模：

模(Modulus)是一個建立在除法之上的二元運算，考慮整數 A、B、C，我們聲稱：

$$A \bmod B = C \text{ 當且僅當 存在整數 } n \text{ 使得 } A=nB+C \text{ 且 } 0≦C<B$$

在模運算之上我們定義等價關係”同餘”：

$$A≡B \pmod C \text{ 當且僅當 } A \bmod C = B \bmod C$$

基本性質：

1. A≡B, B≡C → A≡C
2. A≡B → B≡A
3. A≡A
4. A≡B, C≡D → A+C≡B+D
5. A≡B, C≡D → A-C≡B-D
6. A≡B, C≡D → AC≡BD
7. nA≡nB (mod nm) → A≡B (mod m)
8. nA≡nB (mod m), n is relative prime with m → A≡B (mod m)
9. Consider P a polynomial, A≡B → P(A) ≡P(B)

定理：

费馬小定理：(Fermat’s Little Theorem)

$$a^p \equiv a \pmod p, \text{for all integers a and all primes p}$$

定理二：

$$\gcd(a,b) \times \mathrm{lcm}(a,b) = a \times b, \text{for all integers } a, b$$

中國剩餘定理：(Chinese Remainder Theorem)

Consider Sequence $<A_n>$, $<m_n>$,$<M_n>$ where $M_k = (\prod_{i=1}^{n} m_i)/m_k$.
Assume that $x \equiv A_k \ (mod\ m_k)$, for all k in [1,n]:
Then x is a solution if and only if the following equivalence holds.

$x \equiv \sum_{k=1}^{n} A_k M_k t_k \ (mod\ \prod_{k=1}^{n} m_k)$, where $t_k M_k \equiv 1\ (\text{mod}\ m_k)$ for all k

輾轉相除法：(Euclidean Algorithm)

```
template<typename type>
type GCD(type left, type right){
	if(!left && !right) throw logic_error("Return value does not exist.\n");
	left = abs(left);
	right = abs(right);
	return left?GCD(right%left,left):right;
}
```

最小公倍數：透過定理二求出

```
template<typename type>
type LCM(type left, type right){
	if(!left && !right) return 0;
	return abs(left*right/GCD(left,right));
}
```

模逆元：

對所有整數 A，我們稱 B 為 A 在模 m 下之模逆元，當且僅當 AB≡1 (mod m)，

此地，A 在模 m 下之模逆元存在當且僅當 A、m 互質。

模逆元的同餘性：

考慮$A \times r \equiv 1 \pmod{M}$ ，則所有 A 在模 M 下的模逆元構成如下集合：

$$\{r+nM \mid n \in Z\}, \text{ where Z stands for the set of all integers}$$

考慮整數 A、B 以及方程式 Ax+By=GCD(A,B)，其解空間如下：

$$\{(x, y) | x = \alpha + A \times n, y = \beta - B \times n\}, \text{where } (\alpha, \beta) \textit{ is of one solution}$$

並且，我們可以藉由擴展歐基里德演算法算出其中一組解。

擴展歐基里德算法：(Extended Euclidean Algorithm)

```
template<typename type>
pair<type,type> extGCD(type left, type right){
	if(!left && !right) throw logic_error("Return value does not exist.\n");
	pair<type,type> lCor(1,0), rCor(0,1);
	if(left<0) lCor.first = -1;
	if(right<0) rCor.second = -1;
	left = abs(left);
	right = abs(right);
	while(left){
		swap(lCor,rCor);
		swap(left,right);
		lCor.first -= left/right*rCor.first;
		lCor.second -= left/right*rCor.second;
		left %= right;
	}
	return rCor;
}
```

最小正模逆元之求取：

```
template<typename type>
type modulusInverse(type input, type mod){
	type output = extGCD(input,mod).first;
	if(output<0) output = output%mod+mod;
	return output;
}
```