Week 22: Math & Number Theory Advanced Topics

Topics: - Modular Arithmetic & Modular Inverse - Chinese Remainder Theorem (CRT) - Extended Euclidean Algorithm - Fast Exponentiation & Fermat's Little Theorem - Primes, Sieve of Eratosthenes, Segmented Sieve - Number of Divisors, Sum of Divisors, Euler's Totient Function

Weekly Tips: - Modular arithmetic is crucial for large number computations. - Modular inverse can be computed using Fermat's theorem (for prime mod) or Extended Euclidean. - CRT helps solve simultaneous modular congruences. - Precompute primes using sieve for fast factorization and totients. - Practice problems involving divisibility, prime counting, and modular constraints.

Problem 1: Modular Inverse & Fast Exponentiation Link: CSES Modular Inverse **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```
#include <bits/stdc++.h>
using namespace std;
const int MOD=1e9+7;
long long modpow(long long a,long long b){
    long long res=1;
    while(b){
        if(b&1) res=res*a%MOD;
        a=a*a%MOD;
        b>>=1;
    }
    return res;
}
long long modinv(long long a){ return modpow(a,MOD-2); } // Fermat's Little
Theorem
int main(){
    long long a; cin>>a;
    cout<<modinv(a)<<endl;
}
```

Explanation Comments: - Fast exponentiation computes a^b mod MOD efficiently in O(log b). - Modular inverse uses Fermat's theorem for prime modulus. - Essential for combinatorial calculations modulo prime numbers.

Problem 2: Chinese Remainder Theorem Link: CP-Algorithms CRT **Difficulty:** Advanced

C++ Solution with Explanation Comments:

```
#include <bits/stdc++.h>
using namespace std;
```

```
long long ext_gcd(long long a,long long b,long long &x,long long &y){
    if(b==0){ x=1;y=0;return a; }
    long long x1,y1;
    long long g=ext_gcd(b,a%b,x1,y1);
    x=y1; y=x1-(a/b)*y1;
    return g;
}
pair<long long,long long> crt(long long a1,long long m1,long long a2,long long
m2){
    long long x,y;
    long long g=ext_gcd(m1,m2,x,y);
    if((a2-a1)%g!=0) return {0,-1};
    long long lcm=m1/g*m2;
    long long ans=(a1 + x*(a2-a1)/g%m2*m1)%lcm;
    return {(ans+lcm)%lcm,lcm};
}
int main(){
    long long a1,m1,a2,m2; cin>>a1>>m1>>a2>>m2;
    auto res=crt(a1,m1,a2,m2);
    if(res.second==-1) cout<<"No solution"<<endl;
    else cout<<res.first<<endl;
}
```

Explanation Comments: - Extended Euclidean computes x,y such that ax + by = gcd(a,b). - CRT combines two modular congruences into one. - Solution exists if differences are divisible by gcd of moduli.

End of Week 22 - Advanced number theory skills are essential for combinatorics, modular constraints, and math-heavy ACM-ICPC problems. - Practice modular operations, CRT, fast exponentiation, and prime-related functions.