Lab Manual

Practical and Skills Development

CERTIFICATE

THE ASSIGNMENT ENTERED IN THIS REPORT HAVE BEEN SATISFACTORILY PERFORMED BY

Registration No	:25BCY10120
Name of Student	:TULSI SHARMA
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:tulsi

VIT BHOPAL

Practical Index

S. No.	Title of Practical	Date of Submission	Signature of Faculty
1	Defined a function called aliquot_sum(n) that returns the sum of all proper divisors of n is less than n	16nov (sunday)	
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			

15			

Practical No: 1

Date: _16NOV_____

TITLE:Write a function called aliquot_sum(n) which checks whether the sum of proper divisors of n is greater than n .

AIM/OBJECTIVE(s): Wanted to define a function which checks whether the sum of proper divisors of n is greater than n .

METHODOLOGY & TOOL USED: Will check whether the given number gives remainder zero or not,if zero its proper divisor and will keep on doing this by increasing the i by 1.

BRIEF DESCRIPTION: Will check whether the given number gives remainder zero or not,if zero its proper divisor and will keep on doing this by increasing the i by 1.

RESULTS ACHIEVED:

```
A = int(input("enter a number: "))
total = 0
for i in range (1 , A):
    if A % i == 0:
     total += i

print(total)
```

DIFFICULTY FACED BY STUDENT:

CONCLUSION:


VIT
VELLORE INSTITUTE OF TECHNOLOGY
VIT
BHOPAL

Practical No: 2

Date: ____18NOV_____

TITLE:a)Write a function Modular Multiplicative

Inverse mod_inverse(a, m) that finds the number x such

that (a * x) ≡ 1 mod m.

b)Write a function Lucas Numbers

Generator lucas_sequence(n) that generates the first n

Lucas numbers (similar to Fibonacci but starts with 2,

1).

c)Implement the probabilistic Miller-Rabin

test is_prime_miller_rabin(n, k) with k rounds.

AIM/OBJECTIVE(s):a) find a function which can calculate inverse of two numbers.

b) to create a fibonacci but it should start with 2.

c) to find a function which can tell us wheter a number a prime,coprime or composite.

METHODOLOGY & TOOL USED:a) used greatest common divisor function to check wheter the number is factor the given number or not.

b)Defined the range from 2 as given in question and subtracted 1 from first term and 2 from second term.

c)First checked in particular range if the given number is prime or not thn checked for 2**n terms.

BRIEF DESCRIPTION:a) used greatest common divisor function to check wheter the number is factor the given number or not.

b)Defined the range from 2 as given in question and subtracted 1 from first term and 2 from second term.

c)First checked in particular range if the given number is prime or not thn checked for 2**n terms.

RESULTS ACHIEVED:a)def mod_inverse(a: int, m: int) -> int:

```
    if not isinstance(a, int) or not isinstance(m, int):
        raise TypeError("Both a and m must be integers.")
    if m <= 1:
        raise ValueError("Modulus m must be greater than 1.")


    def extended_gcd(x, y):
        if y == 0:
            return x, 1, 0
        gcd, x1, y1 = extended_gcd(y, x % y)
        return gcd, y1, x1 - (x // y) * y1
```

```
    gcd, x, _ = extended_gcd(a, m)


    if gcd != 1:


        raise ValueError(f'No modular inverse exists for a={a} and m={m}
(gcd={gcd}).")



    return x % m



try:
    print(mod_inverse(3, 15))  #  output: 4, since (3*4) % 11 == 1 or 3,15
no mod exist
    print(mod_inverse(10, 18)) #  output: 12, since (10*12) % 17 == 1 or
10,18 no mod exoist
except (ValueError, TypeError) as e:
    print("Error:", e)
b)def lucas_sequence(n):
    if not isinstance(n, int):
        raise TypeError("n must be an integer.")
    if n < 0:
        raise ValueError("n must be a non-negative integer.")


    if n == 0:
        return []
    elif n == 1:
        return [2]
    elif n == 2:
```

```
        return [2, 1]

    lucas_nums = [2, 1]



    for _ in range(2, n):
        lucas_nums.append(lucas_nums[-1] + lucas_nums[-2])

    return lucas_nums

if __name__ == "__main__":
    try:
        terms = 15

        print(f"First {terms} Lucas numbers:", lucas_sequence(terms))
    except (TypeError, ValueError) as e:
        print("Error:", e)
```

c)import random

```
def is_prime_miller_rabin(n: int, k: int = 5) -> bool:

    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False
```

```
    # Write n-1 as 2^r * d with d odd
    r, d = 0, n - 1
    while d % 2 == 0:
        r += 1
        d //= 2

    for _ in range(k):
        a = random.randrange(2, n - 1)
        x = pow(a, d, n)

        if x == 1 or x == n - 1:
            continue

        for _ in range(r - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False
    return True
if __name__ == "__main__":
    test_numbers = [2, 7, 88, 17, 19, 40, 563, 1235, 7019, 999983]
    for num in test_numbers:
        print(f"{num} -> {'Prime' if is_prime_miller_rabin(num, k=10) else
'Composite'}")
```

DIFFICULTY FACED BY STUDENT:faced difficulty while taking the range and finding how to check a number is composite or not.

CONCLUSION:a),b),c) found a function which can calculate the modulus, lucas numbers that start from 2 and a function which tells the number is prime composite or non prime.