. Basics of Number Theory Divisibility Rules Prime Numbers Composite Numbers Factorization GCD (Greatest Common Divisor) & LCM (Least Common Multiple) Euclidean Algorithm Extended Euclidean Algorithm 50 problem statement in one line for facebook hackeer cup

ChatGPT said:

Here are **50 one-line problem statements** based on the **Basics of Number Theory** topics (Divisibility, Primes, GCD/LCM, Euclidean Algorithm, etc.) — useful for **Facebook Hacker Cup-style practice**, especially with large inputs and numeric constraints:

📌 Divisibility Rules

1. Check if a number is divisible by 2, 3, 5, or 11.
2. Find how many numbers ≤ N are divisible by A or B.
3. Remove minimum digits to make a number divisible by 3.
4. Check if a string-represented number is divisible by 9.
5. Count integers from L to R divisible by D.
6. Find smallest number ≥ N divisible by both A and B.
7. Check if a large number (as string) is divisible by 7.
8. Make a number divisible by 8 by removing at most one digit.
9. Count all substrings divisible by 5 in a numeric string.
10. Find smallest number with given sum of digits divisible by 6.

📌 Prime Numbers

11. Check if a number ≤ 10^9 is prime (multiple test cases).
12. Count primes in range [L, R] ($1 \le R \le 10^6$).
13. Return the Nth prime number ($N \le 10^5$).
14. Count twin primes ≤ N.
15. Find the largest prime factor of a number.
16. Given a number, make it a prime by changing at most one digit.
17. Generate all primes ≤ N using Sieve of Eratosthenes.
18. Sum all prime digits in a number.
19. Find prime gaps ≤ N (difference between consecutive primes).
20. Find the smallest prime ≥ N.

📌 Composite Numbers

21. Find the number of composite numbers in range [L, R].
22. Check if a number is a product of exactly two distinct primes.
23. Find smallest composite number greater than N.
24. Determine if a number can be written as sum of two composites.

25. Count composite divisors of a number.
26. Remove digits to convert a number to composite.
27. Count numbers < N which are not prime or 1.
28. For N queries, check if number is composite or not.
29. Find all numbers between 1 to N having exactly 4 divisors.
30. Construct smallest composite number with given number of digits.

📌 Factorization

31. Find prime factorization of a number $\leq 10^{18}$.
32. Count number of distinct prime factors.
33. Count total number of divisors of N.
34. Sum of all divisors of a number.
35. Count integers ≤ N with exactly K prime factors.
36. Find all pairs (a, b) such that a*b = N.
37. Check if a number can be expressed as product of three integers > 1.
38. Count number of ways to factor a number into 2 or more integers.
39. Print all factor combinations of N.
40. Find largest factor of N less than N.

📌 GCD & LCM

41. Compute GCD of N numbers ($N \leq 10^5$).
42. Compute LCM of an array of N integers.
43. Check if GCD of a pair is 1 or > 1.
44. For each query, find number of pairs with GCD = 1.
45. Given GCD and LCM, find possible number pairs.
46. Count common divisors of A and B.
47. For N queries, return GCD(A, B) and LCM(A, B).
48. Find all numbers that divide both A and B.
49. Check if LCM of array is divisible by K.
50. Count number of integers ≤ N with GCD(A, i) = 1.

These types of problems are suitable for **Facebook Hacker Cup** practice, especially for:

- **Large inputs**
- **String-based number parsing**
- **Optimized math algorithms**
- **Segmented sieve / Prime factor caching**
- **Multiple test cases**

Let me know if you'd like C++/Java code or input/output examples for any of these.

You said:

Large inputs String-based number parsing Optimized math algorithms Segmented sieve / Prime factor caching

ChatGPT said:

Great! Since you're preparing for **Facebook Hacker Cup**-style problems, where **large inputs**, **string-based number parsing**, and **optimized number theory algorithms** are common, here's a breakdown of these advanced concepts with **examples and problem types**.

✅ 1. Large Inputs (up to 10^9 or 10^{18})

You can't brute force these — use optimized math.

◆ Sample Problems:

1. **Count all primes ≤ 10^8** → Use **Segmented Sieve**.
2. **Find GCD of numbers up to 10^{18}** → Use `__int128` or string parsing.
3. **Sum of all divisors of numbers up to 10^{18}** → Optimize with precomputed smallest prime factors.

◆ Techniques:

- Use `scanf/printf` in C++, `BufferedReader` in Java.
- Avoid vector resizing inside loops.
- Modular arithmetic (to avoid overflow).

✅ 2. String-based Number Parsing

Required when numbers are too big for `int` or `long long`.

◆ Sample Problems:

4. **Check if a 500-digit number is divisible by 11**.
5. **Add two 300-digit numbers**.
6. **Check if a large number is divisible by 7 or 13**.

◆ Techniques:

- Use mod rules: `mod = (mod*10 + digit - '0') % d`.
- Store numbers as strings and use modular arithmetic digit-by-digit.

cpp

```
CopyEdit
// C++ Example: Check if string num is divisible by d
bool isDivisible(string num, int d) {
    int rem = 0;
    for (char ch : num)
        rem = (rem * 10 + (ch - '0')) % d;
    return rem == 0;
}
```

✅ 3. Optimized Math Algorithms

◆ Topics and Problems:

7. **Fast GCD** (Euclidean Algorithm) → `gcd(a, b) = gcd(b, a % b)`
8. **Modular Exponentiation** → Compute `a^b % m` in `O(log b)`
9. **Totient Function φ(n)** for all n up to 10^6.
10. **Number of divisors / sum of divisors using prime factorization**.

```
cpp
CopyEdit
// Modular exponentiation
long long modpow(long long a, long long b, long long m) {
    long long res = 1;
    a %= m;
    while (b > 0) {
        if (b & 1) res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
```

✅ 4. Segmented Sieve of Eratosthenes

Used when `N > 10`^{7} but ≤ `10`^{9}. You can't sieve the full array due to memory limits.

◆ Problem Example:

11. **Print all primes between L and R where R ≤ 10^9, R - L ≤ 10^6**

◆ Technique:

1. Generate all primes up to √R using standard sieve.
2. For each prime `p`, mark multiples of `p` in [L, R].

```
cpp
CopyEdit
// Segmented sieve outline
void segmentedSieve(long long L, long long R) {
    vector<bool> isPrime(R - L + 1, true);
```

```
    vector<int> basePrimes = simpleSieve(sqrt(R));
    for (int p : basePrimes) {
        long long start = max(p * p, (L + p - 1) / p * p);
        for (long long j = start; j <= R; j += p)
            isPrime[j - L] = false;
    }
    for (long long i = L; i <= R; ++i)
        if (isPrime[i - L] && i != 1)
            cout << i << "\n";
}
```

✅ 5. Prime Factor Caching / Sieve of Smallest Prime Factors (SPF)

Used for factorization in `O(log N)` per query after precomputation.

◆ Sample Problems:

12. **Find number of distinct prime factors of each number up to N.**
13. **Precompute smallest prime factor (SPF) for all numbers up to 10^7.**

```
cpp
CopyEdit
// Sieve of smallest prime factors
vector<int> spf(N + 1);
void sieveSPF() {
    for (int i = 2; i <= N; i++) spf[i] = i;
    for (int i = 2; i * i <= N; i++) {
        if (spf[i] == i) {
            for (int j = i * i; j <= N; j += i)
                if (spf[j] == j) spf[j] = i;
        }
    }
}

// Factorization using SPF
vector<int> getFactors(int x) {
    vector<int> factors;
    while (x != 1) {
        factors.push_back(spf[x]);
        x /= spf[x];
    }
    return factors;
}
```

□ Facebook Hacker Cup Pattern Summary

Category	Techniques	Time Complexity
Large Input Handling	Fast I/O, Segment Tree, Bit Tricks	O(N log N) or better
Number Parsing	Modular arithmetic with strings	O(N)

Category	Techniques	Time Complexity
Prime Testing	Miller-Rabin (for 64-bit), Sieve, SPF	O(log n) to O(n log log n)
Factorization	SPF, Trial Division, Pollard-Rho (adv.)	O(log n)
GCD/LCM	Euclidean Algorithm	O(log n)
Divisor-related Problems	Factor counting, precompute sums/divs	O(sqrt n) or O(n log n)
Modular Math	Modular inverse, ModExpo, Chinese Remainder	O(log n), varies

---

Would you like me to give:

- Full **practice sheet with test cases**
- or **C++ templates for each concept**
- or build **Facebook Hacker Cup mock contest** based on these?

