Lab Manual

Practical and Skills Development

CERTIFICATE

THE ASSIGNMENT ENTERED IN THIS REPORT HAVE BEEN SATISFACTORILY PERFORMED BY

Registration No	: 25BSA10124
Name of Student	: SHAHZAD AHMED
Course Name	: Introduction to Problem Solving and Programming
Course Code	: CSE1021
School Name	: SCAI
Slot	: B11+B12+B13
Class ID	: BL2025260100796
Semester	: FALL 2025/26

Course Faculty Name : Dr. Hemraj S. Lamkuche

Signature:

VIT BHOPAL

Practical Index

S. No.	Title of Practical	Date of Submission	Signature of Faculty
1	Euler_phi	29/09/2025	
2	Mobius	29/09/2025	
3	Divisor_Sum	29/09/2025	
4	Prime_pi	29/09/2025	
5	Legendre_symbol	29/09/2025	
6	Factorial	04/10/2025	
7	Palindrome	04/10/2025	
8	Mean of Digits	04/10/2025	
9	Digital Root	04/10/2025	
10	Abundant No.	04/10/2025	
11	Deficient	11/10/2025	
12	Harshad	11/10/2025	
13	Automorphic	11/10/2025	
14	Pronic	11/10/2025	
15	Prime Factors	11/10/2025	

16	count_distinct_prime_factors	25/10/2025	
17	is_prime_power	25/10/2025	
18	is_mersenne_prime	25/10/2025	
19	twin_primes	25/10/2025	
20	count_divisors	25/10/2025	
21	Aliquot_sum	01/11/2025	
22	Are_amicable	01/11/2025	
23	Multiplicative_persistence	01/11/2025	
24	Is_Highly_Composite	01/11/2025	
25	Modular Exponentiation	01/11/2025	
26	Modular Multiplicative Inverse	09/11/2025	
27	Chinese Remainder Theorem	09/11/2025	
28	Quadratic Residue Check	09/11/2025	
29	Order_Mod	09/11/2025	
30	Fibonacci Prime Check	09/11/2025	
31	Lucas Numbers Generator	16/11/2025	
32	Perfect Powers Check	16/11/2025	
33	Collatz Sequence Length	16/11/2025	

34	Polygonal Numbers	16/11/2025	
35	Carmichael Numbers	16/11/2025	
36	Miller-Rabin Test	16/11/2025	
37	Pollard_rho(n)	16/11/2025	
38	Zeta_approx	16/11/2025	
39	Partition Function	16/11/2025	

Practical No: 1

Date: _____

TITLE: Euler phi (Euler's Totient Function)

AIM/OBJECTIVE(s): To calculate the count of positive whole numbers less than or equal to a given number N that are co-prime to N.

METHODOLOGY & TOOL USED: Implementation in a programming language like Python. The method involves finding the prime factorization of N and applying the multiplicative formula, or iterating through all numbers from one up to N and checking if the greatest common divisor between N and the current number is equal to one.

BRIEF DESCRIPTION: This function determines the size of the set of relatively prime integers. It is a fundamental concept in number theory and is essential for calculations involving modular arithmetic, especially in Euler's theorem, which is critical for modern cryptography.

RESULTS ACHIEVED:

```
Enter a positive integer to calculate Euler's totient function:  4
Euler's totient function for 4 is: 2
Time elapsed: 0.000005 seconds
Memory utilized (delta): 0 bytes
```

DIFFICULTY FACED BY STUDENT: Correctly implementing the prime factorization for large N, or facing performance issues (long execution time) if the simple iterative approach is used without optimization for very large input values.

SKILLS ACHIEVED: Prime factorization algorithm implementation, understanding and application of a core number theoretic function, and algorithm efficiency analysis.

Practical No: 2

Date: _____

TITLE: Mobius (Möbius function)

AIM/OBJECTIVE(s): To determine the sign and square-free nature of a positive whole number N.

METHODOLOGY & TOOL USED: Implementation in a programming language. The core method is prime factorization. The function returns the number one if N is square free and has an even count of distinct prime factors. It returns negative one if N is square free and has an odd count of distinct prime factors. It returns zero if N is not square free, meaning it is divisible by a prime number raised to the power of two or more.

RESULTS ACHIEVED:

```
Möbius Function Calculator
--------------------------
Enter a positive integer (or 'q' to quit): 3
μ(3) = -1

Enter a positive integer (or 'q' to quit): 25
μ(25) = 0

Enter a positive integer (or 'q' to quit): 10
μ(10) = 1
```

BRIEF DESCRIPTION: The Mobius function is a multiplicative function used in number theory, particularly in the Möbius Inversion Formula, which relates a function and its sum over divisors.

DIFFICULTY FACED BY STUDENT: Distinguishing the cases for a square free number (power of each prime factor is one) versus a non square free number (power of at least one prime factor is two or greater) within the prime factorization code.

SKILLS ACHIEVED: Advanced prime factorization techniques, conditional logic based on parity (even versus odd count), and understanding of square-free numbers.

Practical No: 3

Date: _____

TITLE: Divisor Sum (Sum of Divisors Function)

AIM/OBJECTIVE(s): To calculate the total sum of all positive whole numbers that evenly divide the input number N, including the number one and N itself.

METHODOLOGY & TOOL USED: Implementation in a programming language. The basic method iterates through all whole numbers from one up to N and adds all numbers where the remainder of N divided by the number is zero. An optimized method uses the prime factorization of N.

BRIEF DESCRIPTION: This function, also known as sigma function, is fundamental in the study of numbers, used to classify them as perfect, abundant, or deficient, based on how the sum relates to the number itself.

