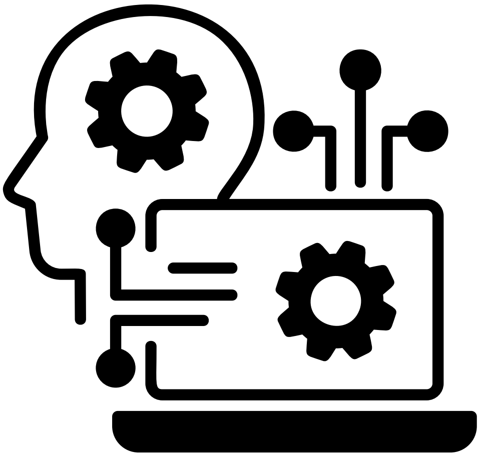
Report on All Functions in "Operations on Numbers"

This report contains all the functions described under the "Operations on Numbers" section.

They are organized into three categories: Basic, Intermediate, and Advanced operations.



----

1. Basic Operations on Numbers

1.1. Sum of Digits

Signature: int sumOfDigits(int num)

Description: Calculates the sum of all digits in a number.

Example: For input 123, the result is 1+2+3=6.

1.2. Reverse Number

Signature: int reverseNumber(int num)

Description: Reverses the digits of a number.

Example: 1234 becomes 4321.

1.3. Palindrome Check

Signature: bool isPalindrome(int num)

Description: Determines whether a number reads the same forwards and backwards.

Example: 121 is a palindrome.

1.4. Prime Check

Signature: bool isPrime(int num)

Description: Verifies if a number is prime (only divisible by 1 and itself).

Example: 7 is prime, but 9 is not.

1.5. Greatest Common Divisor (GCD)

Signature: int gcd(int a, int b)

Description: Uses the Euclidean algorithm to compute the GCD of two numbers.

Example: The GCD of 8 and 12 is 4.

1.6. Least Common Multiple (LCM)

Signature: int lcm(int a, int b)

Description: Finds the smallest number divisible by both a and b.

Example: The LCM of 4 and 6 is 12.

1.7. Factorial

Signature: long factorial(int num)

Description: Calculates the product of all integers from 1 to the given number n.

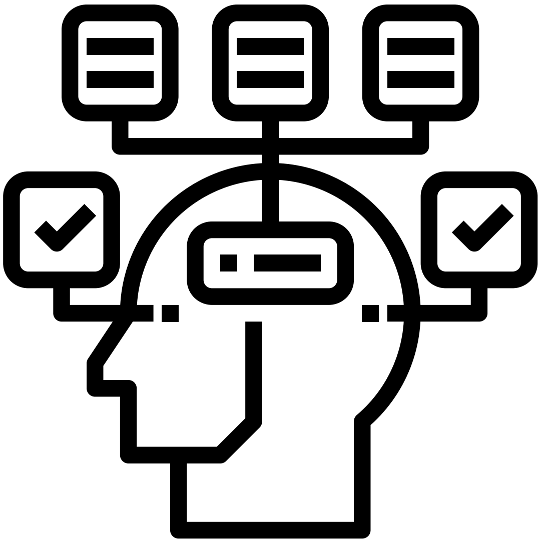
Example: 5! = 1×2×3×4×5 = 120.

1.8. Even or Odd Check

Signature: bool isEven(int num)

Description: Returns true if the number is even; otherwise, false.

Example: 4 is even, while 7 is odd.



----

2. Intermediate Operations on Numbers

2.1. Prime Factorization

Signature: void primeFactors(int num)

Description: Decomposes a number into its prime factors.

Example: 28 = 2×2×7.

2.2. Armstrong Number Check

Signature: bool isArmstrong(int num)

Description: Determines if a number equals the sum of its digits raised to their power count.

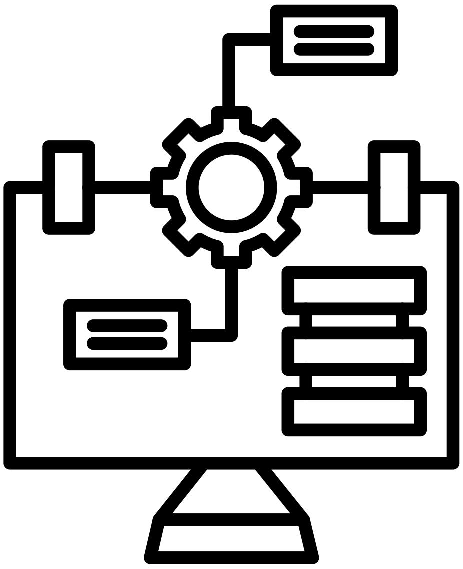
Example: 153 = 1^3 + 5^3 + 3^3.

2.3. Fibonacci Sequence

Signature: void fibonacciSeries(int n)

Description: Generates the Fibonacci sequence up to the nth term.

Example: 0, 1, 1, 2, 3, 5...



2.4. Sum of Divisors

Signature: int sumDivisors(int num)

Description: Computes the sum of all divisors of the number.

Example: Divisors of 6 are 1, 2, 3, 6; sum = 12.

2.5. Perfect Number Check

Signature: bool isPerfect(int num)

Description: Checks if the sum of a number's proper divisors equals the number.

Example: 6 = 1 + 2 + 3.

2.6. Magic Number Check

Signature: bool isMagic(int num)

Description: Verifies if the recursive sum of the digits equals 1.

