Chinese Remainder Theorem

We are given two arrays num[0..k-1] and rem[0..k-1]. In num[0..k-1], every pair is coprime (gcd for every pair is 1).

```
// A C++ program to demonstrate
// working of Chinise remainder
// Theorem
#include <bits/stdc++.h>
using namespace std;
// Returns modulo inverse of a
// with respect to m using
// extended Euclid Algorithm.
// Refer below post for details:
// https://www.geeksforgeeks.org/
// multiplicative-inverse-under-modulo-m/
int inv(int a, int m)
{
	int m0 = m, t, q;
	int x0 = 0, x1 = 1;
	if (m == 1)
		return 0;
	// Apply extended Euclid Algorithm
	while (a > 1) {
		// q is quotient
		q = a / m;
		t = m;
		// m is remainder now, process same as
		// euclid's algo
		m = a % m, a = t;
		t = x0;
		x0 = x1 - q * x0;
		x1 = t;
	}
	// Make x1 positive
	if (x1 < 0)
		x1 += m0;
	return x1;
```

```
}
// k is size of num[] and rem[]. Returns the smallest
// number x such that:
// x % num[0] = rem[0],
// x % num[1] = rem[1],
// .................
// x % num[k-2] = rem[k-1]
// Assumption: Numbers in num[] are pairwise coprime
// (gcd for every pair is 1)
int findMinX(int num[], int rem[], int k)
{
	// Compute product of all numbers
	int prod = 1;
	for (int i = 0; i < k; i++)
		prod *= num[i];
	// Initialize result
	int result = 0;
	// Apply above formula
	for (int i = 0; i < k; i++) {
		int pp = prod / num[i];
		result += rem[i] * inv(pp, num[i]) * pp;
	}

	return result % prod;
}

// Driver method
int main(void)
{
	int num[] = { 3, 4, 5 };
	int rem[] = { 2, 3, 1 };
	int k = sizeof(num) / sizeof(num[0]);
	cout << "x is " << findMinX(num, rem, k);
	return 0;
}
```