Let me know!

You said:

□ 2. Modular Arithmetic Modular Addition, Subtraction, Multiplication Modular Exponentiation (Binary Exponentiation / Fast Power) Negative Modulo Handling Modular Inverse Using Extended Euclidean Algorithm Using Fermat’s Little Theorem (when mod is prime) Modular Division 50 problem statement in one line

ChatGPT said:

Here are **50 one-line problem statements** focused on **Modular Arithmetic**, perfect for **Facebook Hacker Cup**, **competitive programming**, or **interview prep**:

□ 2. Modular Arithmetic - 50 One-Line Problems

◆ Modular Addition, Subtraction, Multiplication

1. Compute (A+B)mod M(A + B) \mod M(A+B)modM.
2. Compute (A−B)mod M(A - B) \mod M(A−B)modM ensuring non-negative result.
3. Compute (A×B)mod M(A \times B) \mod M(A×B)modM for large A, B (up to 10^{18}).
4. For given N, compute (1+2+...+N)mod M(1 + 2 + ... + N) \mod M(1+2+...+N)modM.
5. For a list of N numbers, compute their sum modulo M.
6. Compute (A×B×C)mod M(A \times B \times C) \mod M(A×B×C)modM.
7. Find product of all elements in array modulo M.
8. Compute the factorial of N modulo M.
9. Compute A2+B2+C2mod MA^2 + B^2 + C^2 \mod MA2+B2+C2modM.

10. Compute difference between max and min of array modulo M.

◆ Modular Exponentiation (Fast Power / Binary Exponentiation)

11. Compute ABmod MA^B \mod MABmodM (large B).
12. Compute XYZmod MX^{Y^Z} \mod MXYZmodM efficiently.
13. Compute 2Nmod M2^N \mod M2NmodM for large N.
14. Compute (AB+CD)mod M(A^B + C^D) \mod M(AB+CD)modM.
15. Compute (ABC)mod M(A^{B^C}) \mod M(ABC)modM for small A, large B, C.
16. Compute abmod pa^b \mod pabmodp where p is prime.
17. Compute N!mod MN! \mod MN!modM using binary exponentiation.
18. Compute AB+Cmod MA^{B+C} \mod MAB+CmodM using mod rules.
19. For each number in array, compute x2mod Mx^2 \mod Mx2modM.
20. Compute (AB−CD)mod M(A^B - C^D) \mod M(AB−CD)modM.

◆ Negative Modulo Handling

21. Compute (−Amod M)(-A \mod M)(−AmodM) such that result is non-negative.
22. Compute ((A−B)mod M)((A - B) \mod M)((A−B)modM) with negative result correction.
23. Subtract elements in circular array modulo M (non-negative).
24. Normalize a number X to be in [0,M−1][0, M-1][0,M−1].
25. Compute difference between two timestamps modulo 24 hours.
26. Shift array elements by K positions modulo N (with negatives).
27. Given X, find equivalent value modulo M in [0, M).
28. Compute (A−B+M)%M(A - B + M) \% M(A−B+M)%M for positive result.
29. Apply modulo in a rolling hash with negative values.
30. Handle modular subtraction in prefix sum array.

◆ Modular Inverse (Using Fermat / Extended Euclidean)

31. Compute modular inverse of A mod P (P is prime).
32. Compute modular inverse of A mod M (M not necessarily prime).
33. Solve linear congruence equation Ax≡1mod MAx \equiv 1 \mod MAx≡1modM.
34. Given A and B, compute (A/B)mod M(A / B) \mod M(A/B)modM using inverse.
35. Find modular inverse for all numbers 1 to N modulo prime P.
36. Compute modular inverse using Extended Euclidean Algorithm.
37. Implement division modulo M using Fermat’s Little Theorem.
38. Solve for xxx in B·x≡Amod MB \cdot x \equiv A \mod MB·x≡AmodM.
39. Compute A/B+C/Dmod PA / B + C / D \mod PA/B+C/DmodP where P is prime.
40. Compute modular inverse of a factorial number modulo P.

◆ Modular Division

41. Compute (A/B)mod M(A / B) \mod M(A/B)modM using inverse, M is prime.
42. Find (X / Y) mod M where X and Y are coprime with M.
43. For multiple test cases, compute modular division of A by B mod M.
44. Compute (A2/B2)mod M(A^2 / B^2) \mod M(A2/B2)modM.
45. For each test case, solve X·Y−1mod PX \cdot Y^{-1} \mod PX·Y−1modP.
46. Find smallest x such that B·x≡Amod MB \cdot x \equiv A \mod MB·x≡AmodM.
47. Precompute inverse modulo for an array of N elements.
48. Find modular division result where numerator is a large factorial.
49. Compute division modulo M when mod is not prime (Extended GCD).
50. Validate modular division using inverse multiplication method.

These problems cover:

- All basic modular operations
- Advanced inverse/division handling
- Efficient power computation
- Negative number normalization in modulo
- Real-world applications like time shifts and circular arrays

---

Want **code templates in C++ or Java**, or **input/output test cases** for any of these problems? Just ask!

You said:

🔁 3. Prime Number Algorithms Sieve of Eratosthenes Segmented Sieve Sieve of Sundaram (optional/advanced) Sieve of Atkin (optional/advanced)

ChatGPT said:

Here are **50 one-line problem statements** for 🔁 **Prime Number Algorithms**, organized by algorithm type. These are ideal for **Facebook Hacker Cup**, **ICPC**, and **interview prep**, especially for large constraints:

🔁 3. Prime Number Algorithms – 50 One-Line Problems

✅ A. Sieve of Eratosthenes (Basic Sieve)

1. Find all prime numbers ≤ N.
2. Count number of primes between 1 and N.

3. Check if a number ≤ 10^7 is prime using precomputed sieve.
4. Count primes in multiple queries: [L1, R1], [L2, R2], ...
5. Find the sum of all prime numbers ≤ N.
6. Find the product of all prime numbers ≤ N modulo 10^9+7.
7. Precompute smallest prime factor (SPF) for all numbers up to N.
8. Count number of composite numbers ≤ N using sieve.
9. Precompute number of prime factors for each number ≤ N.
10. Find numbers with exactly K distinct prime factors.

✅ B. Segmented Sieve (For large ranges)

11. Find all primes in the range [L, R] where R ≤ 10^9.
12. Count number of primes in range [L, R] using segmented sieve.
13. Check primality for all numbers in [L, R] using sieve of Eratosthenes.
14. Find smallest and largest prime in a range [L, R].
15. Find all twin primes in the range [L, R].
16. Count primes in range [L, R] for multiple test cases.
17. For Q queries, output number of primes in [L[i], R[i]].
18. Count all primes in a range where difference ≤ 10^6 but R ≤ 10^9.
19. Find all primes in [L, R] where L and R both can be > 10^8.
20. Count numbers with no prime divisors in range [L, R].

✅ C. Sieve of Sundaram (Optional, Advanced)

21. Generate all primes ≤ N using Sieve of Sundaram.
22. Count primes ≤ N using Sundaram's logic.
23. Prove that Sieve of Sundaram gives same output as Eratosthenes.
24. Modify Sundaram Sieve to return prime gaps.
25. Generate list of prime indices using Sieve of Sundaram.

✅ D. Sieve of Atkin (Optional, Advanced, Faster than Eratosthenes for large N)

26. Generate all primes ≤ N using Sieve of Atkin.
27. Count primes ≤ N using Atkin and compare with Eratosthenes.
28. Optimize Atkin for multi-threaded CPU.
29. Benchmark Sieve of Atkin vs Sieve of Eratosthenes for N = 10^8.
30. Use Sieve of Atkin to find all primes ≤ 10^7 in under 1 second.

✅ E. Prime Properties and Queries (With Sieve)

31. Count all primes less than N that are palindromes.
32. Find the Nth prime number using sieve array.
33. Count primes with digit sum divisible by 3.
34. Count circular primes ≤ N (where all rotations are prime).
35. Count prime anagrams ≤ N.
36. Find all primes ≤ N that are also Fibonacci numbers.
37. Count numbers ≤ N which are prime and end with 3.
38. Count primes that are also perfect squares (output = 0).
39. Count primes in array A[] using sieve.
40. Count elements in A[] that have a prime number of divisors.

✅ F. Prime Factorization with Sieve

41. Precompute prime factors for numbers ≤ N using SPF.
42. For each number in array, output its prime factor count.
43. For Q queries of number $X \leq 10^7$, output its prime factors.
44. Find numbers ≤ N that are product of exactly two distinct primes.
45. Find all integers ≤ N that have only two unique prime factors.