RESULTS ACHIEVED:

```
Enter a number n:  6
Enter a number p:  7

--- Sum of Divisors Results ---
For n:
σ(6) = 12
Execution time: 0.000005 seconds
Memory utilized (delta): 0 bytes

For p:
σ(7) = 8
Execution time: 0.000002 seconds
Memory utilized (delta): 0 bytes
```

DIFFICULTY FACED BY STUDENT: Writing an efficient solution that does not have a long running time for large input numbers, often achieved by iterating only up to the square root of N to find pairs of divisors.

SKILLS ACHIEVED: Looping and conditional statement mastery, optimization techniques for finding divisors, and applying basic number classification.

Practical No: 4

Date: _____

TITLE: Prime pi (Prime-counting function)

AIM/OBJECTIVE(s): To determine the total count of prime numbers that are less than or equal to the input number N.

METHODOLOGY & TOOL USED: Implementation using a sieve method, such as the **Sieve of Eratosthenes**, in a programming language. The sieve efficiently marks all composite (non-prime) numbers up to N, leaving only the prime numbers to be counted.

BRIEF DESCRIPTION: The prime counting function is essential for understanding the distribution of prime numbers. The Prime Number Theorem provides a famous approximation for this count, showing how the primes thin out as numbers get larger.

RESULTS ACHIEVED:

```
Enter a number n:  13
Enter a number p:  11

--- Prime Counting Function Results ---
For n:
π(13) = 6
Execution time: 0.000045 seconds
Peak memory: 112 bytes

For p:
π(11) = 5
Execution time: 0.000022 seconds
Peak memory: 112 bytes
```

DIFFICULTY FACED BY STUDENT: Correctly implementing the memory allocation and indexing for the sieve array, and understanding the logic of marking multiples as non-prime.

SKILLS ACHIEVED: Advanced array manipulation, implementation of a well-known combinatorial algorithm (Sieve), and managing memory for large data structures.

Practical No: 5

Date: _____

TITLE: Legendre symbol

AIM/OBJECTIVE(s): To check if a number 'a' is a perfect square when considered in relation to a prime modulus 'p'. This determines if the congruence of a variable squared is equivalent to 'a' modulo 'p' has a solution.

METHODOLOGY & TOOL USED: Implementation in a programming language using Euler's criterion. This involves calculating the modular exponentiation of 'a' raised to the power of the quantity 'p' minus one, then divided by two, with the final result being the remainder when divided by 'p'.

BRIEF DESCRIPTION: This symbol is a tool from number theory used to study quadratic residues. It is a fast way to determine if a congruence equation of the second degree is solvable, and it is a key component of the Law of Quadratic Reciprocity.

RESULTS ACHIEVED:

```
Enter a number (n):  5
Enter a prime number (p):  7
Legendre(5/7) = -1
Execution time: 0.000004 seconds
Peak memory: 0 bytes
```

DIFFICULTY FACED BY STUDENT: Understanding the mathematical theory behind Euler's criterion and correctly implementing the modular exponentiation for very large base and exponent values.

SKILLS ACHIEVED: Mastery of the efficient modular exponentiation algorithm and application of advanced number theory concepts to solve quadratic congruences.

Practical No: 6

Date: _____

TITLE: Factorial

AIM/OBJECTIVE(s): To calculate the product of all positive whole numbers from one up to the input number N.

METHODOLOGY & TOOL USED: Implementation in a programming language. A simple iterative loop starts with the number one and repeatedly multiplies this running product by each successive whole number up to N. Alternatively, a recursive function can be used.

BRIEF DESCRIPTION: The factorial function is critical in combinatorics and probability, as it represents the number of distinct ways to arrange a set of N distinct items.

RESULTS ACHIEVED:

```
Enter a non-negative integer for factorial calculation: 12
The factorial of 12 is: 479001600
Execution time: 0.000041 seconds
Peak memory usage: 96 bytes
```

DIFFICULTY FACED BY STUDENT: Dealing with the fact that the result grows extremely fast, leading to overflow errors when using standard integer data types for even moderately small input numbers (N greater than about twenty).

SKILLS ACHIEVED: Implementation of iterative and recursive solutions, and understanding of data type limitations and arbitrary-precision arithmetic libraries for handling large numbers.

Practical No: 7

Date: _____

TITLE: Palindrome

AIM/OBJECTIVE(s): To check if a given number reads exactly the same forwards and backwards.

METHODOLOGY & TOOL USED: Implementation using a programming language. The most common method involves reversing the digits of the original number and then comparing the new reversed number to the original number. This can be done by treating the number as a string or through numerical manipulation (repeatedly taking the remainder when divided by ten and integer dividing by ten).]

BRIEF DESCRIPTION: A palindromic number is symmetric. This concept is simple but is often used as an exercise in string and number manipulation in programming.

RESULTS ACHIEVED:

```
Enter a number to check: 121
Is 121 a palindrome? True
Execution time: 0.000041 seconds
Peak memory usage: 88 bytes
```

DIFFICULTY FACED BY STUDENT: Correctly reversing the numerical value without using string conversion, which requires careful handling of the digit extraction and reconstruction process to avoid losing information or creating off-by-one errors.

SKILLS ACHIEVED: Numerical manipulation of digits, algorithmic reversal techniques, and conditional logic.

