❓ 1. Basic Arithmetic Functions (abs, fmod, remainder, div)

1. Given an integer `x`, output `abs(x)`.
2. Given two floats `a` and `b`, find `fmod(a, b)`.
3. Compute `remainder(a, b)` for given doubles `a, b`.
4. Find quotient and remainder of integer division using `div()`.
5. Given N integers, output sum of their absolute values.
6. Given float array, print all numbers' absolute values using `fabs()`.
7. Find smallest positive remainder of dividing `a` by `b`.
8. For a given list, count numbers with even `abs(value)`.
9. Calculate `fmod(a*b, c)` for given `a, b, c`.
10. Implement integer division returning quotient & remainder separately.
11. Compute `abs(a - b)` difference.
12. Use `fmod` to wrap angle `θ` into range `[0, 360)`.
13. Given integers, print which has larger remainder mod 7.
14. Calculate average of absolute deviations from mean.
15. Return sum of all remainders when dividing N by 5.
16. Output `remainder(a, b)` result as positive always.
17. Compare results of `fmod` vs `remainder` for same inputs.
18. Compute absolute value of product of three floats.
19. Output number of negative inputs after applying `abs()`.
20. Find integer pair whose remainder division matches given value.
21. Given 2D coordinates, compute Manhattan distance using `abs()`.
22. Find how many numbers give zero remainder when divided by 3.
23. Use `div()` to compute how many full boxes fit and leftover items.
24. Compute `fmod(x, y)` for multiple test cases.
25. Print `abs(sum - product)` of all integers.
26. Compare `fabs()` and `abs()` results for mixed array.
27. Determine remainder of large number using floating method.
28. Compute sum of positive remainders from all pairs (a, b).
29. Implement modular wrapping with `fmod`.
30. Simulate repeated division showing quotient & remainder each time.

📈 2. Power and Root Functions (sqrt, cbrt, pow, hypot)

1. Find square root of integer N.
2. Compute cube root of floating value.
3. Evaluate `pow(a, b)` for given inputs.
4. Find hypotenuse given legs a, b.
5. Compute nth root using `pow(x, 1.0/n)`.
6. Compare `sqrt(x*x + y*y)` vs `hypot(x, y)`.
7. Find how many digits in `pow(2, n)`.
8. Compute geometric mean using `pow(a*b, 0.5)`.
9. Check if number is a perfect square using `sqrt`.

10. Calculate `pow(base, exponent)` for fractional exponent.
11. Compute `cbrt(a*b*c)`.
12. Find largest integer whose square ≤ N.
13. Calculate `pow(10, log10(x))` equality test.
14. Given triangle sides, check Pythagoras using `hypot`.
15. Compute power table: `pow(i, 2)` for i=1..N.
16. Sum of all cube roots of array elements.
17. Compute ratio of `pow(x,3)` and `cbrt(x)`.
18. Find number whose square is nearest to K.
19. Calculate average of `sqrt` of all positive numbers.
20. Use `hypot` to find distance from origin for N points.
21. Evaluate exponential growth `P*pow(1+r, t)`.
22. Compute `sqrt(pow(x,2)+pow(y,2))`.
23. Find sum of square roots modulo M.
24. Compare precision between manual sqrt and `sqrt()`.
25. Generate list of perfect cubes up to N.
26. Calculate energy = `pow(m*v,2)/2` simplified.
27. Find smallest x such that `pow(x,5) > N`.
28. Compute cube of sum vs sum of cubes.
29. Use `cbrt` for temperature scaling problem.
30. Evaluate multiple powers and sum results.

