**Write a program for synchronization of Threads using RLOCK. Accept the two numbers from user and calculate factorial of both Numbers simultaneously .**

import threading

# Creating a Reentrant Lock

lock = threading.RLock()

def factorial(number):

lock.acquire()

result = 1

for i in range(1, number + 1):

result \*= i

lock.release()

return result

def calculate\_factorial(number, name):

print(f"Thread {name} calculating factorial of {number}")

fact = factorial(number)

print(f"Thread {name} - Factorial of {number} is {fact}")

def main():

num1 = int(input("Enter first number: "))

num2 = int(input("Enter second number: "))

# Creating threads

thread1 = threading.Thread(target=calculate\_factorial, args=(num1, '1'))

thread2 = threading.Thread(target=calculate\_factorial, args=(num2, '2'))

# Starting threads

thread1.start()

thread2.start()

# Wait for both threads to complete

thread1.join()

thread2.join()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**49.\*\*Write a multithreading program where one thread prints square of a number and another thread prints factorial of a number. Also display the total time taken for the execution**.

import threading

import time

def square\_number(number):

print(f"Square of {number}: {number \*\* 2}")

def factorial(number):

fact = 1

for i in range(1, number + 1):

fact \*= i

print(f"Factorial of {number}: {fact}")

def main():

number = int(input("Enter a number: "))

# Record the start time

start\_time = time.time()

# Creating threads

thread1 = threading.Thread(target=square\_number, args=(number,))

thread2 = threading.Thread(target=factorial, args=(number,))

# Starting threads

thread1.start()

thread2.start()

# Wait for both threads to complete

thread1.join()

thread2.join()

# Record the end time

end\_time = time.time()

# Display total time taken

print(f"Total time taken: {end\_time - start\_time} seconds")

if \_\_name\_\_ == "\_\_main\_\_":

main()