✅ G. Advanced & Miscellaneous

46. Count number of primes ≤ N using Legendre's formula.
47. Find primes ≤ N that are also Mersenne primes.
48. Count Sophie Germain primes ≤ N.
49. Find prime-rich subarrays in an array of size N.
50. Count primes in a matrix region using preprocessed prime grid.

⚙ Suitable Techniques per Problem

Technique	Problem Range	Use When
Sieve of Eratosthenes	$N \leq 10^7$	Fast & Simple
Segmented Sieve	$R \leq 10^9$, $R - L \leq 10^6$	Memory efficient for large R
SPF (Smallest Prime Factor)	Factorization	Need fast factorization after sieve
Sieve of Atkin	$N \geq 10^7$	High performance with optimization
Sieve of Sundaram	$N \leq 10^6$	Theory, rarely used in contests

---

Would you like:

- Code templates (C++/Java)?
- Test cases for selected problems?
- Full mock contest sheet?

Let me know and I'll tailor it for you.

You said:

□ 4. Theorems in Number Theory Fermat's Little Theorem Euler's Theorem Wilson's Theorem Chinese Remainder Theorem (CRT) Lagrange's Theorem Bezout's Identity

ChatGPT said:

Here are **50 one-line problem statements** based on important **Number Theory Theorems**:

□ 4. Theorems in Number Theory – 50 One-Line Problems

✅ A. Fermat's Little Theorem

(For modulo with prime number)

1. Compute modular inverse of A mod P using Fermat's Theorem.
2. Compute AP−2mod PA^{P-2} \mod PAP−2modP where P is a prime.
3. Given a prime P, test whether AP≡Amod PA^P \equiv A \mod PAP≡AmodP.
4. Find ABmod PA^B \mod PABmodP efficiently using Fermat.
5. Count integers A such that AP−1≡1mod PA^{P-1} \equiv 1 \mod PAP−1≡1modP.
6. Compute (A/B)mod P(A / B) \mod P(A/B)modP using Fermat's inverse.
7. Check Fermat's Little Theorem fails for composite P (Carmichael check).
8. Compute modular inverse of all elements 1 to N modulo P.
9. For given P and N, find count of integers ≤ N coprime to P.
10. Solve AX≡Bmod PA^X \equiv B \mod PAX≡BmodP where P is prime.

✅ B. Euler's Theorem

(Generalization of Fermat's Little Theorem)

11. Compute Aφ(M)mod MA^{φ(M)} \mod MAφ(M)modM where M is not prime.
12. Solve AKmod MA^K \mod MAKmodM using Euler's Theorem when A and M are coprime.
13. Find φ(N) for N ≤ 10^6 using sieve.
14. Precompute φ(N) for all N ≤ 10^6.
15. Solve for A−1mod MA^{-1} \mod MA−1modM using Euler's Theorem.
16. Verify Euler's theorem for given A, M.
17. Count A ≤ N such that A^φ(N) ≡ 1 mod N.
18. Compute modular power ABmod MA^B \mod MABmodM for non-prime M.
19. Count number of integers ≤ N coprime to N (i.e. Euler's φ(N)).

20. Given A and M, find minimum positive X such that AX≡1mod MA^X \equiv 1 \mod MAX≡1modM.

✅ C. Wilson’s Theorem

(Prime P → (P−1)! ≡ −1 mod P)

21. Verify if a given number P is prime using Wilson’s Theorem.
22. Compute (P−1)!mod P(P−1)! \mod P(P−1)!modP and check if result = P − 1.
23. Count primes P such that (P−1)!+1≡0mod P(P−1)! + 1 \equiv 0 \mod P(P−1)!+1≡0modP.
24. Apply Wilson’s theorem to prove primality for small P.
25. For given P, check if Wilson’s congruence holds.

✅ D. Chinese Remainder Theorem (CRT)

(Solve system of modular congruences)

26. Solve: x≡A1mod M1x \equiv A_1 \mod M_1x≡A1modM1, x≡A2mod M2x \equiv A_2 \mod M_2x≡A2modM2 (M1, M2 coprime).
27. Solve 3 equations: x≡aimod mix \equiv a_i \mod m_ix≡aimodmi using CRT.
28. For large moduli, compute x satisfying multiple congruences.
29. Find smallest number such that x ≡ 3 mod 5, x ≡ 4 mod 7.
30. Solve CRT where moduli are not coprime using Extended GCD.
31. Solve birthday puzzle using modular equations (CRT).
32. Given n moduli and remainders, find solution using generalized CRT.
33. Implement CRT to solve linear Diophantine congruences.
34. Find earliest time when 3 repeating events occur together.
35. Solve scheduling problem using CRT and mod inverses.

✅ E. Lagrange's Theorem

(Group theory: Order of element divides order of group)

36. Count number of elements in Zn∗Z_n^*Zn∗ with order dividing k.
37. Find order of an element A modulo M.
38. Given A and M, find smallest k such that Ak≡1mod MA^k ≡ 1 \mod MAk≡1modM.
39. Check whether A is a generator of multiplicative group mod P.
40. List all values x such that xk≡1mod Px^k ≡ 1 \mod Pxk≡1modP.
41. Given group size N, list all divisors of N as possible element orders.
42. Prove that AN≡1mod PA^N ≡ 1 \mod PAN≡1modP for generator A and prime P.
43. Count number of distinct elements of order d in modulo group.
44. Find number of primitive roots modulo P using Lagrange’s Theorem.

45. Solve equation $x^k \equiv 1 \mod p$ using group order.

✅ F. Bézout's Identity

(For integers a, b → ax + by = gcd(a, b))

46. Compute integers x and y such that $ax + by = gcd(a, b)$.
47. Use Bézout to compute modular inverse via Extended Euclidean.
48. Find integer solution to $ax + by = c$ using Bézout's identity.
49. Solve for integer x where $a \cdot x \equiv c \mod b$ using Extended GCD.
50. Determine whether equation $ax + by = c$ has integer solutions.

📌 Summary by Theorem and Use Case

Theorem	Use When…
Fermat's Little	Modulo is prime, inverse via exponentiation
Euler's Theorem	Modulo not prime but A, M coprime
Wilson's Theorem	Prime checking (theoretical), not used for large P
Chinese Remainder	Solve simultaneous congruences efficiently
Lagrange's Theorem	Orders of elements in modulo groups (cyclic, generators)
Bézout's Identity	Modular inverse with non-prime mod, Diophantine equations

---

Would you like:

- C++/Java code snippets for these theorems?
- Practice test cases or problem set for contests?

Let me know — happy to help!

You said:

🧮 5. Euler's Totient Function Definition and Properties Calculating φ(n) efficiently Euler's product formula

ChatGPT said:

Here are **50 one-line problem statements** for 🧮 **Euler's Totient Function** — covering **definition, properties, efficient computation, Euler's product formula**, and their use in contests like **Facebook Hacker Cup**, **ICPC**, and **Codeforces**.

📊 5. Euler’s Totient Function – 50 One-Line Problems

✅ A. Definition and Basic Usage

1. Compute φ(N), the count of integers ≤ N coprime to N.
2. Given A and N, check if gcd(A, N) = 1 (i.e., A ∈ coprime set).
3. Count integers ≤ N that are relatively prime to N.
4. Print all values ≤ N that are coprime to N.
5. Verify that φ(P) = P − 1 for prime P.
6. For N test cases, compute φ(N).
7. Find sum of all integers ≤ N that are coprime to N.
8. Count integers ≤ N with φ(n) = K.
9. Find smallest N such that φ(N) = K.
10. Find number of integers between A and B with φ(n) even.

✅ B. Calculating φ(n) Efficiently

11. Precompute φ(n) for all n ≤ 10^6 using sieve.
12. Compute φ(N) using its prime factorization.
13. Compute φ(n) using `O(log n)` prime factor division.
14. Given N (up to 10^9), compute φ(N) in ≤ O(√n).
15. For Q queries of N, return φ(N) quickly (with precomputation).
16. Compute φ(N!) for N ≤ 10^5.
17. Find φ(N) modulo M.
18. Find product of all φ(n) for n = 1 to N.
19. Find number of values N ≤ 10^6 with φ(N) = N / 2.
20. Count how many φ(N) values are prime for N ≤ 10^6.