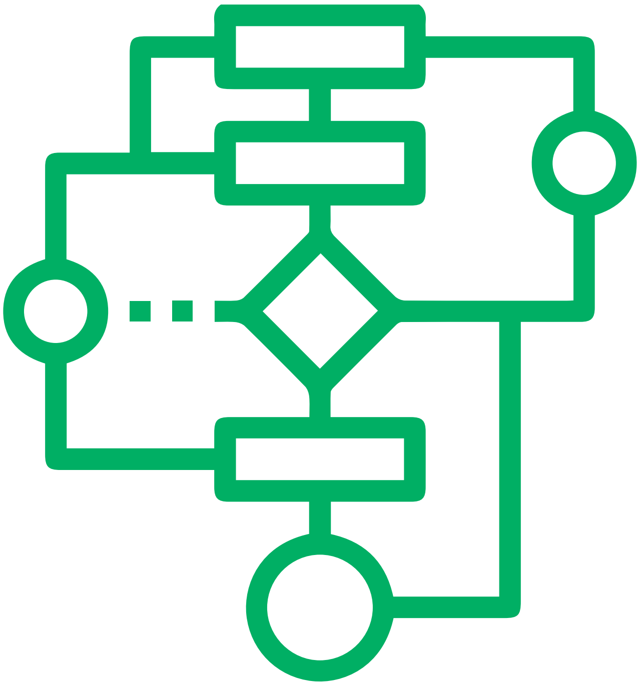
Example: 19 -> 1+9=10 -> 1+0=1.

2.7. Automorphic Number Check

Signature: bool isAutomorphic(int num)

Description: Checks if the square of the number ends with the number itself.

Example: 25^2 = 625 (ends with 25).



----

3. Advanced Operations on Numbers

3.1. Binary Conversion

Signature: void toBinary(int num)

Description: Converts a number to its binary equivalent.

Example: 10 in binary is 1010.

3.2. Narcissistic Number Check

Signature: bool isNarcissistic(int num)

Description: Determines if a number equals the sum of its digits raised to their power count.

Example: 370 = 3^3 + 7^3 + 0^3.

3.3. Square Root Approximation

Signature: double sqrtApprox(int num)

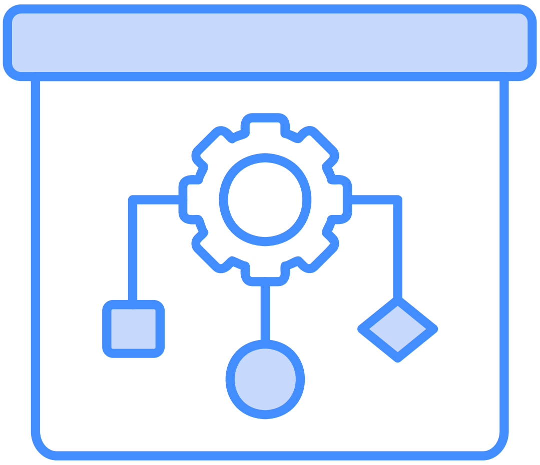
Description: Computes the square root using the Babylonian method for high precision.

3.4. Exponentiation

Signature: double power(int base, int exp)

Description: Calculates base^exp.

Example: 2^3 = 8.



3.5. Happy Number Check

Signature: bool isHappy(int num)

Description: Repeatedly sums the squares of the digits until 1 is reached.

Example: 19 -> 1+81=82 -> 64+4=68...

3.6. Abundant Number Check

Signature: bool isAbundant(int num)

Description: Verifies if the sum of a number's proper divisors exceeds the number itself.

3.7. Deficient Number Check

Signature: bool isDeficient(int num)

Description: Checks if the sum of proper divisors is less than the number.

3.8. Sum of Even Fibonacci Numbers

Signature: int sumEvenFibonacci(int n)

Description: Calculates the sum of all even Fibonacci numbers up to the nth term.

3.9. Harshad Number Check

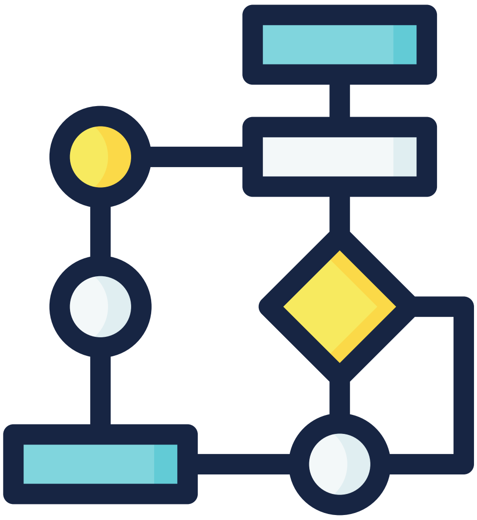
Signature: bool isHarshad(int num)

Description: Verifies if a number is divisible by the sum of its digits.

3.10. Catalan Number Calculation

Signature: unsigned long catalanNumber(int n)

Description: Computes the nth Catalan number for combinatorial problems.



3.11. Pascal Triangle Generation

Signature: void pascalTriangle(int n)

Description: Generates and prints the first n rows of Pascal's triangle.

3.12. Bell Number Calculation

Signature: unsigned long bellNumber(int n)

Description: Computes the nth Bell number for set partitions.

3.13. Kaprekar Number Check

Signature: bool isKaprekar(int num)

Description: Determines if splitting a number's square into two parts and summing them equals the num

Example: 45^2 = 2025 -> 20 + 25 = 45.

3.14. Smith Number Check

Signature: bool isSmith(int num)

Description: Checks if the sum of the digits equals the sum of digits of its prime factors.

3.15. Sum of Prime Numbers

Signature: int sumOfPrimes(int n)

Description: Calculates the sum of all primes less than or equal to n.

These functions cover a comprehensive range of mathematical and numerical analyses, making them a vital part of any numerical operations library. They can be utilized in fields such as number theory, cryptography, and algorithmic problem-solving.