1. Recursive Fibonacci Sequence Generator

Imagine you are developing a program to calculate Fibonacci numbers for a mathematical modeling application. Your function needs to efficiently compute Fibonacci numbers up to a large index requested by the user.

Test Case 1:
Input: 6
Output: 8 (fibonacci(6) = 8)

Test Case 2:
Input: 10
Output: 55 (fibonacci(10) = 55)

Test Case 3:
Input: 0
Output: 0 (fibonacci(0) = 0)

2. Palindrome Checker

You are building a text processing tool that needs to identify palindromes in user input. The function should be able to handle various types of input strings, including alphanumeric characters and punctuation.

Test Case 1:
Input: "radar"
Output: True

Test Case 2:
Input: "hello"
Output: False

Test Case 3:
Input: "Was it a car or a cat I saw?"
Output: True

3. Prime Number Checker

In a cryptography application, you need to verify if a given number is prime to ensure security of generated keys. The function should be accurate and efficient for large numbers.

Test Case 1:
Input: 17
Output: True

Test Case 2:
Input: 15
Output: False

Test Case 3:
Input: 1001
Output: False

4. Anagram Detector

You are developing a word game where players need to identify anagrams of given words. Your function should be capable of quickly determining if two words are anagrams, regardless of letter case or spaces.

Test Case 1:
Inputs: "listen", "silent"
Output: True

Test Case 2:
Inputs: "hello", "world"
Output: False

Test Case 3:
Inputs: "Slot machines", "Cash lost in 'em"
Output: True

5. Recursive Factorial Calculator

In a scientific computing project, you need to calculate factorial values for various mathematical computations. The function should handle both small and large integers efficiently.

Test Case 1:
Input: 5
Output: 120 (5! = 5 * 4 * 3 * 2 * 1)

Test Case 2:
Input: 3
Output: 6 (3! = 3 * 2 * 1)

Test Case 3:
Input: 10
Output: 3628800 (10! = 3628800)

6. Power Function

You are developing a scientific calculator application that requires computing powers of numbers. The function should accurately handle both positive and negative exponents.

Test Case 1:
Inputs: (2, 5)
Output: 32 (2^5 = 32)

Test Case 2:
Inputs: (3, 4)
Output: 81 (3^4 = 81)

Test Case 3:
Inputs: (5, 0)
Output: 1 (5^0 = 1)

7. String Compression

Your data compression algorithm needs a function to compress repetitive sequences of characters in textual data. The function should reduce storage requirements without loss of information.

Test Case 1:
Input: "aabbbccc"
Output: "a2b3c3"

Test Case 2:
Input: "abcd"
Output: "abcd" (no compression needed)

Test Case 3:
Input: "aaabbbcccddd"
Output: "a3b3c3d3"

8. Matrix Transposition

In a matrix manipulation tool for image processing, you need a function to transpose matrices to apply specific transformations efficiently.

Test Case 1:
Input: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Output: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

Test Case 2:
Input: [[1, 2], [3, 4]]
Output: [[1, 3], [2, 4]]

Test Case 3:
Input: [[1, 2], [3, 4], [5, 6]]
Output: [[1, 3, 5], [2, 4, 6]]

9. Word Frequency Counter

You are developing a text analysis tool that requires counting the occurrences of each word in a given document. The function should handle different languages and punctuation marks appropriately.

Test Case 1:
Input: "hello world hello universe world"
Output: {'hello': 2, 'world': 2, 'universe': 1}

Test Case 2:
Input: "a a a b b c"
Output: {'a': 3, 'b': 2, 'c': 1}

Test Case 3:
Input: "To be or not to be, that is the question."
Output: {'to': 2, 'be': 2, 'or': 1, 'not': 1, 'that': 1, 'is': 1, 'the': 1, 'question': 1}

10. Sorting Algorithm Implementation

In a data analytics application, you need to sort large datasets to perform efficient analysis. Your function should implement a robust sorting algorithm to handle various data types and sizes.

Test Case 1:
Input: [5, 2, 9, 1, 5]
Output: [1, 2, 5, 5, 9]

Test Case 2:
Input: [8, 3, 1, 7, 4]
Output: [1, 3, 4, 7, 8]

Test Case 3:
Input: [100, 20, 50, 80, 10]
Output: [10, 20, 50, 80, 100]

11. Matrix Multiplication

In a scientific computing application, you need to multiply matrices to solve complex mathematical equations. The function should be optimized for performance to handle large matrices.

