**BIG MOD: (Iterative)**

1. **int** bigmod(**int** b, **int** p, **int** m) {
2. **int** res = 1 % m, x = b % m;
3. **while** (p) {
4. **if** (p & 1) res = (res \* x) % m;
5. x = (x \* x) % m;
6. p >>= 1;
7. }
8. **return** res;
9. }

**BITWISE SIEVE:**

1. #define M 100000000
2. **int** marked[M / 64 + 2];
3. **void** sieve(**int** n) {
4. **for** (**int** i = 3; i \* i < n; i += 2) {
5. **if** (!bitCheck(i)) {
6. **for** (**int** j = i \* i; j <= n; j += i + i) {
7. bitOn(j);
8. }
9. }
10. }
11. }
12. **bool** isPrime(**int** num) {
13. **return** num > 1 && (num == 2 || ((num & 1) && !on(num)));
14. }

**MODULAR INVERSE + EXTENDED EUCLID:**

1. pii extendedEuclid(**int** a, **int** b) {
2. **if** (b == 0)
3. **return** pii(1, 0);
4. **else** {
5. pii d = extendedEuclid(b, a % b);
6. **return** pii(d.ss, d.ff - d.ss \* (a / b));
7. }
8. }
9. **int** modularInverse(**int** a, **int** n) { // returns a er modular Inverse ; n dara mod kore
10. pii ret = extendedEuclid(a, n);
11. **return** ((ret.ff % n) + n) % n;
12. }

**FACTORIAL:**

1. ///Digits of N! in Different Base
2. **int** factorialDigitExtended(**int** n, **int** base) {
3. **double** x = 0;
4. **for** (**int** i = 1; i <= n; i++) {
5. x += log10(i) / log10(base); ///Base Conversion
6. }
7. **int** res = ((**int**) x) + 1;
8. **return** res;
9. }
10. ///Prime Factorization of Factorial
11. **void** factFactorize(**int** n) {
12. **for** (**int** i = 0; i < primes.size() && primes[i] <= n; i++) {
13. **int** x = n;
14. **int** freq = 0;
15. **while** (x / primes[i]) {
16. freq += x / primes[i];
17. x = x / primes[i];
18. }
19. printf("%d^%d\n", primes[i], freq);
20. }
21. }
22. ///leading digits in factorial
23. ///Find the first K digits of N!
24. ///k=first k digits
25. **int** leadingDigitFact(**int** n, **int** k) {
26. **double** fact = 0;
27. ///Find log(N!)
28. **for** (**int** i = 1; i <= n; i++) {
29. fact += log10(i);
30. }
31. ///Find the value of q
32. **double** q = fact - floor(fact + EPS);
33. **double** B = pow(10, q);
34. ///Shift decimal point k-1 times
35. **for** (**int** i = 0; i < k - 1; i++) {
36. B \*= 10;
37. }
38. ///Don't forget to floor it
39. **return** floor(B + eps);
40. }
41. ///last digit of factorial
42. **int** last\_digit\_factorial(**int** N) {
43. **int** i, j, ans = 1, a2 = 0, a5 = 0, a;
45. **for** (i = 1; i <= N; i++) {
46. j = i;
47. //divide i by 2 and 5
48. **while** (j % 2 == 0) {
49. j /= 2;
50. a2++;
51. }
52. **while** (j % 5 == 0) {
53. j /= 5;
54. a5++;
55. }
56. ans = (ans \* (j % 10)) % 10;
57. }
58. a = a2 - a5;
59. **for** (i = 1; i <= a; i++)
60. ans = (ans \* 2) % 10;
62. **return** ans;
63. }