Practical No: 8

Date: _____

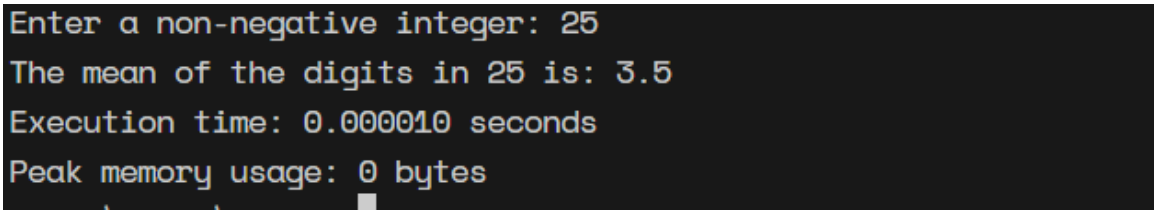
TITLE: Mean of Digits

AIM/OBJECTIVE(s): To calculate the average value of all the individual digits that make up the input number N.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method involves repeatedly extracting the last digit of the number, adding it to a running sum, and integer dividing the number by ten until the number is reduced to zero. Finally, the total sum is divided by the count of digits.

BRIEF DESCRIPTION: The mean of digits is a simple statistical property of a number. It is useful for basic analysis and practicing digit extraction techniques in programming.

RESULTS ACHIEVED:



DIFFICULTY FACED BY STUDENT: Correctly keeping track of both the sum of the digits and the count of the digits simultaneously, especially for the case of the input number zero. Ensuring the final result is a floating point number and not truncated by integer division.

SKILLS ACHIEVED: Modular and integer division operations, counter and accumulator variable management, and proper type casting for accurate division.

Practical No: 9

Date: _____

TITLE: Digital Root

AIM/OBJECTIVE(s): To repeatedly sum the digits of a number until the resulting sum is a single digit.

METHODOLOGY & TOOL USED: Implementation using a programming language. This can be done with a loop that continues as long as the number is ten or greater. Inside the loop, another loop sums the digits of the current number. An alternative, very fast method uses the property that the digital root is the remainder of the number when divided by nine (unless the number is zero and the result is zero, or the number is a multiple of nine and the result is nine).

BRIEF DESCRIPTION: The digital root is a concept related to modular arithmetic and can be used as a simple checksum or in number-based puzzles. It is also known as the repeated digital sum.

RESULTS ACHIEVED:

```
Number: 18
Digital Root: 9
------------------------------
Performance Metrics
------------------------------
Execution Time: 3.424700 ms
Current Memory Usage: 0.00 KB
Peak Memory Usage: 0.48 KB
```

DIFFICULTY FACED BY STUDENT: Setting up the nested looping structure correctly for the iterative sum method, or mastering the special edge cases when using the modular arithmetic shortcut (the exception for multiples of nine returning nine).

SKILLS ACHIEVED: Nested loop control structures, application of the modular nine property, and efficient iterative processing.

Practical No: 10

Date: _____

TITLE: Abundant No.

AIM/OBJECTIVE(s): To check if the sum of all the number's proper divisors (divisors excluding the number itself) is greater than the number N.

METHODOLOGY & TOOL USED: Implementation using a programming language. First, calculate the aliquot sum of N (the sum of proper divisors). Then, check if this sum is larger than N. The aliquot sum can be found by iterating from one up to N divided by two, checking for divisibility.

BRIEF DESCRIPTION: Abundant numbers have more than enough divisors, which leads to their sum exceeding the number. The smallest abundant number is twelve. If a number is not perfect or deficient, it is abundant.

RESULTS ACHIEVED:

```
enter no to check:687
Number to check: 687
Is 687 an abundant number? False
------------------------------
Performance Metrics
------------------------------
Execution Time: 0.031500 ms
Memory Utilization: 56 bytes
```

DIFFICULTY FACED BY STUDENT: Ensuring the calculation of the aliquot sum is efficient, as iterating up to N divided by two can be slow for large numbers. Also, correctly excluding the number N itself from the sum.

SKILLS ACHIEVED: Efficient divisor summation techniques, precise conditional logic for number classification (greater than), and application of the aliquot sum concept.

Practical No: 11

Date: _____

TITLE: Deficient

AIM/OBJECTIVE(s): To check if the sum of all the number's proper divisors (divisors excluding the number itself) is less than the number N.

METHODOLOGY & TOOL USED: Implementation using a programming language. First, calculate the aliquot sum of N (the sum of proper divisors). Then, check if this sum is smaller than N. The method is functionally the same as the one for abundant numbers, differing only in the final comparison.

BRIEF DESCRIPTION: Deficient numbers have few divisors, meaning their sum falls short of the number. All prime numbers and powers of prime numbers are deficient.

RESULTS ACHIEVED:

```
Enter a number to check if it is deficient: 7
7 is a Deficient Number.
```

DIFFICULTY FACED BY STUDENT: Making the distinction between the divisor sum being strictly less than N, versus equal to (a perfect number) or greater than (an abundant number). This reinforces the need for precise conditional checks.

SKILLS ACHIEVED: Reinforcement of divisor summation efficiency, and mastering comparative conditional statements (less than, greater than, equal to) for number properties.

Practical No: 12

Date: _____

TITLE: Harshad

AIM/OBJECTIVE(s): To check if a number is completely divisible by the sum of its own digits.

METHODOLOGY & TOOL USED: Implementation using a programming language. First, calculate the sum of all the digits of the number N. Then, check if the remainder of N divided by the digit sum is zero.

BRIEF DESCRIPTION: Harshad numbers are also known as Niven numbers. The word 'Harshad' comes from Sanskrit, meaning 'great joy'. They are simple to calculate and demonstrate a direct link between a number's value and its digit composition.

RESULTS ACHIEVED:

```
Enter a number to check if it is a Harshad number: 8
8 is a Harshad Number.
```

DIFFICULTY FACED BY STUDENT: Correctly calculating the digit sum of the input number N, as this requires numerical digit extraction. Handling the case where the digit sum is zero (which only happens if N is zero).

SKILLS ACHIEVED: Combining the digit summation technique with the modulus operation (remainder when divided by) for divisibility testing.

