divisors, excluding divisors *n* and 1.

► Let d(n) be the number of divisors of n. Obviously, d(1) = 1.

Let p be prime integer. Then p has two divisors: 1 and p. Hence d(p) = 2.

Let $n = p^k$ be the prime power. Then n has $k + 1$ divisors: 1, p, p^2, p^3, …, p^k. So $d(p^k) = k + 1$.

Let $n = p^k q^l$. Consider two sets:

$$P = \{1, p, p^2, p^3, \ldots, p^k\} \text{ and } Q = \{1, q, q^2, q^3, \ldots, q^l\}$$

Any divisor d of the number $p^k q^l$ can be represented in the form $x * y$, where $x \in P$, $y \in Q$. Divisor x from P can be chosen in $k + 1$ ways, divisor y from Q can be chosen in $l + 1$ ways. Hence the divisor $d = x * y$ can be constructed in $(k + 1) * (l + 1)$ ways.

Decompose the number n into prime factors: $n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$. The number of divisors of n is

$$d(n) = (a_1 + 1) * (a_2 + 1) * \ldots * (a_k + 1)$$

Factorize the number of $n = 18$:

$$18 = 2 * 3^2$$

Therefore

$$d(18) = (1 + 1) * (2 + 1) = 2 * 3 = 6$$

Subtracting two divisors (1 and 18), we get the answer: 4 divisors.

Function ***CountDivisors*** factorize the number n and calculates the number of its divisors $d(n)$. In the variable *res*, we count the number of divisors of the number n. In the *for* loop, when we meet the divisor i of n, in the variable c we calculate the degree with which i is included in the number n. That is, c is the maximum degree for which n is divisible by i^c.

```
int CountDivisors(int n)
{
  int c, i, res = 1;
  for(i = 2; i * i <= n; i++)
  {
    if (n % i == 0)
    {
      c = 0;
      while(n % i == 0)
      {
        n /= i;
        c++;
      }
      res *= (c + 1);
    }
  }
  if (n > 1) res *= 2;
  return res;
}
```

E-OLYMP **1564. Number theory** For the given positive integer n find the number of integers m, such that $1 \le m \le n$, $GCD(m, n) \ne 1$ and $GCD(m, n) \ne m$. GCD is an abbreviation for “greatest common divisor”.

► From the number n, we must subtract the number of coprime numbers with n, that equals to the Euler function $\varphi(n)$ (if m and n are coprime, then $GCD(m, n) = 1$), and the number of its divisors (if m is a divisor of n, then GCD(m, n) = m). In this case, the number 1 will be simultaneously coprime with n and a divisor of n. Therefore, 1 should be added to the resulting difference.

If $n = p_1^{k_1} p_2^{k_2} ... p_t^{k_t}$ is a factorization of n, it has d(n) = (k_1 + 1) * (k_2 + 1) * … * (k_t + 1) divisors.

Thus, the number of required values of m for the given n equals to

$$n - \varphi(n) - d(n) + 1$$

Let n = 10. We have φ(10) = 4 coprime numbers with 10: 1, 3, 7, 9.

Number 10 has d(10) = d(2 * 5) = 2 * 2 = 4 divisors: 1, 2, 5, 10.

The number of integers m, such that $1 \le m \le 10$, GCD(m, 10) ≠ 1 and GCD(m, 10) ≠ m is

$$10 - \varphi(10) - d(10) + 1 = 10 - 4 - 4 + 1 = 3$$

E-OLYMP [4107. Totient extreme] Given the value of n, you will have to find the value of H. The meaning of H is given in the following code:

```
H = 0;
for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j++) {
        H = H + totient(i) * totient(j);
    }
}
```

Totient or ***phi*** function, φ(n) is an arithmetic function that counts the number of positive integers less than or equal to n that are relatively prime to n. That is, if n is a positive integer, then φ(n) is the number of integers k in the range $1 \le k \le n$ for which GCD(n, k) = 1.