✅ C. Euler’s Product Formula

(If n=p1a1⋅p2a2⋯⋅pkakn = p_1^{a_1} \cdot p_2^{a_2} \cdot \dots \cdot p_k^{a_k}n=p1a1⋅p2a2⋯⋅pkak, then:
ϕ(n)=n⋅∏p∣n(1−1/p)\phi(n) = n \cdot \prod_{p|n} (1 - 1/p)ϕ(n)=n⋅∏p∣n(1−1/p))

21. Compute φ(N) using Euler's product formula.
22. Verify Euler's product formula for given N.
23. Given prime factorization of N, compute φ(N).
24. For N = 210 (2×3×5×7), compute φ(N) using product rule.
25. Given N = p^k, compute φ(N).
26. Compute φ(N) without storing φ[1...N] using product rule.
27. Given array of factored numbers, return their φ(N).
28. Compute φ(ab) given φ(a) and φ(b), when gcd(a, b) = 1.

29. Use Euler's product formula to prove that φ(N) is even for N > 2.
30. Find the ratio φ(N)/N for multiple values of N.

✅ D. Totient Function Properties

31. Prove that φ(p^k) = p^k - p^(k−1) for prime p.
32. Show that φ(mn) = φ(m) × φ(n) when m ⊥ n.
33. Compute sum of all φ(d) such that d | N.
34. Count values of n where φ(n) is a power of 2.
35. Find the smallest n for which φ(n) = m, for a given m.
36. Compute gcd(φ(a), φ(b)) for a, b ≤ 10^5.
37. Check if φ(n) divides n.
38. Count values n ≤ N such that φ(n) is square-free.
39. Count values of n for which φ(n) is odd.
40. Compute number of n such that n/φ(n) > k.

✅ E. Applications of φ(n)

41. Compute A^B mod N using φ(N) when A, N are coprime.
42. Count reduced proper fractions with denominator ≤ N.
43. Solve RSA-like encryption using φ(N) for prime P, Q.
44. Use φ(n) to determine order of element in multiplicative group mod n.
45. Use φ(n) in solving ABmod NA^B \mod NABmodN when B is large.
46. Generate Farey sequence of order N using φ(N).
47. Use φ(n) to check if n has a primitive root.
48. Use φ(n) to compute number of coprime pairs (a, b) with a + b = n.
49. Count number of integers n where φ(n) divides n−1.
50. Solve Diophantine equations using φ(N) to simplify mod constraints.

📌 Summary of Techniques

Goal / Topic	Best Technique	Time Complexity
φ(n) for single N	Factor and use product formula	O(√n) or O(log n)
φ(n) for all N ≤ 10^6	Modified sieve (Euler's Totient Sieve)	O(n log log n)
Multiple queries	Precompute φ[] array	O(1) per query
Totient of factorial / multiple	Precompute primes, factorize fast	Depends on use-case
Product formula / proof problems	Prime factor representation	Math + Code logic

---

Would you like:

- C++ or Java code for Euler's Totient Sieve?
- Practice test cases?
- A mock mini-contest based on φ(N)?

Let me know!

You said:

□ 6. Counting and Combinatorics in Number Theory nCr % mod (with large n) Precomputation of Factorials and Inverses Lucas Theorem Stars and Bars Theorem (optional)

ChatGPT said:

Here are **50 one-line problem statements** on **Counting and Combinatorics in Number Theory**, ideal for contests like Facebook Hacker Cup:

□ Counting and Combinatorics (nCr, Factorials, Lucas Theorem)

✅ Factorial and Combinatorics Basics

1. Compute n!mod mn! \mod mn!modm for given n≤106n \leq 10^6n≤106.
2. Count the number of ways to choose rrr items from nnn (mod mmm).
3. Check if nCrmod mnCr \mod mnCrmodm is even or odd.
4. Compute n!k!⋅(n−k)!mod m\frac{n!}{k! \cdot (n-k)!} \mod mk!⋅(n−k)!n!modm.
5. Find nCrnCrnCr for large nnn and small rrr, modulo mmm.
6. Determine the value of (nk)mod p\binom{n}{k} \mod p(kn)modp, where ppp is a prime.
7. Calculate (2nn)mod 109+7\binom{2n}{n} \mod 10^9+7(n2n)mod109+7.
8. Compute the number of combinations where order does not matter, mod mmm.
9. Find number of subsets of a set of size nnn, modulo mmm.
10. Calculate the number of bitstrings of length nnn with exactly kkk ones.

✅ Precomputation Techniques

11. Precompute all n!mod mn! \mod mn!modm for n≤106n \leq 10^6n≤106.
12. Precompute inverse factorials mod 109+710^9+7109+7.
13. Given many nCrnCrnCr queries, compute answers using precomputed factorials.
14. Find first kkk values of Catalan numbers modulo mmm.
15. Precompute nCrmod pnCr \mod pnCrmodp table for all n,r≤1000n,r \leq 1000n,r≤1000.

✅ Modular Inverse for Combinations

16. Compute (nr)mod m\binom{n}{r} \mod m(rn)modm using Fermat's little theorem.
17. Use extended Euclidean algorithm to find modular inverse of r!mod mr! \mod mr!modm.

18. Compute modular division in n!/r!mod mn! / r! \mod mn!/r!modm.
19. Compute nCrr+1mod p\frac{nCr}{r+1} \mod pr+1nCrmodp.
20. Evaluate n!ab·k!mod m\frac{n!}{a^b \cdot k!} \mod mab·k!n!modm.

✅ Lucas Theorem

21. Compute nCrmod pnCr \mod pnCrmodp for large n,rn, rn,r, ppp prime using Lucas Theorem.
22. Use Lucas theorem to find (1018109)mod 13\binom{10^{18}}{10^9} \mod 13(1091018)mod13.
23. Handle nCrmod pnCr \mod pnCrmodp where p≤105p \leq 10^5p≤105, n≤1018n \leq 10^{18}n≤1018.
24. Solve nCrmod mnCr \mod mnCrmodm where mmm is not prime using Chinese Remainder Theorem + Lucas.
25. Use Lucas to compute number of subsets of size kkk from nnn elements modulo prime ppp.

✅ Stars and Bars Theorem (Integer Solutions)

26. Count non-negative integer solutions to x1+x2+...+xk=nx_1 + x_2 + ... + x_k = nx1+x2+...+xk=n.
27. Count positive integer solutions to x1+x2+...+xk=nx_1 + x_2 + ... + x_k = nx1+x2+...+xk=n.
28. Compute number of integer partitions of nnn into kkk parts (modulo mmm).
29. Number of ways to distribute nnn identical balls into kkk distinct boxes.
30. Ways to distribute nnn identical balls into kkk boxes with each getting ≥ 1 ball.

✅ Large Input Combinatorics

31. For TTT test cases, compute nCrmod 109+7nCr \mod 10^9+7nCrmod109+7, n≤106n \leq 10^6n≤106.
32. For n≤1018n \leq 10^{18}n≤1018, r≤106r \leq 10^6r≤106, compute nCrmod pnCr \mod pnCrmodp.
33. Compute nCrmod mnCr \mod mnCrmodm, where mmm is a product of primes.
34. Given array AAA of size nnn, find number of ways to choose 3 distinct elements.
35. Given a set of nnn pairs, count all 2-pair combinations mod mmm.

✅ Applications and Variants

36. Find the number of triangle triplets (a,b,c)(a,b,c)(a,b,c) such that a+b>ca + b > ca+b>c, from array.
37. Number of paths from (0,0) to (n,k) in a grid using combinatorics mod mmm.
38. Compute number of derangements of nnn objects mod 109+710^9+7109+7.
39. Count number of permutations of length nnn with kkk inversions.
40. Number of binary strings of length nnn with no two consecutive 1s.

✅ Mixed/Advanced

41. Compute nth Catalan number using nCr mod 109+710^9+7109+7.

42. Compute Bell numbers Bnmod mB_n \mod mBnmodm using partition formula.
43. Calculate number of surjective functions from A→BA \to BA→B where |A|=n,|B|=k|A|=n, |B|=k|A|=n,|B|=k.
44. Compute total number of permutations of multiset mod 109+710^9+7109+7.
45. Number of integer solutions to equation ax+by+cz=nax + by + cz = nax+by+cz=n, x,y,z≥0x,y,z \ge 0x,y,z≥0.
46. For given nnn, find ∑r=0n(nr)2mod m\sum_{r=0}^n \binom{n}{r}^2 \mod m∑r=0n(rn)2modm.
47. Find number of valid bracket sequences of length 2n2n2n using Catalan.
48. Compute ∑i=1ni⋅(ni)mod m\sum_{i=1}^n i \cdot \binom{n}{i} \mod m∑i=1ni⋅(in)modm.
49. Compute multinomial coefficient (nk1,k2,...,kr)mod m\binom{n}{k_1,k_2,...,k_r} \mod m(k1,k2,...,krn)modm.
50. Count the number of labeled trees for nnn nodes using Cayley's formula mod mmm.