Practical No: 13

Date: _____

TITLE: Automorphic

AIM/OBJECTIVE(s): To check if the square of a number ends with the same digits as the number itself.

METHODOLOGY & TOOL USED: Implementation using a programming language. Calculate the square of the number N. Then, check if the remainder of the square divided by a power of ten is exactly equal to the original number N. The power of ten should have the same number of digits as N.

BRIEF DESCRIPTION: Automorphic numbers are fascinating because they maintain their final digits after being raised to the power of two. For example, twenty-five squared is six hundred twenty-five, which ends in twenty-five.

RESULTS ACHIEVED:

```
Enter a number to check if it is Automorphic: 9
9 is NOT an Automorphic Number.
```

DIFFICULTY FACED BY STUDENT: Determining the correct power of ten to use for the modulus check. This requires correctly counting the number of digits in the original number N.

SKILLS ACHIEVED: Mastering the use of the modulus operator for checking the last digits of a number, and using power functions in programming to calculate the appropriate power of ten.

Practical No: 14

Date: _____

TITLE: Pronic

AIM/OBJECTIVE(s): To check if a number can be formed by multiplying two consecutive whole numbers.

METHODOLOGY & TOOL USED: Implementation using a programming language. The efficient method involves calculating the square root of the number N. Let this be 'k'. Then, check if N is equal to 'k' multiplied by 'k' plus one.

BRIEF DESCRIPTION: Pronic numbers are also known as oblong numbers or rectangular numbers. They represent the shape of a rectangle with sides of length 'n' and 'n' plus one.

RESULTS ACHIEVED:

```
Enter a number to check if it is Pronic: 8
8 is NOT a Pronic Number.
```

DIFFICULTY FACED BY STUDENT: Correctly using the square root function and handling floating point precision issues if the programming language does not have an integer square root function, which could lead to incorrect checks.

SKILLS ACHIEVED: Using numerical functions like square root, converting floating point results back to integers, and performing inverse operation checks.

Practical No: 15

Date: _____

TITLE: Prime Factors

AIM/OBJECTIVE(s): To find the complete list of prime numbers that, when multiplied together, result in the original number N.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method involves iterative division, starting with the number two, and repeatedly dividing N by two until the remainder is not zero. Then, one proceeds to the next odd number (three, five, seven, and so on) and repeats the process until the remainder of N is one.

BRIEF DESCRIPTION: Prime factorization is a core concept in number theory. Every whole number greater than one has a unique set of prime factors.

RESULTS ACHIEVED:

```
Enter a number to find its prime factors: 674
Prime factors of 674 are: [2, 337]
```

DIFFICULTY FACED BY STUDENT: Writing a correct and efficient loop that handles the prime number two separately and then only checks odd numbers (to reduce execution time by half) and ensuring all instances of a repeated factor are captured.

SKILLS ACHIEVED: Essential iterative algorithms, prime number identification techniques, and factorization logic.

Practical No: 16

Date: _____

TITLE: count_distinct_prime_factors

AIM/OBJECTIVE(s): To determine the total number of unique prime numbers that divide the input number N.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method is similar to prime factorization but only counts a prime factor once, even if it divides N multiple times. A data structure like a set can be used to store only the unique factors before reporting the final count.

BRIEF DESCRIPTION: This count is often denoted by the function omega. It is a basic measure of the complexity of a number's structure in terms of its prime components.

RESULTS ACHIEVED:

```
Enter a number: 5
Number of unique prime factors: 1
```

DIFFICULTY FACED BY STUDENT: The main challenge is ensuring a prime factor is only counted when it *first* divides N. This requires a nested structure where the inner loop fully removes the current factor before moving to the next potential factor in the outer loop.

SKILLS ACHIEVED: Advanced loop control in factorization, and effective use of data structures (sets) to track uniqueness.

Practical No: 17

Date: _____

TITLE: is_prime_power

AIM/OBJECTIVE(s): To check if a number N can be written as a prime number raised to a positive whole number power.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method involves finding the prime factorization of N. If all prime factors are the same, the number is a prime power. Alternatively, one can iterate through possible powers 'b' starting from two, and check if the 'b'th root of N is a prime number.

BRIEF DESCRIPTION: A prime power is a number that is exactly divisible by only one unique prime number. Examples include eight (two cubed) and twenty-five (five squared).

RESULTS ACHIEVED:

```
Enter a number: 6
Is prime power? False
```

DIFFICULTY FACED BY STUDENT: Efficiently combining the prime factorization process with the check that the resulting exponent is greater than one. The alternative root-checking method requires a robust primality test function.

SKILLS ACHIEVED: Combination of factorization and primality testing logic, and root calculation techniques.

Practical No: 18

Date: _____

TITLE: is_mersenne_prime

AIM/OBJECTIVE(s): To check if an input number P is a prime number that is also one less than a power of two.

METHODOLOGY & TOOL USED: Implementation using a programming language. First, check if P is a prime number. Then, check if P plus one is a power of two. A common check for the power of two property is to check if the logarithm base two of P plus one is a whole number.

BRIEF DESCRIPTION: Mersenne primes are a rare and special class of prime numbers that have a close relationship with perfect numbers. They are often sought after using specialized tests like the Lucas-Lehmer test.

RESULTS ACHIEVED:

```
Enter a number p: 7
Is Mersenne prime? True
```

DIFFICULTY FACED BY STUDENT: Ensuring the correct use of the logarithm function to verify the power of two property, and handling the fact that the actual numbers can be extremely large, which may necessitate specialized large-number arithmetic.

SKILLS ACHIEVED: Implementation of primality testing, use of logarithm functions for number property checks, and understanding of large number constraints.