► Let us rewrite the sum H as follows:

φ(1) * φ(1) + φ(1) * φ(2) + … φ(1) * φ(n) +
φ(2) * φ(1) + φ(2) * φ(2) + … φ(2) * φ(n) +
. . .
φ(n) * φ(1) + φ(n) * φ(2) + … φ(n) * φ(n) =

φ(1) * (φ(1) + φ(2) + … φ(n)) +
φ(2) * (φ(1) + φ(2) + … φ(n)) +
. . .
φ(n) * (φ(1) + φ(2) + … φ(n)) =

$$= (\varphi(1) + \varphi(2) + \ldots \varphi(n))^2$$

Let's implement a sieve that will calculate all values of the Euler function from 1 to 10^4 and put them into the array fi. Let's fill in the array of partial sums sum[i] = φ(1) + φ(2) + … φ(i). Next, for each input value of n, print sum[n] * sum[n].

Consider the arrays with values of Euler function fi and the array of partial sums sum:

i	1	2	3	4	5	6	7	8	9	10
φ(*i*)	1	1	2	2	4	2	6	4	6	4
sum(*i*)	1	2	4	6	10	12	18	22	28	32

For $n = 10$ the answer is

$$(\varphi(1) + \varphi(2) + \ldots \varphi(10))^2 = \text{sum}[10]^2 = 32^2 = 1024$$

Function ***FillEuler*** filles the array fi[*i*] with values of Euler function: fi[*i*] = φ(*i*) ($1 \le i < \text{MAX}$).

```
void FillEuler(void)
{
  int i, j;
```

Initialize φ(*i*) = *i*.

```
  for (i = 0; i < MAX; i++) fi[i] = i;
  for (i = 2; i < MAX; i++)
    if (fi[i] == i)
```

Number *i* is prime. Iterate through all values of $j > i$ for which *i* is a prime divisor.

```
      for (j = i; j < MAX; j += i)
```

If *i* is a prime divisor of *j*, then $\varphi(j) = \varphi(j) * (1 - 1 / i) = \varphi(j) - \varphi(j) / i$.

```
        fi[j] -= fi[j] / i;
}
```

Consider an example. Initialize φ(*i*) = *i*:

i	1	2	3	4	5	6	7	8	9	10	11	12
φ(i)	1	2	3	4	5	6	7	8	9	10	11	12

Start the *for* loop from *i* = 2. fi[2] = 2, so 2 is prime.
Start *for j* loop, *j* = 2, 4, 6, 8, 10, 12, recalculate fi[*j*] = fi[*j*] * (1 – 1 / 2) = fi[*j*] / 2.

i	1	2	3	4	5	6	7	8	9	10	11	12
φ(i)	1	1	3	2	5	3	7	4	9	5	11	6

Next value of *i* = 3. fi[3] = 3, so 3 is prime.
Start *for j* loop, *j* = 3, 6, 9, 12, recalculate fi[*j*] = fi[*j*] * (1 – 1 / 3) = fi[*j*] * 2 / 3.

i	1	2	3	4	5	6	7	8	9	10	11	12
φ(i)	1	1	2	2	5	2	7	4	6	5	11	4

Next value of *i* for which fi[*i*] = *i*, is 5 (5 is prime).
Start *for j* loop, *j* = 5, 10, recalculate fi[*j*] = fi[*j*] * (1 – 1 / 5) = fi[*j*] * 4 / 5.

i	1	2	3	4	5	6	7	8	9	10	11	12
φ(i)	1	1	2	2	4	2	7	4	6	4	11	4

Next value of *i* for which fi[*i*] = *i*, is 7 (7 is prime).
Start *for j* loop, *j* = 7, recalculate fi[*j*] = fi[*j*] * (1 – 1 / 7) = fi[*j*] * 6 / 7.

i	1	2	3	4	5	6	7	8	9	10	11	12
φ(i)	1	1	2	2	4	2	6	4	6	4	11	4

Next value of *i* for which fi[*i*] = *i*, is 11 (11 is prime).
Start *for j* loop, *j* = 11, recalculate fi[*j*] = fi[*j*] * (1 – 1 / 11) = fi[*j*] * 10 / 11.

i	1	2	3	4	5	6	7	8	9	10	11	12
φ(i)	1	1	2	2	4	2	6	4	6	4	10	4

E-OLYMP 1128. Longge's problem Longge is good at mathematics and he likes to think about hard mathematical problems which will be solved by some graceful algorithms. Now a problem comes:

Given an integer n ($1 < n < 2^{31}$), you are to calculate $\sum \gcd(i, n)$ for all $1 \le i \le n$.