Let me know if you want full **code**, **explanations**, or **test cases** for any of these.

You said:

7. Diophantine Equations Linear Diophantine Equations Number of integer solutions Applications using Extended Euclidean Algorithm

ChatGPT said:

Here are **50 one-line problem statements** based on **Diophantine Equations** (Linear, Integer solutions, Applications using Extended Euclidean Algorithm), suitable for contests like Facebook Hacker Cup:

Diophantine Equations – 50 Problem Statements

1. Find integer solutions to 7x+5y=17x + 5y = 17x+5y=1.
2. Determine if 14x+21y=7714x + 21y = 7714x+21y=77 has integer solutions.
3. Count all integer solutions for 4x+6y=244x + 6y = 244x+6y=24 within 0≤x,y≤100 \le x, y \le 100≤x,y≤10.
4. Solve ax+by=cax + by = cax+by=c where a=35,b=64,c=7a = 35, b = 64, c = 7a=35,b=64,c=7.
5. Find all integer pairs (x,y)(x, y)(x,y) such that 3x−8y=13x - 8y = 13x−8y=1.
6. Determine minimum positive solution of 11x+13y=111x + 13y = 111x+13y=1.
7. Find one solution to 17x+23y=gcd(17,23)17x + 23y = \gcd(17, 23)17x+23y=gcd(17,23).
8. Count number of solutions to 5x+9y=15x + 9y = 15x+9y=1 in range x,y∈[−10,10]x, y \in [-10, 10]x,y∈[−10,10].
9. Find smallest positive solution xxx for 31x+19y=131x + 19y = 131x+19y=1.
10. Does the equation 10x+15y=410x + 15y = 410x+15y=4 have integer solutions?
11. Find all non-negative solutions to 6x+9y=456x + 9y = 456x+9y=45.

12. For given a,b,ca, b, ca,b,c, print “YES” if ax+by=cax + by = cax+by=c is solvable in integers.
13. Given x+2y=5x + 2y = 5x+2y=5, find integer solutions with x>0,y>0x > 0, y > 0x>0,y>0.
14. Count integer solutions of 13x+17y=22113x + 17y = 22113x+17y=221.
15. Find smallest solution of 12x−5y=112x - 5y = 112x−5y=1.
16. Determine integer solutions of 21x+14y=721x + 14y = 721x+14y=7.
17. Check if ax+by=cax + by = cax+by=c has infinitely many solutions.
18. Given x=1mod 2x = 1 \mod 2x=1mod2 and x=3mod 5x = 3 \mod 5x=3mod5, find smallest x.
19. Solve 9x+7y=19x + 7y = 19x+7y=1 using Extended Euclidean Algorithm.
20. Count number of integer pairs for 2x+3y=122x + 3y = 122x+3y=12.
21. Find integer solutions of 100x+25y=125100x + 25y = 125100x+25y=125.
22. Given nnn, solve x+y=nx + y = nx+y=n such that 5x+8y=m5x + 8y = m5x+8y=m.
23. Determine if x≡amod mx \equiv a \mod mx≡amodm and x≡bmod nx \equiv b \mod nx≡bmodn has a solution.
24. Find general solution of ax+by=dax + by = dax+by=d.
25. Find number of solutions to x+y+z=nx + y + z = nx+y+z=n for integers.
26. Given kkk, find all x,y∈Zx, y \in \mathbb{Z}x,y∈Z such that x−2y=kx - 2y = kx−2y=k.
27. Find integer solutions of x+y=1000x + y = 1000x+y=1000 and 3x+2y=20003x + 2y = 20003x+2y=2000.
28. For a given nnn, check if it can be expressed as 3x+5y3x + 5y3x+5y.
29. Count how many xxx exist such that 3x+4y=1003x + 4y = 1003x+4y=100 has integer solution for y.
30. Check solvability of a1x1+a2x2+...+anxn=ca_1x_1 + a_2x_2 + ... + a_nx_n = ca1x1+a2x2+...+anxn=c.
31. Solve 3x+4y=2z3x + 4y = 2z3x+4y=2z in integers.
32. Find smallest positive solution of 97x+83y=197x + 83y = 197x+83y=1.
33. Count integer solutions of 7x+11y=1007x + 11y = 1007x+11y=100.
34. Find all solutions xmod nx \mod nxmodn for ax≡bmod nax \equiv b \mod nax≡bmodn.
35. Solve x≡2mod 3,x≡3mod 4x \equiv 2 \mod 3, x \equiv 3 \mod 4x≡2mod3,x≡3mod4.
36. Given nnn, find integer x,yx, yx,y such that x2−y2=nx^2 - y^2 = nx2−y2=n.
37. Find number of integer solutions for x2+y2=zx^2 + y^2 = zx2+y2=z.
38. Check if ax+by+cz=dax + by + cz = dax+by+cz=d has integer solution.
39. Count all positive integer solutions of 15x+20y=10015x + 20y = 10015x+20y=100.
40. Solve x2−2y2=1x^2 - 2y^2 = 1x2−2y2=1 for smallest x,y>0x, y > 0x,y>0.
41. Solve x3+y3+z3=kx^3 + y^3 + z^3 = kx3+y3+z3=k for given kkk.
42. Given ppp, check if px+1px + 1px+1 is divisible by qqq.
43. Solve ax≡bmod max \equiv b \mod max≡bmodm where gcd(a,m)≠1\gcd(a, m) \neq 1gcd(a,m)□=1.
44. Count integer points on line ax+by=cax + by = cax+by=c.
45. Find smallest x satisfying 2x≡3mod 72x \equiv 3 \mod 72x≡3mod7.
46. Solve linear equation in three variables with integer solutions.
47. Find smallest solution to ax+by=lcm(a,b)ax + by = \text{lcm}(a, b)ax+by=lcm(a,b).
48. Does 9x+15y=309x + 15y = 309x+15y=30 have non-negative integer solutions?
49. For nnn, count solutions to ax+by=nax + by = nax+by=n with x, y > 0.
50. Construct integer solution to 99991x+99989y=199991x + 99989y = 199991x+99989y=1 efficiently.

Let me know if you'd like **solutions**, **code templates (C++/Java/Python)**, or **test cases** for any of these.

You said:

□ 8. Congruences and Modular Systems Solving Linear Congruences Systems of Congruences Applications of CRT

ChatGPT said:

Here are **50 one-line problem statements** covering the topic **Congruences and Modular Systems**—perfect for competitive programming like Facebook Hacker Cup:

□ 8. Congruences and Modular Systems – 50 Problem Statements

1. Solve the congruence equation: `3x ≡ 4 (mod 7)`.
2. Find smallest `x` such that `5x ≡ 1 (mod 11)`.
3. Given `a`, `b`, `m`, check if solution exists for `ax ≡ b (mod m)`.
4. Find all solutions of `x` satisfying `7x ≡ 3 (mod 10)`.
5. Solve for `x`: `12x ≡ 9 (mod 15)` and output smallest positive `x`.
6. Count number of solutions to `ax ≡ b (mod m)` for given `a, b, m`.
7. Given `a1, m1` and `a2, m2`, find `x` such that `x ≡ a1 (mod m1)` and `x ≡ a2 (mod m2)`.
8. Solve a system of 3 congruences using CRT.
9. Find smallest `x` such that `x % 3 = 2`, `x % 5 = 3`, and `x % 7 = 2`.
10. Check whether a system of congruences has a solution.
11. Compute `x mod m` when `x` is negative.
12. Given large mod `m`, solve `a*x ≡ b (mod m)` using Extended Euclidean Algorithm.
13. Find modular inverse of `a` modulo `m` using Fermat (mod is prime).
14. Reduce a congruence with common divisor in `a`, `b`, `m`.
15. Count all solutions of `x` in `0 <= x < m` for `a*x ≡ b (mod m)`.
16. Solve congruence chain: `x ≡ a (mod m)`, `x ≡ b (mod n)`, `x ≡ c (mod p)`.
17. Optimize CRT implementation for very large modulus values.
18. Given a range, count values of `x` satisfying a linear congruence.
19. Reduce congruence `ax ≡ b (mod m)` to standard form.
20. Output smallest solution `x` where `x mod 12 = 4` and `x mod 15 = 9`.
21. Given `n` congruences with pairwise coprime moduli, solve for `x`.
22. Solve `x ≡ 1 (mod 2)`, `x ≡ 2 (mod 3)`, ..., `x ≡ n (mod n+1)`.
23. Construct a number `x` that leaves specific remainders on division.
24. Compute least common multiple of moduli in a system of congruences.
25. Check if a set of congruences can be combined using CRT.
26. Given `x % a = r1`, `x % b = r2`, find `x` or state no solution.
27. Solve modular inverse problem for each `a` in array under mod `m`.
28. Compute `x` such that it satisfies all `xi ≡ ai (mod mi)` for given vectors.
29. Determine number of distinct solutions to `ax ≡ b (mod m)` in `0 ≤ x < m`.
30. Count `x` satisfying two congruences where moduli are not coprime.