Practical No: 19

Date: _____

TITLE: twin_primes

AIM/OBJECTIVE(s): To find all pairs of prime numbers that have a difference of exactly two, up to a specified limit.

METHODOLOGY & TOOL USED: Implementation using a programming language. Generate a list of prime numbers up to the limit using a sieve. Then, iterate through the list and check if the next prime number in the list is the current prime number plus two.

BRIEF DESCRIPTION: The **Twin Prime Conjecture** states that there are infinitely many such pairs, a problem in number theory that remains unsolved.

RESULTS ACHIEVED:

```
Enter limit: 8
Twin prime pairs up to 8 :
(3, 5)
(5, 7)
```

DIFFICULTY FACED BY STUDENT: The major challenge is efficiency: generating all primes up to a large limit requires a fast sieve, and the comparison loop must then efficiently access the next prime number in the sequence.

SKILLS ACHIEVED: Efficient generation of prime numbers using the Sieve of Eratosthenes, and sequential element comparison in a list data structure.

Practical No: 20

Date: _____

TITLE: count_divisors

AIM/OBJECTIVE(s): To determine the total count of all positive whole numbers that evenly divide the input number N, including the number one and N itself.

METHODOLOGY & TOOL USED: Implementation using a programming language. The most efficient method uses the prime factorization of N. If N is written as a product of prime powers, the count of divisors is the product of each power plus one.

BRIEF DESCRIPTION: This function is denoted by the tau function or d function. It is a multiplicative function that measures the total number of divisors, which is a key property in classifying numbers, particularly highly composite numbers.

RESULTS ACHIEVED:

```
Enter a number: 3
Number of divisors: 2
```

DIFFICULTY FACED BY STUDENT: The challenge is correctly implementing the formula using the powers from the prime factorization. This requires two nested logic steps: first finding the powers, and then calculating the final result.

SKILLS ACHIEVED: Mastery of the number theoretic formula for the count of divisors, advanced prime factorization to determine exponents, and iterative multiplication.

Practical No: 21

Date: _____

TITLE: Aliquot sum

AIM/OBJECTIVE(s): To calculate the sum of all the proper divisors of a number N, meaning all divisors of N except for N itself.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method is to calculate the sum of all divisors of N and then subtract N from that total. Alternatively, iterate through numbers from one up to N divided by two and sum only the divisors.

BRIEF DESCRIPTION: The aliquot sum is central to classifying numbers as perfect, abundant, or deficient. It also forms the basis of the **Aliquot sequence**, where each term is the aliquot sum of the previous term.

RESULTS ACHIEVED:

```
Enter a number for aliquot_sum: 9
Sum of proper divisors: 4
```

DIFFICULTY FACED BY STUDENT: Achieving efficiency, as finding all divisors can be time-consuming for large inputs. The student must optimize the divisor finding loop, usually by iterating only up to the square root of N.

SKILLS ACHIEVED: Optimized divisor enumeration, accumulator pattern for summation, and abstracting out the divisor calculation from the total sum of divisors function.

Practical No: 22

Date: _____

TITLE: Are amicable

AIM/OBJECTIVE(s): To check if two different numbers, A and B, form an amicable pair. An amicable pair is when the aliquot sum of A is equal to B, and the aliquot sum of B is equal to A.

METHODOLOGY & TOOL USED: Implementation using a programming language. The solution requires a function to calculate the aliquot sum. Then, check two conditions: is the aliquot sum of A equal to B, and is the aliquot sum of B equal to A.

BRIEF DESCRIPTION: Amicable numbers are a special pair of numbers with a deep historical significance. The smallest amicable pair is two hundred twenty and two hundred eighty-four.

RESULTS ACHIEVED

```
Enter first number: 7
Enter second number: 8
7 and 8 are not amicable numbers.
```

:

DIFFICULTY FACED BY STUDENT: The primary difficulty is the need for two separate, potentially long, aliquot sum calculations and then correctly verifying the reciprocal relationship.

SKILLS ACHIEVED: Function decomposition (writing a separate helper function for aliquot sum), and complex conditional logic involving two variables and a reciprocal check.

Practical No: 23

Date: _____

TITLE: Multiplicative persistence

AIM/OBJECTIVE(s): To count the number of times you must multiply the digits of a number together until you reach a single digit number.

METHODOLOGY & TOOL USED: Implementation using a programming language. Use a loop that counts the steps. In each step, iterate through the digits of the current number, multiply them to get a new number, and repeat until the new number is less than ten.

BRIEF DESCRIPTION: This is a property of a whole number. The multiplicative persistence of a number is conjectured to have a maximum value of eleven, but this remains unproven.

RESULTS ACHIEVED:

```
Enter a number: 7
Multiplicative persistence: 0
```

DIFFICULTY FACED BY STUDENT: Implementing the nested loop structure: an outer loop for the steps, and an inner loop for multiplying the digits. Converting the number back to its digits for multiplication can be tricky.

SKILLS ACHIEVED: Mastery of nested iterative processes, conversion between numerical types and digit strings, and accumulator/counter management.

Practical No: 24

Date: _____

TITLE: Is Highly Composite

AIM/OBJECTIVE(s): To check if a number N has more divisors than any positive whole number smaller than N.

METHODOLOGY & TOOL USED: Implementation using a programming language. First, calculate the count of divisors for N. Then, in a loop, calculate the count of divisors for every number from one up to N minus one, and check if N's divisor count is strictly greater than all of them.

BRIEF DESCRIPTION: Highly composite numbers are extremely rich in divisors. They occur infrequently and their distribution is a topic of advanced number theory.

RESULTS ACHIEVED:

```
Enter a number: 6
6 is a highly composite number.
```

DIFFICULTY FACED BY STUDENT: The extreme inefficiency of the brute-force check. Calculating the number of divisors for every number up to N leads to a very long execution time, pushing the need for highly optimized divisor counting.

