Operating Systems
LAB 7 Readme

Harshita Kalani (B20CS019)

1. Unzip the submitted file B20CS019_os_lab5.zip.
2. Open the file in any IDE
3. Compile the program using: **gcc -pthread <file_name> -o <exceutable_file>**
4. Execute Q1 which has two parts:
   With_deadlock - **g++ with_deadlock.c -o with_deadlock**
   ./with_deadlock
   No_deadlock - **g++ no_deadlock.c -o no_deadlock**
   ./no_deadlock
5. Run reader_writer problem with the following command:
   g++ reader_writer.c -o reader_writer
   ./reader_writer

Dining Philosophers Problem
We were instructed to implement the **Dining Philosophers Problem** in both its **deadlock-containing and deadlock-avoiding forms.** A typical synchronization problem is the dilemma of the dining philosophers.
Process Coordinating the execution of processes so that no two processes can access the same shared data and resources is the task of synchronization.

A deadlock occurs when two computer programmes that are using the same resource effectively block each other from using it, which causes both programmes to stop working.
In this scenario, if all philosophers are forbidden from eating, that indicates a stalemate issue. It might be resolved if each philosopher takes hold of their left fork first and works to free their right fork, but this is a stuck situation that won't resolve itself.

When the programme abruptly terminates, deadlock has occurred, and all philosophers would have reached for their left fork at that point. However, if the programme is restarted, it may run normally this time, indicating that deadlock hasn't occurred.

Reader_writers problem
A shared item, like a file, that is used by numerous processes is related to the **readers-writers** problem. Some of these processes are readers, meaning that they simply want to read data from the object, while some of these processes are writers, meaning that they want to write into the object.

To control synchronization and ensure that the object data is not affected, the readers-writers problem is used. As an illustration, there won't be any issues if two readers access the object simultaneously. The object may cause issues if two writers, or a reader and writer, access it simultaneously.

To resolve this issue, a writer should be granted exclusive access to an item, meaning that no readers or other writers should be able to access it while the writer is doing so. However, the item is accessible by numerous readers at once.