Test Case 1:
Input: [[1, 2], [3, 4]], [[5, 6], [7, 8]]
Output: [[19, 22], [43, 50]]

Test Case 2:
Input: [[1, 2, 3], [4, 5, 6]], [[1, 4], [2, 5], [3, 6]]
Output: [[14, 32], [32, 77]]

Test Case 3:
Input: [[1, 2], [3, 4], [5, 6]], [[1, 1, 1], [1, 1, 1]]
Output: Error (Mismatched dimensions for matrix multiplication)

12. String Reversal

You are developing a text manipulation tool where users can reverse the order of characters in a string.

Test Case 1:
Input: "hello"
Expected Output: "olleh"

Test Case 2:
Input: "hello world"
Expected Output: "dlrow olleh"

Test Case 3:
Input: "12345"
Expected Output: "54321"

13. Calculating Building Dimensions

You are an architect working on the design of a new office building. As part of your design process, you need to calculate the dimensions of various structural elements, such as the size of the foundation, the height of the floors, and the thickness of the walls.
One of the key calculations you need to perform is finding the square root of the total floor area of the building. This will help you determine the appropriate size and layout of the foundation.

Write a program that includes a function **“findSquareRoot()”** that takes a float value representing the total floor area and returns the square root of that value as a float. Use this function in your program to calculate the square root of the floor area and display the result.
Assume that the user will input the total floor area of the building when prompted.

Test Case 1:
Input: 100
Output: The square root of 100.00 is 10.00

Test Case 2:
Input: 225.0
Output: The square root of 225.00 is 15.00

Test Case 3:
Input: -16.0
Output: Error: Cannot calculate the square root of a negative number.

14. Analyzing Text Documents

You are working on a project that involves analyzing text documents to extract various statistics, such as the number of words, the number of unique words, and the length of the longest word. As a first step, you need to write a program that can determine the length of a given string.

Your program should include a function called **“findStringLength()”** that takes a string as input and returns the length of the string as an integer. You will then use this function

in your main program to prompt the user to enter a string, call the findStringLength() function, and display the length of the string.
This functionality will be a crucial component of your larger text analysis project, as it will allow you to quickly and accurately determine the length of words, sentences, and entire documents.

Test Case 1:
Input: ""
Output: The length of the string is: 0

Test Case 2:
Input: " Hello, world! "
Output: The length of the string is: 17

Test Case 3:
Input: "Hello, world!"
Output: The length of the string is: 13

15. Calculating Sums for Budgeting

You are working on a budgeting application that helps users keep track of their expenses. One of the features you want to implement is the ability to calculate the sum of all even or odd expenses within a given range.
Your program should include a recursive function called **"sumEvenOdd()"** that takes the starting number, ending number, and a flag to indicate whether to sum even or odd numbers. The function should return the sum of all even or odd numbers in the given range.

In the main program, you will prompt the user to enter the starting and ending numbers of the range, as well as a choice to sum either even or odd numbers. You will then call the "sumEvenOdd()" function with the user-provided inputs and display the result.
This functionality will be useful for users who want to analyze their spending patterns and identify areas where they can cut back on expenses.

Test Case 1:
Input:
Start = 2, End = 10, Choice = 1
Output:
The sum of even numbers in the range 2 to 10 is: 30

Test Case 2:
Input:
Start = 3, End = 11, Choice = 0
Output:
The sum of odd numbers in the range 3 to 11 is: 36

Test Case 3:
Input:
Start = -4, End = -2, Choice = 1
Output:
The sum of even numbers in the range -4 to -2 is: -6

16. GCD (Greatest Common Divisor) Calculator

You are developing a utility for a mathematics app that needs to compute the greatest common divisor of two numbers provided by users to help them simplify fractions. Create a function to compute the greatest common divisor of two numbers using recursion.

Test Case 1:
Inputs: 48, 18
Output: 6

Test Case 2:
Inputs: 101, 103
Output: 1

Test Case 3:
Inputs: 56, 98
Output: 14

17. Binary to Decimal Conversion

You are developing a digital logic simulation tool for an electronics engineering course. As part of this tool, you need to implement a feature that allows students to convert binary numbers (represented as strings) to their decimal equivalents. This will help students understand and verify their work on binary arithmetic and digital circuit design. Create a function to convert a binary number (given as a string) to its decimal equivalent.

Test Case 1:
Input: "1010"
Output: 10

Test Case 2:
Input: "1111"
Output: 15

Test Case 3:
Input: "100101"
Output: 37

18. Convert Lowercase into Uppercase using Function

Develop a C program that processes user input and needs to convert any lowercase letters entered into uppercase for consistency and further processing. You decide to implement a function specifically for this conversion task to maintain clarity and reusability.