31. Construct smallest positive `x` such that `x % 17 = 3` and `x % 19 = 4`.
32. Check if inverse of `a mod m` exists and compute it.
33. Optimize solution of congruences when inputs are large (up to 10^18).
34. Given constraints, minimize `x` satisfying modular equations.
35. Validate if solution exists to system of congruences using extended GCD.
36. Apply CRT in circular clock problems.
37. Solve system of congruences using precomputed modulo inverses.
38. Encode a password using Chinese Remainder Theorem.
39. Implement generalized CRT for non-coprime moduli.
40. Construct number from its residues using CRT and verify uniqueness.
41. Solve simultaneous modular equations with overflow-safe operations.
42. Find next time when 3 clocks with different cycles align.
43. Encode multiple modular equations as one modulo `LCM`.
44. Solve for `x`: `x ≡ a (mod m1)`, `x ≡ b (mod m2)` when `gcd(m1, m2) ≠ 1`.
45. Use extended CRT to solve linear system with inconsistent moduli.
46. Find `x` for `x ≡ 0 (mod 4)`, `x ≡ 3 (mod 5)`, `x ≡ 4 (mod 7)`.
47. Handle congruences involving negative remainders.
48. Find smallest `x` satisfying `x ≡ i (mod pi)` for given primes `pi`.
49. Given values and moduli, reconstruct the number using CRT.
50. Apply CRT to decrypt RSA with multiple primes.

Let me know if you want full code or explanations for any specific problem!

You said:

 9. Quadratic Residues and Non-Residues Definition and Examples Legendre Symbol Euler’s Criterion Quadratic Reciprocity Theorem Tonelli-Shanks Algorithm (modular square roots)

ChatGPT said:

Here are **50 one-line problem statements** for **Quadratic Residues and Non-Residues** in Number Theory, especially useful for **Facebook Hacker Cup-style** contests:

---

9. Quadratic Residues and Non-Residues – 50 Problem Statements

1. Determine if a given integer `a` is a quadratic residue modulo a prime `p`.
2. Find all `x` such that `x² ≡ a (mod p)` for given `a` and prime `p`.
3. Count how many integers from `1` to `p-1` are quadratic residues modulo `p`.
4. Given `a` and `p`, compute the **Legendre Symbol (a/p)**.
5. Compute `a^((p-1)/2) % p` and verify Euler’s criterion for a prime `p`.
6. Given an array of numbers and a prime `p`, find how many are residues mod `p`.
7. Find the smallest quadratic residue modulo a large prime `p`.
8. Find the smallest non-residue modulo a given odd prime `p`.
9. Given `a` and `p`, find a square root of `a` modulo `p` using Tonelli-Shanks.
10. Given two distinct odd primes `p` and `q`, verify the Quadratic Reciprocity Theorem.

11. Implement a function to return +1, -1, or 0 using the Legendre symbol.
12. Solve $x^2 \equiv a \pmod{p}$ when p is a large prime, or return “No Solution”.
13. Precompute Legendre symbols for all $1 \le a < p$ for a given large p.
14. For a given odd prime p, generate all quadratic residues modulo p.
15. Check if two numbers a and b are congruent mod p and both residues.
16. Given a, p, and q, compute (a/p) * (a/q) and compare to (a/(p*q)).
17. Use Euler’s Criterion to verify a list of quadratic residues.
18. Given a composite n, determine whether Euler’s Criterion holds.
19. Verify Wilson's Theorem using properties of quadratic residues.
20. Find all integers x such that $x^2 \equiv -1 \pmod{p}$ for a given prime p.
21. Implement Tonelli-Shanks for multiple test cases efficiently.
22. Given a number a and large p, decide if a has a modular square root.
23. Find the number of distinct modular square roots for numbers modulo p.
24. For large p, find the first 10 quadratic residues in ascending order.
25. Generate the list of non-residues for a given prime p.
26. Calculate (a/p) using Euler’s Criterion and compare with direct method.
27. Check if the product of two residues is always a residue modulo p.
28. For given a and b, check if (a/p)*(b/p) == (ab/p).
29. Solve for x in $x^2 \equiv a \pmod{p}$ when $p \equiv 3 \bmod 4$.
30. Reduce problem size using Quadratic Reciprocity when $a > p$.
31. Given p, count how many a satisfy (a/p) = -1.
32. Prove that there are (p - 1) / 2 quadratic residues mod p.
33. Find an efficient method to determine all x such that $x^2 \equiv a \pmod{p}$ using precomputation.
34. Generate a random number and find its Legendre symbol mod a large prime.
35. Apply Quadratic Reciprocity to compare (p/q) and (q/p).
36. Write a modular exponentiation routine and use it for Euler's Criterion.
37. Implement naive and Tonelli-Shanks solutions and compare runtime.
38. Use properties of quadratic residues to determine a pattern in output.
39. Prove whether 2 is a quadratic residue modulo an odd prime p.
40. Given a, b, and p, determine whether both are residues and compute (a*b/p).
41. Calculate the sum of all quadratic residues modulo a large prime p.
42. Find a generator g such that g^i mod p covers only residues.
43. Use modular inverse with Tonelli-Shanks in a cryptographic setting.
44. Explore quadratic residues modulo non-primes like $n = pq$.
45. Solve $x^2 + 1 \equiv 0 \pmod{p}$ for a prime p.
46. Prove Fermat’s theorem on sums of two squares using residues.
47. Count x such that $x^2 \equiv k \pmod{n}$ for general n.
48. Determine whether the equation $x^2 \equiv a \pmod{m}$ has solution when m is not prime.
49. Show how Legendre Symbol fails when p is not prime.
50. Optimize root finding using CRT with multiple moduli and Tonelli-Shanks.

Let me know if you'd like full solutions, grouped contests, or C++/Java implementations for any of these.

You said:

🔢 10. Base Conversions & Representations Binary, Decimal, Octal, Hex conversions Base b to Base b' Representing Numbers in Different Bases

ChatGPT said:

Here are **50 one-line problem statements** based on **Base Conversions & Representations** suitable for contests like Facebook Hacker Cup:

🔢 Base Conversions & Representations – Problem Statements

1. Convert a given binary string to its decimal representation.
2. Convert a given decimal number to its binary form without using built-in functions.
3. Convert a hexadecimal string to its equivalent decimal number.
4. Convert a decimal number to a base-7 string representation.
5. Convert an octal string to binary using only string manipulation.
6. Given a number in base `b1`, convert it to base `b2`.
7. Check if a binary string represents a power of 2.
8. Find the number of `1`s in the binary representation of a given number.
9. Given a binary string, output its reversed base-10 number.
10. Add two large binary strings and print the sum in decimal.
11. Find the smallest base `b` ($2 \le b \le 16$) in which a number is a palindrome.
12. Convert a number from base 10 to base 62 (digits + lowercase + uppercase).
13. Multiply two base-16 numbers and return the result in base-10.
14. Convert a decimal number to a base `b` string using recursion.
15. Check if a string is a valid representation in base `b`.
16. Given a list of binary numbers, sort them based on their decimal values.
17. Count the number of digits in base `b` for a given decimal number `n`.
18. Given a number, find its representation in base `-2`.
19. Convert a large decimal number (as string) to binary without overflow.
20. Convert a fractional decimal to binary (e.g. 0.625 → 0.101).
21. Find the base in which the number `121` is a square.
22. Find the base `b` such that the number `121` in base `b` is 25 in decimal.
23. Convert a list of hexadecimal colors to binary RGB format.
24. Represent a number in a mixed-base system (e.g., time: HH:MM:SS).
25. Convert base-3 representation to base-9.
26. Compress a binary string by grouping every 4 bits into a hex digit.
27. Find the number of trailing zeroes in binary representation of `n!`.
28. Generate the Gray code sequence for `n` bits and convert to decimal.
29. Convert a number from base `b` to its factorial number system representation.
30. Convert a decimal number to Roman numerals (base-like system).
31. Convert a large hexadecimal number to base 3.
32. Check whether a decimal number has alternating bits in binary.
33. Print all numbers from 1 to `n` in base 5.
34. Count numbers from 1 to `n` whose base-8 representation ends with `7`.
35. Convert a decimal number into base `b` and reverse it.
36. Add two numbers given in base-3 and return in base-3.
37. Check if a number is palindromic in both base 10 and base 2.

