Exercise 1 on Mutex :

Version 1

Two threads write on the screen 10 messages ("I am Process 0", "I am Process 1") in an alternate manner.

Version 2

Generalise the procedure with P threads.

Exercise 2 on Semaphores:

In this exercise, several variants of the producer/consumer problem are studied using Posix semaphores. It is supposed that several producers and consumers have to synchronise their access to a shared buffer.

Version 1

All messages exchanged by both types of processes are of the same type and they are produced and consumed in a sequential order in the buffer. This buffer may contain up to N messages and is handled in a *circular manner.*

☞ We suppose that the function `Insert(buffer, position, message)` is existing: It puts the *message* in the *buffer*, at position *position*. Similarly the function `Extract(buffer, position, message)` extracts the *message* at position *position* from the *buffer*.

☞ Write the code of one `Producer` process and one `Consumer` process; it is supposed that each process carries out a given number of operations `Insert` or `Extract`.

☞ Write a program in which N `Producers` and M `Consumers` communicate through the shared buffer (N and M are parameters of the program and may vary from an execution to the next one).

Version 2

Messages may be of two different types (Black/White, 0/1, …). One Producer is producing messages of one type only.

☞ `Producers` have to alternate messages types in the buffer. The type of the first message in the buffer is defined during the initialisation.

☞ `Consumers` consume the message also in an alternate manner.

Version 3

Messages may be of several different types (Black/White/Red/Green/…).

One Producer is producing messages of one type only. However, the Producers do not have to alternate messages in the buffer.

One consumer is consuming only messages of one type.

The messages must be inserted and consumed in their order of arrival.