"Oh, I know, I know!" Longge shouts! But do you know? Please solve it.

► **Theorem.** If the function $f(n)$ is multiplicative, then the summation function $S_f(n) = \sum_{d|n} f(d)$ is also multiplicative.

Proof. Let $x, y \in N$, where x and y are coprime. Let $x_1, x_2, \ldots, x_k$ be all divisors of x. Let $y_1, y_2, \ldots, y_m$ be all divisors of y. Then $GCD(x_i, y_j) = 1$, and all possible products $x_i y_j$ give all divisors of xy. Then

$$S_f(x) * S_f(y) = \sum_{i=1}^{k} f(x_i) * \sum_{j=1}^{m} f(y_j) = \sum_{i,j} f(x_i) f(y_j) = \sum_{i,j} f(x_i y_j) = S_f(xy)$$

Corollary. Consider the function $f(n) = GCD(n, c)$, where c is a constant. If x and y are coprime, then $f(x * y) = GCD(x * y, c) = GCD(x, c) * GCD(y, c) = f(x) * f(y)$. Therefore the function $f(n) = GCD(n, c)$ is multiplicative.

Let $g(n) = \sum_{i=1}^{n} НОД(i, n)$. Then

$$g(p_1^{a1} p_2^{a2} ... p_k^{ak}) = g(p_1^{a1}) * g(p_2^{a2}) * \ldots * g(p_k^{ak})$$

Theorem. For any prime p and positive integer a holds the relation:

$$g(p^a) = (a + 1)p^a – ap^{a–1}$$

► For a = 1 we have:

$$g(p) = GCD(1, p) + GCD(2, p) + \ldots + GCD(p, p) = (p – 1) + p = 2p – 1$$

Similarly for a = 2:

g(p^2) =

GCD(1,p^2)+	GCD(2,p^2)+	. . .	GCD(p,p^2)+
GCD(p+1,p^2)+	GCD(p+2,p^2)+	. . .	GCD(2p,p^2)+
GCD(2p+1,p^2)+	GCD(2p+2,p^2)+	. . .	GCD(3p,p^2)+
. . .			
GCD((p-1)p+1,p^2)+	GCD((p-1)p+2,p^2)+	. . .	GCD(p^2,p^2)

=

$$= (1 + 1 + \ldots + 1 + p) +$$
$$(1 + 1 + \ldots + 1 + p) +$$
$$\ldots$$
$$(1 + 1 + \ldots + 1 + p^2) =$$

$$= (p – 1 + p) * (p – 1) + (p – 1 + p^2) =$$
$$(2p – 1) * (p – 1) + (p^2 + p – 1) =$$
$$2p^2 – 2p – p + 1 + (p^2 + p – 1) =$$
$$= 3p^2 – 2p$$

Lemma. If d is a divisor of n, then there are exactly $\varphi(n/d)$ numbers i such that GCD(i, n) = d.

► Obviously i must be divisible by d, let $i = dj$. Then

$$GCD(i, n) = GCD(dj, n) = d * GCD(j, n / d)$$

If the last expression is equal to d, then GCD(j, n / d) = 1. The number of such j that GCD(j, n / d) = 1 is $\varphi(n/d)$.

Example. The number of such i that GCD(i, 24) = 3 is $\varphi(8)$ = 4.

GCD(j, 8) = 1 for $j \in \{1, 3, 5, 7\}$, therefore GCD(i, 24) = 3 for $i \in \{3, 9, 15, 21\}$ (we have $i = 3j$).