□ 3. Exponential and Logarithmic Functions (exp, log, log10)

1. Compute e^x for given x.
2. Compute 2^x using `exp2`.
3. Compare `expm1(0.001)` vs `exp(0.001)-1`.
4. Find natural log of input number.
5. Find log base 10 of number.
6. Compute log base 2 of number.
7. Use `log1p(x)` for small x and compare accuracy.
8. Evaluate growth rate `exp(r*t)` of population.
9. Convert between exponential and log forms.
10. Check if `log(exp(x)) == x`.
11. Compute number of bits needed: floor(log2(N))+1.
12. Evaluate expression `log(a*b)` vs `log(a)+log(b)`.
13. Compute compound interest using `exp`.
14. Given y=log10(x), find x.
15. Compute entropy `-Σ p*log2(p)`.
16. Find smallest x such that exp(x) > M.
17. Compute `log10(exp(5))`.
18. Evaluate expression `log(pow(a,b))` using log properties.
19. Approximate log(1+x) manually vs `log1p()`.
20. Find difference between exponential and linear growth.
21. Compute inverse of exponential using `log`.

22. Print table of `exp(x)` for x from -5 to 5.
23. Compute doubling time using `log(2)/r`.
24. Convert between decibel and power using log10.
25. Calculate ln(x) of large dataset.
26. Evaluate normalized log probability.
27. Compute 10^(log10(x)) and compare to x.
28. Find x satisfying exp(x) = 100.
29. Compute e^(-x²) for Gaussian function.
30. Evaluate growth model using `exp2` for base-2 systems.

🔁 4. Rounding and Remainder Functions (ceil, floor, trunc, round)

1. Compute ceil and floor of float value.
2. Round number to nearest integer using `round`.
3. Truncate number to integer part using `trunc`.
4. Compare `ceil` and `floor` difference.
5. Round to long integer using `lround`.
6. Find smallest integer ≥ sqrt(x).
7. Compute fractional part using `modf`.
8. Count numbers rounding up to even values.
9. Sum of all `ceil(x/2)` for given list.
10. Compare `rint` vs `nearbyint` results.
11. Use `modf` to split fractional and integer parts.
12. Compute number of pages = `ceil(total/itemsPerPage)`.
13. Round total cost to nearest rupee.
14. Truncate all decimals in list.
15. Find nearest integer to Pi using `round`.
16. Use `ceil` to allocate minimum rooms for guests.
17. Compare rounding modes for multiple values.
18. Use `floor` to find largest integer ≤ sqrt(N).
19. Calculate average after truncating decimals.
20. Round all elements and compute new sum.
21. Use `modf` to extract decimal parts sum.
22. Compute rounded mean vs truncated mean.
23. Print difference between `round()` and `lround()`.
24. Compute step size with rounding.
25. Determine ceil(log2(N)) using `ceil`.
26. Format number to nearest 0.1 using rounding.
27. Find sum of integer parts from float array.
28. Apply `floor` to find tile count for floor size.
29. Use `round` to approximate sensor readings.
30. Compare truncation error for different rounding functions.

🔄 5. Trigonometric Functions (sin, cos, tan, atan2)

1. Compute sin(x), cos(x), tan(x).
2. Find angle from coordinates using `atan2(y,x)`.
3. Convert degrees to radians and compute sine.
4. Compare `tan(π/4)` with 1.
5. Compute $\sin^2+\cos^2$ and verify ≈1.
6. Find smallest positive angle where sin(x)=0.
7. Plot sine wave values for 0–360°.
8. Use `asin` to recover angle from sine value.
9. Compute tangent of angle given in degrees.
10. Find sum of sine of N numbers.
11. Use `cos` to compute shadow length.
12. Verify angle addition formula using `sin(a+b)`.
13. Compute `atan(1/x)` and compare to π/2 - atan(x).
14. Calculate angular difference using `atan2`.
15. Find resultant vector magnitude using sin/cos.
16. Use `acos` to find angle between two vectors.
17. Compute average sine of evenly spaced points.
18. Check equality sin(x)=cos(π/2−x).
19. Convert polar to Cartesian using cos/sin.
20. Compute phase shift using `atan2`.
21. Use `sin` to simulate oscillation amplitude.
22. Find angle in degrees from sine ratio.
23. Compare sin(x) and sin(x+2π).
24. Determine quadrant using atan2(y,x).
25. Compute maximum of sin(x) over interval.
26. Find number of zero crossings of sine function.
27. Use `tan` to calculate slope from angle.
28. Compute vertical height using sin(θ)*length.
29. Use `cos` to find projection length.
30. Combine trig identities in expression and verify output.