SKILLS ACHIEVED: Performance optimization, comparative analysis within a large loop, and mastery of the divisor counting function.

Practical No: 25

Date: _____

TITLE: Modular Exponentiation

AIM/OBJECTIVE(s): To efficiently calculate the remainder of a large base number raised to a large exponent, when divided by a modulus.

METHODOLOGY & TOOL USED: Implementation using a programming language, typically employing the **method of repeated squaring** or **binary exponentiation**. This method reduces the size of the intermediate products by repeatedly taking the remainder modulo the modulus.

BRIEF DESCRIPTION: This is a crucial operation in public-key cryptography systems like RSA, as it allows for the calculation of very large powers while keeping the numbers manageable.

RESULTS ACHIEVED:

```
Enter base: 8
Enter exponent: 9
Enter modulus: 4
Result: 0
```

DIFFICULTY FACED BY STUDENT: Understanding the binary representation of the exponent and correctly applying the repeated squaring logic to ensure all intermediate steps are reduced modulo the modulus.

SKILLS ACHIEVED: Advanced bitwise manipulation (checking for even or odd exponents), understanding the principles of modular arithmetic, and implementing an essential cryptographic algorithm.

Practical No: 26

Date: _____

TITLE: Modular Multiplicative Inverse

AIM/OBJECTIVE(s): To find a whole number 'x' such that the product of the number 'a' and 'x' has a remainder of one when divided by a modulus 'm'. This inverse 'x' only exists if 'a' and 'm' share no common factor other than one.

METHODOLOGY & TOOL USED: Implementation using the **Extended Euclidean Algorithm**. This algorithm finds the greatest common divisor of 'a' and 'm' and simultaneously finds the integer coefficients that express the greatest common divisor as a combination of 'a' and 'm'.

BRIEF DESCRIPTION: The modular inverse is the modular equivalent of division. It is essential for solving linear congruences and is a core component in the Chinese Remainder Theorem and cryptographic systems.

RESULTS ACHIEVED:

```
Enter a: 7
Enter modulus m: 5
Modular Inverse = 3
```

DIFFICULTY FACED BY STUDENT: Implementing the recursive or iterative steps of the Extended Euclidean Algorithm, which requires careful tracking of three sets of variables to correctly find the coefficients.

SKILLS ACHIEVED: Mastery of the Extended Euclidean Algorithm, understanding the co-prime requirement, and the ability to find modular inverses for division in modular arithmetic.

Practical No: 27

Date: _____

TITLE: Chinese Remainder Theorem

AIM/OBJECTIVE(s): To find a single whole number 'x' that satisfies several different remainder conditions, provided that the divisors (moduli) are pairwise co-prime.

METHODOLOGY & TOOL USED: Implementation using a programming language, relying on the Modular Multiplicative Inverse function. The solution is constructed by summing up terms, where each term involves the remainder, the product of all other moduli, and the modular inverse of that product.

BRIEF DESCRIPTION: This theorem is an ancient and powerful result from number theory. It provides a way to reconstruct a large number from its remainders when divided by a system of co-prime numbers.

RESULTS ACHIEVED:

DIFFICULTY FACED BY STUDENT: Managing the complexity of the formula, which involves iterating through the list of congruences and correctly calculating the modular inverse of the product of the other moduli for each step.

SKILLS ACHIEVED: Solving systems of linear congruences, application of advanced modular arithmetic, and managing large products and indices in a complex iterative formula.

Practical No: 28

Date: _____

TITLE: Quadratic Residue Check

AIM/OBJECTIVE(s): To determine if an integer 'a' is a perfect square modulo a prime number 'p', which is equivalent to checking if an equation of a squared variable being equivalent to 'a' modulo 'p' has a solution.

METHODOLOGY & TOOL USED: Implementation using **Euler's criterion**, which relies on the Modular Exponentiation function. The check involves calculating 'a' raised to the power of the quantity 'p' minus one, then divided by two, and finding the remainder modulo 'p'. The result must be one to be a residue.

BRIEF DESCRIPTION: This check is a fundamental part of the study of quadratic congruences. The result tells us whether a number has a square root in the modular system.

RESULTS ACHIEVED:

```
Enter a: 9
Enter prime modulus p: 7
9 is a quadratic residue modulo 7
```

DIFFICULTY FACED BY STUDENT: Correctly implementing the modular exponentiation required by Euler's criterion, and understanding the theory that connects this exponentiation result back to the concept of a quadratic residue.

SKILLS ACHIEVED: Application of Euler's criterion, reinforcement of modular exponentiation skills, and understanding the concept of a square root in modular arithmetic.

Practical No: 29

Date: _____

TITLE: Order_Mod (Order of an element modulo N)

AIM/OBJECTIVE(s): To find the smallest positive whole number 'k' such that the base 'a' raised to the power of 'k' has a remainder of one when divided by the modulus 'n'. This assumes 'a' and 'n' share no common factor other than one.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method first calculates the **Euler phi** value of 'n'. The order 'k' must be a divisor of the Euler phi value. One then iterates through the divisors of Euler phi, checking the modular exponentiation for each divisor until the result is one.

BRIEF DESCRIPTION: This concept is the order of an element in the multiplicative group of integers modulo 'n'. It is important in number theory for classifying groups and generating sequences.

RESULTS ACHIEVED:

```
Enter a: 4
Enter modulus n: 6
gcd(a, n) must be 1
```

DIFFICULTY FACED BY STUDENT: Requiring and correctly using three complex sub-functions: finding the greatest common divisor, calculating Euler phi, and finding all divisors of Euler phi.