Theorem.

$$g(n) = \sum_{i=1}^{n} GCD(i,n) = n\sum_{d|n} \frac{\varphi(d)}{d}$$

► According to the above lemma, the number of pairs (i, n) for which GCD(i, n) = e, is exactly $\varphi(n/e)$. Replacing n / e = d, we get:

$$g(n) = \sum_{e|n} e\varphi\left(\frac{n}{e}\right) = \sum_{d|n} \frac{n}{d}\varphi(d) = n\sum_{d|n}\frac{\varphi(d)}{d}$$

Example. Let n = 6.

i	1	2	3	4	5	6
GCD(i,6)	1	2	3	2	1	6

Then g(6) = $\sum_{i=1}^{6} GCD(i,6)$ =

= GCD(1, 6) + GCD(2, 6) + GCD(3, 6) + GCD(4, 6) + GCD(5, 6) + GCD(6, 6) =
= 1 + 2 + 3 + 2 + 1 + 6 = 15

In the same time g(6) = g(2) * g(3) =
(GCD(1, 2) + GCD(2, 2)) * (GCD(1, 3) + GCD(2, 3) + GCD(3, 3)) =
(1 + 2) * (1 + 1 + 3) = 3 * 5 = 15

Compute g(6) using the formula g(n) = $n\sum_{d|n}\frac{\varphi(d)}{d}$:

$$g(6) = 6\sum_{d|6}\frac{\varphi(d)}{d} = 6\cdot\left(\frac{\varphi(1)}{1}+\frac{\varphi(2)}{2}+\frac{\varphi(3)}{3}+\frac{\varphi(6)}{6}\right) =$$

$$= 6\varphi(1)+3\varphi(2)+2\varphi(3)+\varphi(6) = 6 + 3 + 4 + 2 = 15$$

Let's calculate g(6) based on the multiplicativity of the function f(x) = GCD(x, n):
g(6) = g(2) * g(3) = (2*2 – 1) * (2*3 – 1) = 3 * 5 = 15

Example. Let n = 12. Then g(12) = $\sum_{i=1}^{12} GCD(i,12)$ =

1 + 2 + 3 + 4 + 1 + 6 + 1 + 4 + 3 + 2 + 1 + 12 = 40

i	1	2	3	4	5	6	7	8	9	10	11	12
НОД(i,12)	1	2	3	4	1	6	1	4	3	2	1	12

In the same time g(12) = g(4) * g(3) =
(GCD(1, 4) + GCD(2, 4) + GCD(3, 4) + GCD(4, 4)) *
* (GCD(1, 3) + GCD(2, 3) + GCD(3, 3)) =
(1 + 2 + 1 + 4) * (1 + 1 + 3) = 8 * 5 = 40

Compute g(12) using the formula g(n) = $n\sum_{d|n}\frac{\varphi(d)}{d}$:

$$g(12) = 12\sum_{d|12}\frac{\varphi(d)}{d} = 12\cdot\left(\frac{\varphi(1)}{1}+\frac{\varphi(2)}{2}+\frac{\varphi(3)}{3}++\frac{\varphi(4)}{4}+\frac{\varphi(6)}{6}+\frac{\varphi(12)}{12}\right) =$$

$$= 12\varphi(1) + 6\varphi(2) + 4\varphi(3) + 3\varphi(4) + 2\varphi(6) + \varphi(12) =$$
$$= 12 + 6 + 8 + 6 + 4 + 4 = 40$$

The divisors of 12 are: 1, 2, 3, 4, 6, 12. The number of *i* such that GCD(*i*, 12) = *d* equals to $\varphi(12/d)$. For example GCD(*i*, 12) = 3 holds for $\varphi(12/3)$ = $\varphi(4)$ = 2 different *i*, namely for *i* = 3, 9.

Let's calculate g(12) based on the multiplicativity of the function f(*x*) = GCD(*x*, *n*):

$$g(12) = g(2^2) * g(3) = (3 * 2^2 – 2 * 2) * (2*3 – 1) = 8 * 5 = 40$$

Function ***euler*** computes the Euler function.

```
long long euler(long long n)
{
  long long i, result = n;
  for (i = 2; i * i <= n;i++)
  {
    if (n % i == 0) result -= result / i;
    while (n % i == 0) n /= i;
  }
  if (n > 1) result -= result / n;
  return result;
}
```