Test Case1:
Input:hello
Output:HELLO

Test Case 2:
Input:bit
Output:BIT

Test Case 3:
Input:bannari
Output:BANNARI

19. Leap Year Check using Function

Developing a program that needs to determine if a given year is a leap year. To achieve this, you decide to implement a function that performs the leap year check based on the following criteria:

A year is a leap year if it is divisible by 4.
However, if the year is divisible by 100, it is not a leap year unless it is also divisible by 400.

Test Case 1:
Input:2016
Output:Leap Year

Test Case 2:
Input:2023
Output:Non-Leap Year

Test Case 3:
Input:1997
Output:Non-Leap Year

20. Remove Duplicates in a given array using Function

Develop a program that processes an array of integers and needs to remove any duplicate elements. To achieve this, you decide to implement a function that:
Takes an array of integers as input.
Removes duplicates from the array while maintaining the order of remaining elements.
Returns the new size of the array after removing duplicates.

Test Case1:
Input:{1, 2, 2, 3, 4, 4, 5}
Output:1 2 3 4 5

Test Case 2:
Input:{5, 5, 5}
Output:5

Test Case 3:
Input:{6, 3, 8, 2, 6, 3, 8, 3}
Output:2 3 6 8

21. Reverse the sentence

Implement functions in reversing sentences, which can be a technique used to achieve a particular style or effect, such as emphasizing the importance of certain words or creating a sense of disorientation.

Test Case 1:
Input : The quick brown fox jumps over the lazy dog
Output : god yzal eht revo spmuj xof nworb kciuq ehT

Test Case 2:
Input : Hello, world!
Output : !dlrow ,olleH

Test Case 3:
Input : The quick brown fox.
Output : .xof nworb kciuq ehT

22. Sum of primes

Some mathematical competitions and problem-solving challenges involve finding whether a given number can be expressed as the sum of two prime numbers. A number theory calculator with this feature can help participants quickly solve such problems and explore related concepts. Write a program to find whether the given number can be expressed as the sum of two prime numbers.

Test Case 1:
Input number = 34
Output: True
Explanation: 34 can be expressed as the sum of 3 (a prime) and 31 (a prime), so the function should return True.

Test Case 2:
Input number = 37
Output: False
Explanation: 37 cannot be expressed as the sum of two prime numbers (since it is a prime number itself), so the function should return False.

Test Case 3:
Input number = 28
Output: True
Explanation: 28 can be expressed as the sum of 5 (a prime) and 23 (a prime), so the function should return True.

23. Armstrong or perfect numbers

In various scientific and engineering domains, dealing with large datasets or numerical simulations, it may be necessary to validate the integrity of the data by checking for special number properties, such as Armstrong numbers or Perfect numbers. Develop a program to find the given number is an armstrong or perfect number.

Test Case 1:
Input number = 6
Expected Output: Perfect number

Test Case 2:
Input number = 28
Expected Output: Not a perfect number

Test Case 3:
Input number = 153
Expected Output: Armstrong number

24. Simple Calculator using Function

Develop a program for a simple calculator that performs basic arithmetic operations (addition, subtraction, multiplication, division). To ensure modularity and reusability, you decide to implement each arithmetic operation as a separate function.

Test Case 1:
Input:'+'
a=5
b=3
Output:8

Test Case 2:
Input:'-'
a=3
b=8
Output:-5

Test Case 3:
Input:'/'
a=10
b=5
Output:2

25. Counting the Even Numbers in a given list using Function

Developing a C program that needs to count the number of even numbers in a list of integers. To achieve this, you decide to implement a function that takes the list and its size as parameters, iterates through the list, and counts how many numbers are even.

Test Case 1:
Input:
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Output:5

Test Case 2:
Input:
{1, 3, 5, 7, 9}
Output:0

Test Case 3:
Input:
{6, 44, 54, 53}
Output:3

26. Counting the Odd Numbers in a given list using Function

Implement a C program that needs to count the number of odd numbers in a list of integers. To achieve this, you decide to implement a function that takes the list and its size as parameters, iterates through the list, and counts how many numbers are odd.

Test Case 1:
Input:
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Output:5

Test Case 2:
Input:
{1, 3, 5, 7, 9}
Output:5

Test Case 3:
Input:
{6, 44, 54, 53}
Output:1

27. Unique Character Identification in Text Messages

You are developing a mobile messaging application that allows users to send and receive text messages. One of the features you want to implement is the ability to identify the first unique character in a message.
Your task is to write a function that takes a string as input and returns the first non-repeating character in the string. If all characters in the string are repeating, the function should return a null character or a special value to indicate that no unique character was found.

Test Case 1:
Input:
Hello
Output:
First non-repeating character is h

Test Case 2:
Input:
Abab
Output:
First non-repeating character is a

Test Case 3:
Input:
aabbcc
Output:
First non-repeating character is Null