□ 6. Hyperbolic Functions (sinh, cosh, tanh)

1. Compute sinh(x), cosh(x), tanh(x).
2. Verify identity $\cosh^2-\sinh^2=1$.
3. Compute asinh(x) and check inverse.
4. Compare sinh(x) with exp(x)/2 for small x.
5. Calculate tanh growth saturation.
6. Find x where tanh(x) ≈ 1.
7. Compute asinh(sinh(x)) to verify inverse.
8. Evaluate energy using hyperbolic cosine.
9. Simulate sigmoid using tanh(x).
10. Plot tanh output range.
11. Compare tanh(x) with logistic function.

12. Compute acosh(x) for x>1.
13. Use asinh() to approximate log(2x).
14. Check even/odd properties of sinh and cosh.
15. Compute hyperbolic distance approximation.
16. Find smallest x such that cosh(x)=value.
17. Compare growth rates exp(x) vs cosh(x).
18. Compute sinh(x)/x for small x.
19. Evaluate asinh(x) for negative inputs.
20. Use tanh for neural activation simulation.
21. Compute tanh(x) difference for large values.
22. Compute average of cosh for list.
23. Compare inverse results acosh(cosh(x)).
24. Compute difference between sinh and x.
25. Verify limits of tanh as $x \to \infty$.
26. Evaluate hyperbolic triple-angle formula.
27. Compute sinh(x+y) using formula.
28. Find derivative numerically using sinh/cosh.
29. Compare acosh(1) and asinh(0).
30. Compute mean tanh output for multiple x.

⚙ 7. Floating-Point Manipulation (frexp, ldexp, copysign)

1. Split float into mantissa and exponent using `frexp`.
2. Recombine using `ldexp`.
3. Compare `x == ldexp(m, e)` after `frexp`.
4. Copy sign of y to x using `copysign`.
5. Compute next representable float using `nextafter`.
6. Use `scalbn` to multiply x by 2^n.
7. Display exponent part of float using frexp.
8. Compare float and double exponents.
9. Generate sequence by doubling with ldexp.
10. Reverse operation of frexp using ldexp.
11. Change sign using copysign to simulate absolute negation.
12. Compute next larger float of given number.
13. Find mantissa exponent product accuracy.
14. Show binary scaling effect using scalbn.
15. Normalize floats using frexp.
16. Extract exponent to classify magnitude.
17. Adjust brightness by scaling with scalbn.
18. Copy positive sign from one float to another.
19. Compute exponent difference between numbers.
20. Round exponent to nearest integer.
21. Convert small float to normalized form.
22. Display next value toward zero using nextafter.
23. Compare direction between two floats.
24. Compute ldexp of 0.5 with varying exponents.
25. Apply `copysign` to swap signs of pairwise elements.

26. Use scalbn to shift mantissa in power of two.
27. Find distance between two adjacent floats.
28. Normalize large number to $0.5 \leq x < 1$ range.
29. Display difference between nextafter and nexttoward.
30. Demonstrate mantissa change effect using frexp.

□ 8. Classification and Comparison (isfinite, isnan, signbit)

1. Check if input is finite.
2. Detect infinite result from division.
3. Test for NaN from invalid operation.
4. Check signbit of negative zero.
5. Classify number using fpclassify.
6. Count finite numbers in array.
7. Filter NaN values from dataset.
8. Print “INF”, “NAN”, or “NORMAL” classification.
9. Generate infinity and test with isinf().
10. Test sqrt(-1) and handle NaN safely.
11. Compare two floats safely ignoring NaN.
12. Detect overflow producing infinity.
13. Check negative signbit on small negatives.
14. Determine whether number is subnormal.
15. Replace NaN with zero in array.
16. Validate if number is finite before division.
17. Compute ratio only if inputs finite.
18. Detect sign of -0.0 using signbit.
19. Print message for each fpclassify case.
20. Count NaNs produced from invalid inputs.
21. Compare signbit results for array.
22. Detect overflow using isinf().
23. Check input category before mathematical operation.
24. Validate finite exponential results.
25. Handle NaN propagation through calculations.
26. Display type of result after each operation.
27. Compare number’s category (finite/infinite).
28. Check both infinities and NaNs in dataset.
29. Classify each input as zero, normal, or subnormal.
30. Verify safe arithmetic using isfinite() guard.