The main part of the program. Read value of *n*. Compute the value g(*n*) by the formula $\sum_{e|n} e\varphi\left(\frac{n}{e}\right)$. Search for all divisors of *n* among the numbers from 1 to $\lfloor\sqrt{n}\rfloor$. If *i* is a divisor of *n*, then *n* / *i* will be also the divisor of *n*. Therefore, for each found divisor $i \le \lfloor\sqrt{n}\rfloor$ we must add to result *res* the value $i\varphi\left(\frac{n}{i}\right) + \frac{n}{i}\varphi(i)$. If *n* is a full square, *i* = *sq* = $\lfloor\sqrt{n}\rfloor$, then $i\varphi\left(\frac{n}{i}\right) = \frac{n}{i}\varphi(i)$ and two identical terms will be added to the *res* sum. Therefore we'll subtract one of them from *res* during the initialization of the variable.

```
while(scanf("%lld",&n) == 1)
{
  sq = (long long)sqrt(1.0*n);
  res = (sq * sq == n) ? -sq * euler(sq) : 0;
  for(i = 1; i <= sq; i++)
    if(n % i == 0) res = res + i * euler(n/i) + (n / i) * euler(i);
  printf("%lld\n",res);
}
```

E-OLYMP **1129. GCD Extreme II** For a given number *n* calculate the value of G, where

$$G = \sum_{i=1}^{i<n} \sum_{j=i+1}^{j\le n} GCD(i, j)$$

Here GCD(*i*, *j*) means the greatest common divisor of integers *i* and *j*.

For those who have trouble understanding summation notation, the meaning of G is given in the following code:

```
G = 0;
for(i = 1; i < n;i++)
for(j = i + 1 ;j <= n; j++)
{
    G += GCD(i,j);
}
```

► Let $d[k] = \sum_{i=1}^{i<k} \sum_{j=i+1}^{j \le k} \text{GCD}(i, j)$.

For example $d[2] = \sum_{i=1}^{i<2} \sum_{j=i+1}^{j \le 2} \text{GCD}(i, j) = \sum_{j=2}^{j \le 2} \text{GCD}(1, j) = \text{GCD}(1, 2) = 1$.

You can see that

$$d[k] = \sum_{i=1}^{i<k} \sum_{j=i+1}^{j \le k} \text{GCD}(i, j) = \sum_{i=1}^{i<k-1} \sum_{j=i+1}^{j \le k-1} \text{GCD}(i, j) + \sum_{i=1}^{i<k} \text{GCD}(i, k) = d[k-1] + \sum_{i=1}^{i<k} \text{GCD}(i, k)$$

d[k-1] equals to the sum of GCD over all pairs (i, j), marked with grey

(1,2)					
(1,3)	(2,3)				
(1,4)	(2,4)	(3,4)			
. . .	. . .	. . .	. . .		
(1,k-1)	(2,k-1)	(3,k-1)	. . .	(k-2,k-1)	
(1,k)	(2,k)	(3,k)	. . .	(k-2,k)	(k-1,k)

d[k] equals to sum of GCD for all pairs (i, j)

$$d[k] = d[k-1] + \sum_{i=1}^{k-1} GCD(i,k)$$

It remains to show how to calculate the value of $\sum_{i=1}^{i<k} \text{GCD}(i, k)$ faster than usual summation.

Lema. Let *n* is divisible by *d* and GCD(*x*, *n*) = *d*. Then *x* = *dk* for some positive integer *k*. From the relation GCD(*dk*, *n*) = *d* it follows that $\text{GCD}\left(k, \frac{n}{d}\right) = 1$.

Theorem. Let $f(n) = \sum_{i=1}^{n} \text{GCD}(i, n)$. Then $f(n) = \sum_{d|n} d \cdot \varphi\left(\frac{n}{d}\right)$ for all divisors *d* of number *n*. $\varphi(n)$ indicates here the Euler function.

Proof. The number of such *i*, for which GCD(*i*, *n*) = 1, equals to $\varphi(n)$. The number of such *i* ($i \le n$), for which GCD(*i*, *n*) = *d* (*d* is a divisor of *n*, *i* = *dk*), equals to the number of such *k* ($k \le \frac{n}{d}$), for which $\text{GCD}\left(k, \frac{n}{d}\right) = 1$ or $\varphi\left(\frac{n}{d}\right)$. The value of GCD(*i*, *n*) can be only the divisors of *n*. To find the value *f*(*n*) it remains to sum the values $d \cdot \varphi\left(\frac{n}{d}\right)$ over all divisors *d* of *n*.