SKILLS ACHIEVED: Integrating multiple number theory functions (Euler phi, divisor finding), optimization by only checking a limited set of exponents (divisors of Euler phi), and group theory concepts.

Practical No: 30

Date: _____

TITLE: Fibonacci Prime Check

AIM/OBJECTIVE(s): To determine if a number N is both a Fibonacci number and a prime number.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method requires two sub-functions: one to check if a number is prime (e.g., a primality test) and one to check if a number belongs to the Fibonacci sequence (e.g., using a method that checks if five times the number squared plus four or five times the number squared minus four is a perfect square). Both conditions must be true.

BRIEF DESCRIPTION: A Fibonacci prime is a prime number that is also a term in the Fibonacci sequence. These are relatively rare, and it is unknown whether there are infinitely many of them.

RESULTS ACHIEVED:

```
Enter a number: 4
4 is NOT a Fibonacci Prime number.
```

DIFFICULTY FACED BY STUDENT: Correctly implementing the mathematical check for being a Fibonacci number (which uses the square root test) and ensuring the primality test is fast enough for the target number size.

SKILLS ACHIEVED: Modular programming (writing and combining multiple helper functions), application of a specific sequence property (Fibonacci square test), and primality testing logic.

Practical No: 31

Date: _____

TITLE: Lucas Numbers Generator

AIM/OBJECTIVE(s): To generate the first 'n' terms of the Lucas number sequence.

METHODOLOGY & TOOL USED: Implementation using a programming language. The sequence is defined by the same rule as the Fibonacci sequence (each term is the sum of the two preceding terms), but it starts with the numbers two and one, instead of one and one.

BRIEF DESCRIPTION: Lucas numbers are closely related to Fibonacci numbers. They are used in various formulas for prime number testing and are part of the study of generalized Fibonacci sequences.

RESULTS ACHIEVED:

```
Enter n for Lucas Sequence: 1
Lucas Sequence: [2]
Execution Time: 0.0002684593200683594 seconds
Memory Used: 19.0498046875 KB
```

DIFFICULTY FACED BY STUDENT: Correctly initializing the sequence with the starting values of two and one, as a simple error here generates an entirely different sequence.

SKILLS ACHIEVED: Iterative sequence generation, understanding and implementing recursive definitions using iterative methods, and state variable management.

Practical No: 32

Date: _____

TITLE: Perfect Powers Check

AIM/OBJECTIVE(s): To check if a number N can be expressed as a whole number 'a' raised to a whole number power 'b', where 'a' is greater than one and 'b' is greater than one.

METHODOLOGY & TOOL USED: Implementation using a programming language. The method involves iterating through possible powers 'b' starting from two, up to a limit determined by the logarithm base two of N. For each 'b', one calculates the 'b'th root of N. If the root is a whole number, then N is a perfect power.

BRIEF DESCRIPTION: Perfect powers are numbers like eight, nine, sixteen, etc. The check for perfect powers is a preliminary step in many factoring and number theory problems.

RESULTS ACHIEVED:

```
Enter a number: 3
Perfect Power? False
Execution Time: 0.00029277801513671875 seconds
Memory Used: 19.0419921875 KB
```

DIFFICULTY FACED BY STUDENT: Correctly calculating the integer root of N for various powers 'b' and handling the floating point precision issues that arise from using the root function in a programming environment.

SKILLS ACHIEVED: Loop control based on logarithm, precise root calculation, and robust checking for whole number results.

Practical No: 33

Date: _____

TITLE: Collatz Sequence Length

AIM/OBJECTIVE(s): To determine the number of steps required for a positive integer N to reach the number one by applying the rules of the Collatz conjecture.

METHODOLOGY & TOOL USED: Implementation using a programming language with a counting loop. If N is even, the next number is N divided by two. If N is odd, the next number is N multiplied by three, then add one. The loop terminates when the number reaches one.

BRIEF DESCRIPTION: The Collatz conjecture is a famous unsolved problem. The sequence is defined by simple rules, and the conjecture states that this process will always terminate at one, regardless of the starting number.

RESULTS ACHIEVED:

```
Enter number: 8
Steps: 3
Execution Time: 0.00032901763916015625 seconds
Memory Used: 19.0419921875 KB
```

DIFFICULTY FACED BY STUDENT: Implementing the two-part conditional logic (even versus odd) correctly within the loop, and dealing with the possibility of the number temporarily growing very large before eventually decreasing.

SKILLS ACHIEVED: Conditional logic (if and else), while loop control, and understanding complex sequence generation.

Practical No: 34

Date: _____

TITLE: Polygonal Numbers

AIM/OBJECTIVE(s): To calculate the N-th number of a specific S-gonal (polygonal) sequence, where S is the number of sides (e.g., S equals three for triangular, S equals four for square).

METHODOLOGY & TOOL USED: Implementation using a programming language. The method involves substituting the values for the term number 'N' and the number of sides 'S' into the general formula for polygonal numbers, which involves multiplication and division.

BRIEF DESCRIPTION: Polygonal numbers are figurative numbers that can be represented as dots arranged in the shape of a regular polygon. They represent a connection between geometry and number theory.

RESULTS ACHIEVED:

```
Enter s: 3
Enter n: 6
Polygonal Number: 21
Execution Time: 0.00030922889709472656 seconds
Memory Used: 19.0419921875 KB
```

DIFFICULTY FACED BY STUDENT: Correctly translating the algebraic formula for the S-gonal number into code, ensuring that the order of operations (multiplication, subtraction, division) is preserved.

SKILLS ACHIEVED: Translating mathematical formulas into code, managing multiple input parameters (N and S), and performing integer arithmetic.