💡 9. Min/Max and Utility (fmax, fmin, fdim)

1. Find maximum of two floats using fmax.
2. Find minimum of two floats using fmin.

3. Compute fdim(a, b) for given inputs.
4. Print max of three numbers using fmax.
5. Find min of array elements using fmin.
6. Compute positive difference for list pairs.
7. Compare fmax/fmin output to std::max/std::min.
8. Replace negatives with fmax(x,0).
9. Compute fdim(sum, avg) for dataset.
10. Calculate `fmax(fmin(a,b), c)` combinations.
11. Find largest of 10 floating numbers.
12. Compute smallest of multiple floats.
13. Use fdim to compute profit gain (no negative).
14. Compute fmax of x and -x (absolute value trick).
15. Combine fmax and fmin to clamp values.
16. Find smaller angle between two directions.
17. Compare fmax(x, y+1) results.
18. Compute fmin(x, y, z) chained.
19. Use fdim to filter increases in sequence.
20. Compare efficiency between fmax/fmin vs ternary.
21. Find difference if positive else zero.
22. Clamp number to [0,100] using fmin/fmax.
23. Compute difference between largest and smallest numbers.
24. Use fmax to compute upper envelope values.
25. Compare fdim results for multiple pairs.
26. Compute maximum of reciprocals.
27. Compute element-wise max for two arrays.
28. Find positive difference sum across list.
29. Compute fmin of squared values.
30. Implement piecewise function using fmax/fmin.

□ 10. Special Mathematical Constants (M_PI, M_E)

1. Print π with 10 decimal places.
2. Compute area of circle using M_PI.
3. Calculate circumference for given radius.
4. Evaluate e^1 using M_E.
5. Compute sin(M_PI/6).
6. Convert radians to degrees using M_PI.
7. Compute $\log_2(e)$ using M_LOG2E.
8. Find ln(2) using M_LN2.
9. Compute √2 using M_SQRT2.
10. Compare pow(M_E,1) with exp(1).
11. Find cos(M_PI).
12. Compute tan(M_PI/4).
13. Use M_PI to compute sphere volume.
14. Calculate natural exponential using M_E constant.
15. Use constants to verify trigonometric identities.
16. Find distance around semicircle.
17. Compute 2πr directly.

18. Verify M_LOG2E * M_LN2 ≈ 1.
19. Compute degrees-to-radians conversion factor.
20. Calculate e^π and π^e.
21. Use M_SQRT2 to normalize vector.
22. Compute log(M_E) base M_E = 1.
23. Approximate golden ratio using e and π.
24. Find difference between M_PI and 22/7.
25. Use constants in power formula.
26. Compute angle in radians per degree.
27. Verify double-precision accuracy of constants.
28. Compute area of quarter circle.
29. Evaluate function involving both e and π.
30. Output all constants with labels.

? 1. Basic Arithmetic Functions

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    cout << "abs(-5) = " << abs(-5) << endl;
    cout << "fabs(-3.7) = " << fabs(-3.7) << endl;
    cout << "fmod(5.3, 2) = " << fmod(5.3, 2) << endl;
    cout << "remainder(5.3, 2) = " << remainder(5.3, 2) << endl;

    div_t result = div(10, 3);
    cout << "div(10,3): quotient = " << result.quot << ", remainder = " <<
result.rem << endl;
    return 0;
}
```

---

2. Power and Root Functions

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    cout << "sqrt(9) = " << sqrt(9) << endl;
    cout << "cbrt(8) = " << cbrt(8) << endl;
    cout << "pow(2,3) = " << pow(2,3) << endl;
    cout << "hypot(3,4) = " << hypot(3,4) << endl;
    return 0;
}
```