Example. Consider the direct calculation: $f(6) = \sum_{i=1}^{6} \text{GCD}(i,6) = \text{GCD}(1, 6) + \text{GCD}(2, 6) + \text{GCD}(3, 6) + \text{GCD}(4, 6) + \text{GCD}(5, 6) + \text{GCD}(6, 6) = 1 + 2 + 3 + 2 + 1 + 6 = 15$.

Consider the calculation using the formula: $f(6) = \sum_{d|6} d \cdot \varphi\left(\frac{6}{d}\right) =$

$$1 \cdot \varphi\left(\frac{6}{1}\right) + 2 \cdot \varphi\left(\frac{6}{2}\right) + 3 \cdot \varphi\left(\frac{6}{3}\right) + 6 \cdot \varphi\left(\frac{6}{6}\right) =$$

$$1 \cdot \varphi(6) + 2 \cdot \varphi(3) + 3 \cdot \varphi(2) + 6 \cdot \varphi(1) =$$

$$2 + 4 + 3 + 6 = 15$$

In the first and in the second case 15 is the sum of two units ($1 \cdot \varphi(6)$), two doules ($2 \cdot \varphi(3)$), one triple ($3 \cdot \varphi(2)$) and one sextuple ($6 \cdot \varphi(1)$).

Declare the arrays. fi[*i*] stores the value of the Euler function φ(*i*).

```
#define MAX 4000010
long long d[MAX], fi[MAX];
```

The function ***FillEuler*** fills the array *fi* so that fi[*i*] = φ(*i*), *i* < MAX.

```
void FillEuler(void)
{
```

Initially set the value of fi[*i*] equal to *i*.

```
  for(i = 1; i < MAX; i++) fi[i] = i;
```

Each even number *i* has a prime divisor *p* = 2. To speed up the function working time, process it separately. For each even number *i* set fi[*i*] = fi[*i*] * (1 – 1 / 2) = fi[*i*] / 2.

```
  for(i = 2; i < MAX; i+=2) fi[i] /= 2;
```

Enumerate all the possible odd divisors *i* = 3, 5, 7, … .

```
  for(i = 3; i < MAX; i+=2)
    if(fi[i] == i)
```

If fi[*i*] = *i*, then the number *i* is prime. The number *i* is a prime divisor for any *j*, represented in the form *k* * *i* for any positive integer *k*.

```
      for(j = i; j < MAX; j += i)
```

If *i* is a prime divisor of *j*, then set fi(*j*) = fi(*j*) * (1 – 1/*i*).

```
        fi[j] -= fi[j]/i;
}
```

Before calling the function f the values d[i] already contain φ(i). The body of the function f adds to d[j] the values so that when the function finishes its work, the value d[j] contains $\sum_{i=1}^{j-1} \text{GCD}(i, j)$ according to the formula given in the theorem.

```
void f(void)
{
 int i, SQRT_MAX = sqrt(1.0*MAX);
 for(i = 2; i <= SQRT_MAX; i++)
 {
   d[i*i] += i * fi[i];
```

The number i is a divisor of j. So we need to add to d[j] the value of $i \cdot \varphi\left(\frac{j}{i}\right)$. Since the number j has also a divisor j / i, add to d[j] the value of $\frac{j}{i} \cdot \varphi\left(\frac{j}{j/i}\right) = \frac{j}{i} \cdot \varphi(i)$. If $i^2 = j$, add to d[j] not two terms, but only one $i \cdot \varphi\left(\frac{j}{i}\right) = i \cdot \varphi(i)$.

```
  // for(j = i * i + i; j < MAX; j += i)
  //   d[j] += i * fi[j / i] + j / i * fi[i];
```

We can avoid integer division in implementation. To do this note, that since the value of the variable j is incremented each time by i, then the value j / i will be increase by one in a loop. Set initially $k = j / i = (i * i + i) / i = i + 1$ and then increase k by 1 in each iteration.

```
   for(j = i * i + i, k = i + 1; j < MAX; j += i, k++)
     d[j] += i * fi[k] + k * fi[i];

 }
```