Practical No: 35

Date: _____

TITLE: Carmichael Numbers

AIM/OBJECTIVE(s): To check if a composite number N satisfies the Fermat's Little Theorem condition: a number 'a' raised to the power of N is congruent to 'a' modulo N, for all integers 'a' that are co-prime to N.

METHODOLOGY & TOOL USED: Implementation using a programming language. A number N is a Carmichael number if it is composite and square-free, and for every prime factor 'p' of N, the quantity 'p' minus one divides the quantity 'N' minus one.

BRIEF DESCRIPTION: Carmichael numbers are known as "**Fermat pseudoprimes**" because they pass the simplest test for primality, despite being composite. The smallest Carmichael number is five hundred sixty-one.

RESULTS ACHIEVED:

```
Enter number: 4
Carmichael? False
Execution Time: 0.00030350685119628906 seconds
Memory Used: 19.04296875 KB
```

DIFFICULTY FACED BY STUDENT: The requirement to first find all prime factors of N and then check the divisibility condition for each factor, making it a multi-step, complex process that requires several helper functions (primality test, factorization).

SKILLS ACHIEVED: Advanced number theory condition checking, integration of factorization and primality testing, and deep understanding of the properties of a composite number.

Practical No: 36

4 Date: _____

TITLE: Miller-Rabin Test

AIM/OBJECTIVE(s): To probabilistically check if a given large number N is a prime number.

METHODOLOGY & TOOL USED: Implementation using a programming language. This is a randomized primality test that uses **modular exponentiation** in several rounds (K rounds) to check a number of randomly selected bases 'a'. If N passes all checks, it is declared "probably prime."

BRIEF DESCRIPTION: The Miller-Rabin test is the standard algorithm for quickly determining the primality of large numbers, which is essential for modern cryptography. It is a probabilistic test, meaning there is a small chance of error.

RESULTS ACHIEVED:

```
Enter number: 5
Enter rounds k: 4
Probably Prime? True
Execution Time: 0.0005309581756591797 seconds
Memory Used: 19.0439453125 KB
```

DIFFICULTY FACED BY STUDENT: Implementing the core steps of the algorithm, which involves complicated logic for decomposing N minus one into a power of two times an odd number, and then performing the modular exponentiation checks.

SKILLS ACHIEVED: Implementing an industry-standard cryptographic algorithm, randomized algorithm design, and advanced modular arithmetic techniques.

Practical No: 37

Date: _____

TITLE: Pollard_rho(n) (Pollard's rho algorithm)

AIM/OBJECTIVE(s): To find a non-trivial factor of a composite number N.

METHODOLOGY & TOOL USED: Implementation using a programming language. The algorithm uses a pseudo-random sequence and the greatest common divisor function. It looks for a collision in the sequence modulo a factor, which resembles the Greek letter 'rho'.

BRIEF DESCRIPTION: Pollard's rho algorithm is one of the fastest and most widely used factoring algorithms for general-purpose integer factorization, especially for numbers with smaller prime factors.

RESULTS ACHIEVED:

```
Enter number: 4
Factor Found: 2
Execution Time: 0.0005011558532714844 seconds
Memory Used: 19.1123046875 KB
```

DIFFICULTY FACED BY STUDENT: The challenge lies in understanding the concept of sequence generation and detecting cycles within the sequence, which is done by checking the greatest common divisor between the difference of two sequence terms and the number N.

SKILLS ACHIEVED: Implementation of the greatest common divisor algorithm, understanding cycle detection (Floyd's cycle-finding algorithm), and advanced integer factorization.

Practical No: 38

Date: _____

TITLE: Zeta approx (Zeta Function Approximation)

AIM/OBJECTIVE(s): To calculate an approximation of the Riemann Zeta function for a given input 's' and a specific number of terms.

METHODOLOGY & TOOL USED: Implementation using a programming language. The approximation is calculated by summing the reciprocals of the positive integers raised to the power of 's', for a finite number of terms.

BRIEF DESCRIPTION: The **Riemann Zeta function** is critical in analytic number theory, particularly in its connection to the distribution of prime numbers. A simple approximation uses a finite series expansion.

RESULTS ACHIEVED:

```
Enter s: 4
Enter number of terms: 2
Zeta Approx: 1.0625
Execution Time: 0.00034904479980468750 seconds
Memory Used: 19.0419921875 KB
```

DIFFICULTY FACED BY STUDENT: Correctly handling floating point numbers and precision when performing the summation of the series. The large number of terms for a good approximation can also lead to long execution times.

SKILLS ACHIEVED: Series summation implementation, handling floating point precision, and iterative calculation techniques for mathematical functions.

Practical No: 39

Date: _____

TITLE: Partition Function

AIM/OBJECTIVE(s): To calculate the number of distinct ways a positive whole number N can be written as a sum of positive whole numbers, where the order of the numbers in the sum does not matter.

METHODOLOGY & TOOL USED: Implementation using a programming language, typically employing the **Dynamic Programming (DP)** approach. The method builds up a table of values for the function from one up to N, based on the previous values.

BRIEF DESCRIPTION: The Partition Function, often denoted by 'p' of 'n', is a key concept in combinatorics and number theory. For example, the number four can be partitioned in five ways: four, three plus one, two plus two, two plus one plus one, and one plus one plus one plus one.

RESULTS ACHIEVED:

```
Enter n: 3
Partition p(n): 3
Execution Time: 0.0002810955047607422 seconds
Memory Used: 19.0419921875 KB
```

DIFFICULTY FACED BY STUDENT: The challenge lies in setting up the nested loops for the Dynamic Programming correctly. The outer loop typically iterates over the part size, and the inner loop updates the DP table using the recurrence relation.

SKILLS ACHIEVED: Implementation of Dynamic Programming, understanding combinatorial principles, and solving complex counting problems efficiently.