□ 3. Exponential and Logarithmic Functions

```
#include <iostream>
```

```
#include <cmath>
using namespace std;

int main() {
    cout << "exp(1) = " << exp(1) << endl;
    cout << "exp2(3) = " << exp2(3) << endl;
    cout << "expm1(0.001) = " << expm1(0.001) << endl;
    cout << "log(e) = " << log(M_E) << endl;
    cout << "log10(1000) = " << log10(1000) << endl;
    cout << "log2(8) = " << log2(8) << endl;
    cout << "log1p(0.001) = " << log1p(0.001) << endl;
    return 0;
}
```

🔁 4. Rounding and Remainder Functions

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    cout << "ceil(3.2) = " << ceil(3.2) << endl;
    cout << "floor(3.8) = " << floor(3.8) << endl;
    cout << "trunc(-3.7) = " << trunc(-3.7) << endl;
    cout << "round(3.6) = " << round(3.6) << endl;
    cout << "lround(3.6) = " << lround(3.6) << endl;

    double i;
    double frac = modf(3.14, &i);
    cout << "modf(3.14): int=" << i << ", frac=" << frac << endl;
    return 0;
}
```

🔄 5. Trigonometric Functions

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    cout << "sin(π/2) = " << sin(M_PI/2) << endl;
    cout << "cos(0) = " << cos(0) << endl;
    cout << "tan(π/4) = " << tan(M_PI/4) << endl;
    cout << "asin(1) = " << asin(1) << endl;
    cout << "acos(1) = " << acos(1) << endl;
    cout << "atan(1) = " << atan(1) << endl;
    cout << "atan2(1,1) = " << atan2(1,1) << endl;
    return 0;
}
```

□ 6. Hyperbolic Functions

```
#include <iostream>
```

```
#include <cmath>
using namespace std;

int main() {
    cout << "sinh(0) = " << sinh(0) << endl;
    cout << "cosh(0) = " << cosh(0) << endl;
    cout << "tanh(0) = " << tanh(0) << endl;
    cout << "asinh(1) = " << asinh(1) << endl;
    cout << "acosh(1) = " << acosh(1) << endl;
    cout << "atanh(0.5) = " << atanh(0.5) << endl;
    return 0;
}
```

⚙ 7. Floating-Point Manipulation

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    int exp;
    double mantissa = frexp(8.0, &exp);
    cout << "frexp(8): mantissa = " << mantissa << ", exp = " << exp <<
endl;
    cout << "ldexp(0.5,4) = " << ldexp(0.5,4) << endl;
    cout << "scalbn(3,2) = " << scalbn(3,2) << endl;
    cout << "copysign(3, -5) = " << copysign(3,-5) << endl;
    return 0;
}
```

□ 8. Classification and Comparison

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    cout << boolalpha;
    cout << "isfinite(10) = " << isfinite(10.0) << endl;
    cout << "isinf(INFINITY) = " << isinf(INFINITY) << endl;
    cout << "isnan(NAN) = " << isnan(NAN) << endl;
    cout << "signbit(-2.5) = " << signbit(-2.5) << endl;
    return 0;
}
```

💡 9. Min/Max and Utility

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    cout << "fmax(3,5) = " << fmax(3,5) << endl;
```

```
    cout << "fmin(3,5) = " << fmin(3,5) << endl;
    cout << "fdim(7,5) = " << fdim(7,5) << endl;
    return 0;
}
```

□ 10. Special Mathematical Constants (C++20+)

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    cout << "M_PI = " << M_PI << endl;
    cout << "M_E = " << M_E << endl;
    cout << "M_LOG2E = " << M_LOG2E << endl;
    cout << "M_LN2 = " << M_LN2 << endl;
    cout << "M_SQRT2 = " << M_SQRT2 << endl;
    return 0;
}
```