Its sufficiently to continue the loop by i till $\sqrt{\text{MAX}}$, because if i is a divisor of j and $i > \sqrt{\text{MAX}}$, then considering the fact that $j / i < \sqrt{\text{MAX}}$ we can state that the divisor i of the number j was taken in account when we considered the divider j / i.

```
}
```

The main part of the program. Initialize the arrays. Let d[i] = φ(i).

```
memset(d,0,sizeof(d));
FillEuler();
memcpy(d,fi,sizeof(fi));
```

i	1	2	3	4	5	6	7	8	9	10
d[*i*]	0	1	2	2	4	2	6	4	6	4

```
f();
```

i	1	2	3	4	5	6	7	8	9	10
d[*i*]	0	1	2	4	4	9	6	12	12	17

```
for(i = 3; i < MAX; i++)
  d[i] += d[i-1];
```

i	1	2	3	4	5	6	7	8	9	10
d[*i*]	0	1	3	7	11	20	26	38	50	67

```
while(scanf("%lld",&n),n)
  printf("%lld\n",d[n]);
```

E-OLYMP 5141. LCM sum Given *n*, calculate the sum LCM(1, *n*) + LCM(2, *n*) + … + LCM(*n*, *n*), where LCM(*i*, *n*) denotes the Least Common Multiple of the integers *i* and *n*.

► Let S = $\sum_{i=1}^{n} LCM(i,n)$ = $\sum_{i=1}^{n-1} LCM(i,n)$ + LCM(*n*, *n*) = $\sum_{i=1}^{n-1} LCM(i,n)$ + *n*, wherefrom

S – *n* = LCM(1, *n*) + LCM(2, *n*) + . . . + LCM(*n* – 1, *n*)

Rearrange the terms in the right side in reverse order and write the equality in the form

S – *n* = LCM(*n* – 1, *n*) + . . . + LCM(2, *n*) + LCM(1, *n*)

Let's add two equalities:

2(S – *n*) = (LCM(1, *n*) + LCM(*n* – 1, *n*)) + . . . + (LCM(*n* – 1, *n*) + LCM(1, *n*))

Consider the expression in parentheses:

$$LCM(i, n) + LCM(n - i, n) = \frac{in}{GCD(i,n)} + \frac{(n-i)n}{GCD(n-i,n)}$$

Note that the denominators of the last two terms are equal: GCD(*i*, *n*) = GCD(*n* – *i*, *n*), hence

$$\frac{in}{GCD(i,n)} + \frac{(n-i)n}{GCD(n-i,n)} = \frac{in+(n-i)n}{GCD(i,n)} = \frac{n^2}{GCD(i,n)}$$

So

$$2(S - n) = \sum_{i=1}^{n-1} \frac{n^2}{GCD(i,n)} = n\sum_{i=1}^{n-1} \frac{n}{GCD(i,n)}$$

GCD(*i*, *n*) = *d* can take only the values of divisors of the number *n*, while the number of *i* for which the specified equality holds is φ (*n* / *d*). Hence

$$2(S - n) = n\sum_{i=1}^{n-1} \frac{n}{GCD(i,n)} = n\sum_{\substack{d|n \\ d\neq n}} \frac{n}{d}\cdot\varphi\left(\frac{n}{d}\right) = n\sum_{\substack{d|n \\ d\neq 1}} d\cdot\varphi(d) = n\left(\sum_{d|n} d\cdot\varphi(d) - 1\right)$$

The second equality is true because if *d* is a divisor of *n*, then *n* / *d* is also a divisor of *n*. Moreover, if *d* ≠ *n*, then *n* / *d* ≠ 1. The last equality is valid, since the summand 1 * φ (1) = 1 is included in the sum. It remains to extract the value S from the equation:

$$2(S - n) = n\left(\sum_{d|n} d\cdot\varphi(d) - 1\right),$$

$$2S - 2n = n\sum_{d|n} d \cdot \varphi(d) - n,$$

$$S = \frac{n}{2}\left(\sum_{d|n} d \cdot \varphi(d) + 1\right)$$