Certainly, here are some simple logical and arithmetic problem-solving programs that you can work on:

1. **Even or Odd Checker:** Write a program that takes an integer input from the user and determines whether it's even or odd.
2. **Factorial Calculator:** Create a program that calculates and prints the factorial of a given positive integer.
3. **Multiplication Table:** Generate the multiplication table for a given number. Allow the user to specify how far the table should go (e.g., up to 10, 12, etc.).
4. **Prime Number Checker:** Write a program that checks if a given number is prime or not.
5. **Palindrome Checker:** Develop a program that checks if a given string is a palindrome (reads the same forwards and backwards).
6. **GCD (Greatest Common Divisor):** Create a program that calculates and prints the greatest common divisor of two integers.
7. **LCM (Least Common Multiple):** Write a program to find and print the least common multiple of two integers.
8. **Simple Calculator:** Build a basic calculator that can perform addition, subtraction, multiplication, and division operations based on user input.
9. **Temperature Converter:** Create a program that converts temperatures between Celsius and Fahrenheit.
10. **Power of a Number:** Write a program to calculate the result of raising a number to a given power.
11. **Sum of Digits:** Develop a program that calculates the sum of the digits of a given positive integer.
12. **Fibonacci Series:** Generate and print the Fibonacci series up to a specified number of terms.
13. **Swap Two Variables:** Write a program to swap the values of two variables without using a temporary variable.
14. **Simple Interest Calculator:** Build a program that calculates and prints the simple interest based on principal amount, interest rate, and time.
15. **Factorization:** Create a program that factors a given positive integer into its prime factors.
16. **Area of Shapes:** Write a program that calculates the area of different geometric shapes such as rectangles, triangles, and circles based on user inputs.
17. **Divisibility Checker:** Develop a program that checks if a number is divisible by another number without leaving a remainder.
18. **Linear Equation Solver:** Build a program that solves a simple linear equation (e.g., ax + b = 0) for the value of x.
19. **Leap Year Checker:** Write a program that determines whether a given year is a leap year or not.
20. **Digit Count:** Create a program that counts the number of digits in a given integer.

These simple programs will help you practice logical thinking and basic arithmetic operations. As you work on them, try to optimize your code for efficiency and readability

Sure, here's a list of top basic programming problem-solving questions that programmers should be able to solve. These problems cover fundamental concepts and are often used to assess coding and problem-solving skills during interviews or practice sessions:

1. **FizzBuzz:** Write a program that prints the numbers from 1 to 100. But for multiples of 3, print "Fizz" instead of the number, and for multiples of 5, print "Buzz". For numbers which are multiples of both 3 and 5, print "FizzBuzz".
2. **Palindrome Check:** Write a function to check if a given string is a palindrome (reads the same forwards and backwards).
3. **Factorial:** Write a function to calculate the factorial of a given positive integer.
4. **Fibonacci Sequence:** Write a function to generate the nth Fibonacci number in the sequence using both iterative and recursive approaches.
5. **Prime Numbers:** Write a function to check if a given number is prime or not.
6. **Reverse a String:** Write a function to reverse a given string.
7. **Anagram Check:** Write a function to check if two given strings are anagrams of each other.
8. **Array Sum:** Given an array of integers, write a function to find the sum of all elements.
9. **Max and Min in an Array:** Write functions to find the maximum and minimum values in an array of integers.
10. **Duplicate Elements:** Write a function to find and remove duplicates from an array.
11. **Sorting Algorithms:** Implement simple sorting algorithms like Bubble Sort, Selection Sort, or Insertion Sort.
12. **Binary Search:** Write a function to perform binary search on a sorted array.
13. **GCD and LCM:** Write functions to find the greatest common divisor (GCD) and least common multiple (LCM) of two integers.
14. **Two Sum:** Given an array of integers, find two numbers that add up to a specific target sum.
15. **Rotate an Array:** Write a function to rotate an array by a given number of positions.
16. **String Manipulation:** Implement basic string manipulation functions like reversing a string, converting to uppercase/lowercase, and trimming whitespace.
17. **Stack Implementation:** Implement a basic stack data structure and its operations (push, pop, peek).
18. **Queue Implementation:** Implement a basic queue data structure and its operations (enqueue, dequeue, peek).
19. **Linked List:** Implement a singly linked list and its basic operations (insertion, deletion, traversal).
20. **Recursion:** Solve problems using recursion, such as calculating the factorial, generating permutations, or solving the Tower of Hanoi problem.

These problems cover a wide range of fundamental programming concepts and can help programmers develop strong problem-solving skills. Keep in mind that practice is key, so try solving these problems in different programming languages to deepen your understanding.