//Euler’s Toitent

#define MAX 1000

/\*

\* Euler’s Totient function Φ(n) for an input n is count of numbers in {1, 2, 3, …, n}

\* that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1.

\* Phi(4) : GCD(1, 4) = 1, GCD(3, 4)

\* so, Phi(4) = 2

\*/

// Computes phi of n

int Phi(int n) {

int result = n; // Initialize result as n

// Consider all prime factors of n and subtract their

// multiples from result

for (int p=2; p\*p<=n; ++p) {

if (n % p == 0) { //p is a prime factor of n

while (n % p == 0) //eleminate all p factors from n

n /= p;

result -= result / p;

}

}

if (n > 1) //if n is still greater than 1, then it is also a prime

result -= result / n;

return result;

}

// Computes phi or Euler Phi 1 to n

long long phi[MAX];

void computeTotient(int n) {

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

phi[i] = i;

for (int p=2; p<=n; p++) { // Computation

if (phi[p] == p) { //if phi is not computed

phi[p] = p-1; //p is prime and phi(prime) = prime-1;

for (int i = 2\*p; i<=n; i += p) { // Sieve like implementation

// Add contribution of p to its multiple i by multiplying with (1 - 1/p)

phi[i] = (phi[i]/p) \* (p-1);

}

}

}

}