38. Count how many numbers ≤ `n` have an even number of 1s in binary.
39. Convert a base-4 number to a base-16 number without converting to decimal.
40. Find the longest prefix of a base-2 string that is a perfect square.
41. Convert a base `b` number to decimal using Horner's method.
42. Given a base-10 number, find its representation in Fibonacci base.
43. Convert all digits of a number to words (e.g. 120 → one-two-zero).
44. Given a string of binary digits, group them into valid ASCII characters.
45. Print all `n`-digit binary numbers with exactly `k` ones.
46. Count the number of digits required to represent `n!` in binary.
47. Convert a decimal number to base `b` using bitwise operations.
48. Add a number to itself in binary without using `+`.
49. Count the number of distinct digits in base `b` representation of `n`.
50. Convert a decimal number to IEEE-754 32-bit binary format.

Would you like this compiled into a PDF or CSV for practice sets or contest prep?

You said:

📏 11. Continued Fractions Construction of Continued Fractions Convergents Application in Rational Approximations Solving Pell's Equation

ChatGPT said:

Here are **50 one-line problem statements** based on the topic **Continued Fractions**, ideal for contests like **Facebook Hacker Cup**:

📏 Continued Fractions: 50 Problem Statements

□ Basics and Construction

1. Convert a rational number `a/b` into its continued fraction representation.
2. Construct a continued fraction for √`N` up to k terms.
3. Determine whether the continued fraction of √`N` is periodic.
4. Given a continued fraction, reconstruct the original rational number.
5. Output the first `k` terms of continued fraction of **π** (approximate).
6. Write a function to generate continued fraction of `e` up to n terms.
7. Check if a continued fraction sequence corresponds to a rational number.
8. Find the number of terms in continued fraction of `a/b`.
9. For given `N`, find length of period in continued fraction of √`N`.
10. Compare two continued fractions and say which represents a larger number.

📏 Convergents and Rational Approximations

11. Compute the first `k` convergents of a given continued fraction.
12. Given a rational number, list all convergents.

13. For given real number `x` and max denominator `d`, find best convergent.
14. Approximate irrational number using continued fraction within epsilon error.
15. Given a convergent, find the next one.
16. Check if a convergent of `√N` gives an approximation within a threshold.
17. Find the closest convergent to `π` with denominator < `10^6`.
18. Given k-th convergent of `e`, compute numerator and denominator.
19. For given continued fraction terms, verify if convergent equals `a/b`.
20. Find which convergent gives best approximation to `√N` under limit.

Pell's Equation

21. Solve Pell's equation `x^2 - N*y^2 = 1` for minimal x, y.
22. For given N, find smallest solution to `x^2 - N*y^2 = -1` if it exists.
23. Count number of `N` ≤ 1000 for which Pell's equation has large minimal x.
24. Determine whether a given `x, y` solve Pell's equation.
25. Given a convergent of `√N`, check if it solves Pell's equation.
26. Compute next solution of Pell's equation from previous solution.
27. Find all solutions of `x^2 - N*y^2 = 1` below given x limit.
28. Find smallest `N` such that Pell's equation has x > 1000.
29. For multiple values of `N`, solve corresponding Pell's equations.
30. Use continued fraction expansion to generate solution of `x^2 - N*y^2 = 1`.

Rational Approximations

31. Given real number x and error e, find best rational approximation.
32. For input x in decimal form, compute continued fraction and best rational.
33. Determine if `a/b` is a convergent of a given real number.
34. Given decimal string, approximate it using convergents.
35. Minimize |x - p/q| over all q ≤ N for irrational x.
36. Generate closest rational number to x using CF up to k steps.
37. Output convergents of x until desired accuracy is met.
38. For decimal representation of `e`, find rational closest within 10^{-6}.
39. Approximate `log(2)` using continued fractions.
40. For given transcendental number, find first convergent with even numerator.

Advanced Continued Fractions

41. Analyze periodicity in CF of `√N` for N in given range.
42. For fixed N, generate matrix form of continued fraction recurrence.
43. Generate reverse CF from last convergent to first.
44. Convert CF of golden ratio into decimal form using k terms.
45. Use continued fraction to detect perfect squares.
46. Construct a continued fraction with Fibonacci numbers as numerators.
47. Optimize Pell's solution using symmetry of CF expansion.
48. Derive generalized CF representation for $\tan^{-1}(x)$.
49. Build recursive solution to find CF terms of algebraic number.
50. Analyze growth of numerator/denominator in convergents of `√2`.

Would you like the **C++ or Python code templates** for solving these?

You said:

🔐 12. Cryptography & Number Theory Public-Key Cryptography (e.g., RSA Algorithm) Primality Testing Miller-Rabin (Probabilistic) Fermat Test Modular Multiplicative Inverse in Cryptosystems

ChatGPT said:

Here are **50 one-line problem statements** related to **Cryptography & Number Theory** with a focus on modular arithmetic, primality testing, and public-key cryptosystems like RSA:

🔐 Cryptography & Number Theory – 50 One-Line Problems

1. Implement RSA key generation with two small prime numbers.
2. Encrypt a message using RSA given public key `(e, n)`.
3. Decrypt an RSA-encrypted message using private key `(d, n)`.
4. Generate two large primes for RSA key generation using Miller-Rabin test.
5. Check if a number is prime using the Fermat primality test.
6. Perform modular exponentiation to encrypt a message using RSA.
7. Find modular inverse of `a` modulo `m` using Extended Euclidean Algorithm.
8. Check primality of a number up to 1e9 using Miller-Rabin.
9. Simulate sending encrypted messages using RSA with random primes.
10. Generate public and private keys for 512-bit RSA encryption.
11. Prove that the modular inverse exists for given `a` and `m`.
12. Factor `n = p*q` given `n` and `phi(n)` to find primes `p`, `q`.
13. Demonstrate RSA decryption failing with non-coprime e and phi(n).
14. Encrypt a numeric password with RSA and then decrypt it.
15. Validate digital signatures using RSA and SHA256 hash.
16. Encode a text message into numbers for RSA encryption.
17. Decrypt a hexadecimal RSA message to get original plaintext.
18. Crack Fermat's primality test by showing a Carmichael number passes it.
19. Apply Chinese Remainder Theorem to solve RSA decryption in parts.
20. Simulate an RSA system with pre-defined small key pairs.
21. Encrypt a string using character-wise RSA encoding.
22. Convert a binary message to decimal for modular encryption.
23. Encrypt and decrypt a 64-bit integer using fast power mod.
24. Perform Fermat's test with multiple `a` values for accuracy.
25. Find the first strong pseudoprime to a base `a`.
26. Detect Fermat liars and witnesses for a number `n`.
27. Count the number of Miller-Rabin iterations needed for 99% accuracy.
28. Generate a probable prime in range [1e6, 1e7] using Miller-Rabin.
29. Use Euler's theorem to simplify a^b mod m in encryption.
30. Create a program to simulate brute-force cracking of small RSA keys.
31. Secure a password using modular hashing.
32. Encrypt a file line by line using RSA.
33. Decrypt a file encrypted using line-wise modular exponentiation.

34. Explain why RSA fails if p = q.
35. Show how modular inverse fails if `a` and `m` are not coprime.
36. Derive Euler's totient for composite `n` = p*q for RSA.
37. Convert a sentence to ASCII before modular encryption.
38. Prove a number is a strong probable prime to base `a`.
39. Encrypt a single-digit number using mini-RSA logic.
40. Use modular inverse in affine cipher encryption.
41. Use CRT to speed up RSA decryption with large primes.
42. Write a function to return all coprime numbers to `n`.
43. Test a batch of numbers for primality using Fermat + Miller-Rabin.
44. Convert base64-encoded message to decimal for encryption.
45. Compute RSA keys where `e` is chosen as 65537.
46. Encrypt two numbers `a` and `b` and find (a+b) mod n after decryption.
47. Detect if RSA key is invalid due to incorrect totient.
48. Implement Miller-Rabin using random bases for multiple checks.
49. Explain modular exponentiation overflow and its fix.
50. Build a secure communication simulator using RSA with user input.

