**CC-311 Operating Systems LAB**

***LAB MANUAL # 7:***

***Roll\_no 068464***

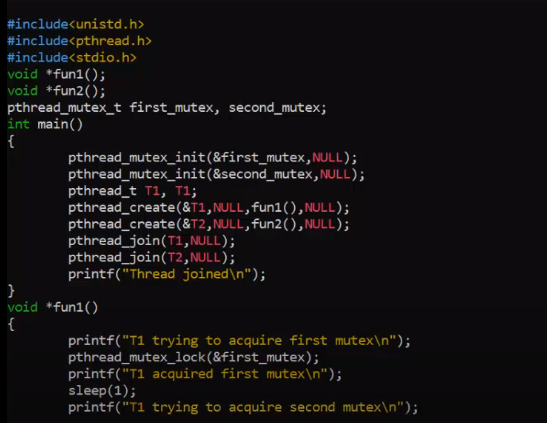
***Submitted \_by Nabila Naz***

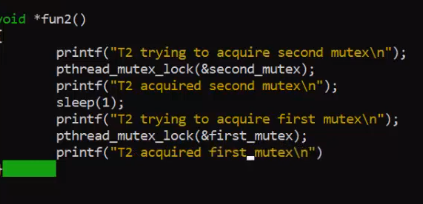
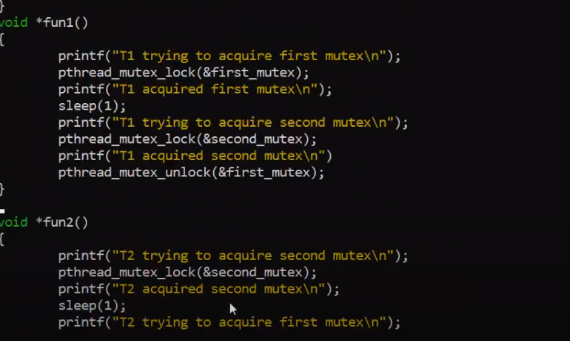
***Submitted To Sir Kamran .***

***Bs (IT) 5th(sem) Morning***

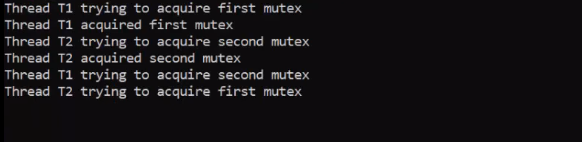
***“Simulate Deadlock “***

Q1. Write a program to simulate deadlock between three threads?

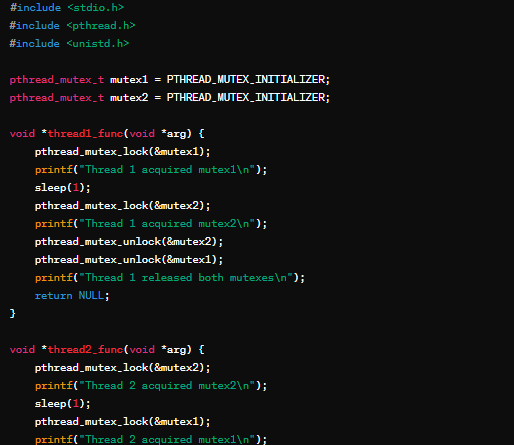


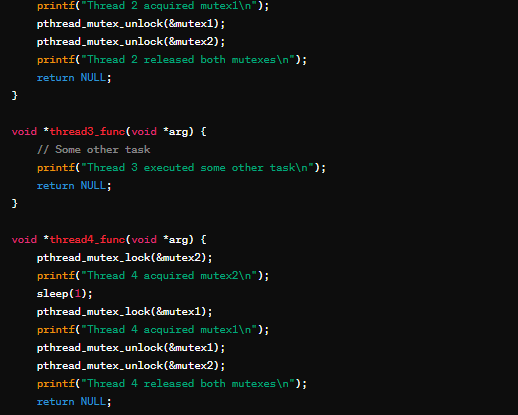


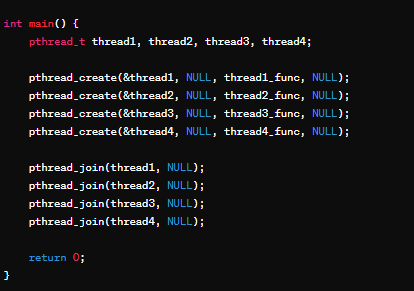
Compilation:



Q2.Write a program to create 4 threads (thread1, thread2, thread3 and thread4). Create a deadlock situation between thread2 and thread4?







**Compilation**



Viva questions

Q1. What is deadlock?

Deadlock is a situation in concurrent programming where two or more processes or threads are unable to proceed because each is waiting for the other to release a resource they need. This creates a circular waiting condition where none of the processes can continue, leading to a halt in the entire system.

Q2. What is the minimum number of threads/process required for deadlock to occur?

Deadlock can occur with a minimum of two threads or processes. However, it typically involves multiple resources being held and requested by different threads or processes. A deadlock situation arises when each thread/process holds one resource and waits for another resource held by another thread/process, forming a cycle of dependencies.

Q3. What is the significance of pthread\_mutex\_lock() function?

The pthread\_mutex\_lock() function is used in multithreaded programming to acquire ownership of a mutex (mutual exclusion object) before accessing a shared resource. It ensures that only one thread can execute a critical section of code at a time, preventing data corruption or race conditions. This function allows threads to synchronize their access to shared resources, ensuring mutual exclusion and preventing conflicts.

Q4. Why the sleep() function is used in the program?

The **sleep()** function is used to introduce a delay or pause in the execution of a program for a specified amount of time. It is often used in scenarios where timing or synchronization is required.

**For example:**

To simulate a waiting period or to control the rate of execution.To synchronize the timing between different threads or processes.

To allow other processes or threads to execute before continuing.

In testing scenarios to create specific timing conditions.

**sleep()** is particularly useful in situations where a process/thread needs to wait for certain conditions to be met before proceeding with its execution.