Let me know if you want code for any of these or need a quiz/assignment-style format.

You said:

🎲 13. Miscellaneous Advanced Topics Sum of Divisors Number of Divisors Mobius Function Dirichlet Convolution Inclusion-Exclusion Principle in Number Theory Primitive Roots and Discrete Logarithm Carmichael Numbers

ChatGPT said:

Here are **50 one-line problem statements** for **Miscellaneous Advanced Topics** in Number Theory:

📌 Sum of Divisors / Number of Divisors

1. Compute the sum of all divisors of a number `n`.
2. Find the number of divisors of `n` given its prime factorization.
3. For a range `1 to N`, find the total number of divisors for each number.
4. Count how many numbers from `1 to N` have exactly `k` divisors.
5. Given `n`, find the product of all its divisors.
6. For multiple test cases, output sum of divisors for each number.
7. Count how many numbers less than `n` have sum of divisors > `n`.
8. Check if the sum of proper divisors of `n` is equal to `n` (Perfect Number).
9. Sum all divisors of numbers from `1 to N` efficiently.
10. Find the smallest number with exactly `k` divisors.

🌀 Mobius Function (μ)

11. Calculate the Mobius function value for a number `n`.
12. Compute μ(n) for all `n` in range `1 to 10^6`.
13. Count numbers `<= N` such that μ(n) ≠ 0.
14. Check if `n` is square-free using Mobius function.
15. Use Mobius inversion to compute count of coprime pairs up to `n`.
16. Count the number of positive integers `≤ n` with even number of prime factors.
17. Sum of μ(n) for `1 ≤ n ≤ N`.
18. Evaluate Dirichlet convolution using μ(n) and f(n).
19. Find all numbers `≤ n` for which μ(n) = 1.
20. Use μ(n) to solve inclusion-exclusion-based problems.

🔁 Dirichlet Convolution

21. Define and implement Dirichlet convolution of two functions `f` and `g`.
22. Use Dirichlet convolution to express totient function φ(n).
23. Show that convolution of `1` and `id` is σ(n) (sum of divisors).
24. Given `f(n)` and `g(n)`, find `h(n) = (f * g)(n)` for `n ≤ 100000`.
25. Prove multiplicativity of Dirichlet convolution via code.
26. Compute Dirichlet inverse of Euler's totient function.
27. For given `f(n)`, compute all values of its Dirichlet inverse.
28. Evaluate convolution using prefix sums of divisor functions.
29. Write function to compute convolution of Mobius and identity functions.
30. Given `f` and `g`, verify `f * μ = g` via code.

♻️ Inclusion-Exclusion Principle in Number Theory

31. Count numbers ≤ `N` divisible by `a`, `b`, or `c` using inclusion-exclusion.
32. Find how many numbers up to `N` are not divisible by any of a list of primes.
33. Count integers ≤ `N` having gcd with `K` equal to 1.
34. For set of mod values, find count of numbers ≤ `N` not divisible by any.
35. Use inclusion-exclusion to count co-prime pairs in range.
36. Use mobius inversion and IE to count numbers with at most `k` prime factors.
37. Find how many numbers ≤ `N` are divisible by at least two of the given primes.
38. Count numbers up to `N` which are divisible by neither 2 nor 3 nor 5.
39. Use inclusion-exclusion for union of multiple sets defined by mod constraints.
40. Given N and list of m primes, compute how many numbers ≤ N are coprime to all.

□ Primitive Roots and Discrete Logarithm

41. Find smallest primitive root of a prime number `p`.

42. Given `a, b, m`, find x such that `a^x ≡ b (mod m)` (Discrete Log).
43. Count number of primitive roots modulo `p`.
44. Check whether a number `g` is a primitive root modulo `p`.
45. Solve discrete logarithm problem using Baby-step Giant-step.
46. Compute the discrete logarithm for large `p` (up to 10^9).
47. Print all primitive roots modulo `p` if `p` is prime.
48. Given `g`, generate the full primitive root cycle modulo `p`.
49. Count how many numbers have discrete log equal to a fixed value.
50. Find the order of an integer `a` modulo `n`.

Let me know if you'd like the **C++/Java/Python code templates** for any of these problems.

You said:

📚 Optional for Research or Advanced Study Elliptic Curves Algebraic Number Theory Gaussian Integers Zeta Functions

ChatGPT said:

Here is the **complete Number Theory syllabus** including foundational to advanced research-oriented topics, organized clearly by theme:

□ 1. Basics of Number Theory

- Divisibility and GCD
- Prime Numbers and Fundamental Theorem of Arithmetic
- Euclidean Algorithm
- Extended Euclidean Algorithm
- LCM and GCD properties

🔢 2. Modular Arithmetic

- Modular Addition, Subtraction, Multiplication
- Modular Exponentiation
- Negative Modulo
- Modular Inverses (using Extended Euclidean / Fermat’s Little Theorem)
- Applications: Fast Power, Inverses in combinatorics

🔁 3. Prime Number Algorithms

- Sieve of Eratosthenes
- Segmented Sieve
- Sieve of Sundaram *(optional)*
- Sieve of Atkin *(optional)*
- Prime Factorization
- Prime Factor Caching
- Large Input Handling
- String-Based Number Parsing

□ 4. Theorems in Number Theory

- Fermat's Little Theorem
- Euler's Theorem
- Wilson's Theorem
- Chinese Remainder Theorem (CRT)
- Lagrange's Theorem
- Bézout's Identity

📊 5. Euler's Totient Function

- Definition: $\varphi(n)$ = number of integers $\leq n$ that are coprime to n
- Properties: $\varphi(p^k)$, multiplicative property
- Euler's Product Formula: $\varphi(n) = n * \prod(1 - 1/p)$ over distinct primes
- Precomputing $\varphi(n)$ for 1 to N

□ 6. Counting and Combinatorics in Number Theory

- Computing `nCr % mod` with large `n`
- Precomputing Factorials and Inverses (using Fermat's Little Theorem)
- Lucas Theorem (for large `n`, small prime mod)
- Stars and Bars Theorem *(optional)*
- Inclusion-Exclusion in combinatorics

□ 7. Diophantine Equations

- Linear Diophantine Equations: $ax + by = c$
- Existence of integer solutions using GCD
- Number of integer solutions
- Extended Euclidean Algorithm applications

---

8. Congruences and Modular Systems

- Solving Linear Congruences: ax ≡ b mod n
- Systems of Congruences
- Chinese Remainder Theorem (CRT) applications
- Inverses and modulo properties

---

9. Quadratic Residues and Non-Residues

- Definition and examples
- Euler’s Criterion
- Legendre Symbol
- Law of Quadratic Reciprocity
- Tonelli-Shanks Algorithm (modular square roots)

🔢 10. Base Conversions and Representations

- Binary ↔ Decimal ↔ Octal ↔ Hexadecimal
- Base `b` to base `b'` conversions
- Representing numbers as strings
- String to Big Integer conversion

📏 11. Continued Fractions

- Constructing Continued Fractions for rational numbers
- Convergents and their recurrence
- Applications: Best rational approximations
- Solving Pell’s Equation using continued fractions

🔐 12. Cryptography & Number Theory

- RSA Algorithm & Modular Arithmetic
- Public-Key Cryptography basics
- Modular Inverse in Cryptosystems
- Primality Testing
  - Fermat’s Test
  - Miller-Rabin (Probabilistic)

    - Deterministic versions for small bounds

🎲 13. Miscellaneous Advanced Topics

- Number/Count of Divisors
- Sum of Divisors function
- Mobius Function μ(n)
- Dirichlet Convolution
- Primitive Roots and Discrete Logarithms
- Carmichael Numbers
- Inclusion-Exclusion in arithmetic functions

📚 Optional (Advanced Research Topics)

- Elliptic Curves
- Gaussian Integers
- Algebraic Number Theory (Ideals, Fields)
- Zeta Functions (especially Riemann Zeta, Dirichlet Series)

If you'd like, I can turn this into a **PDF**, **Notion doc**, or **interactive checklist**. Just let me know your preferred format.