

Complete these exercises and submit answers to these questions.

Make sure you attach a green assignment cover sheet to your answers.

For Q 1.2 - 1.4 submit the XCircaI code you added to the "SemaphoreQuestion1.2.xtc" example file. Indicate whether your code actually runs correctly (partial marks may be given for non-working solutions, no marks will be given for code that claims to work but doesn't).

The following XCircaI example files have been provided for this lab exercise:

SemaphoreQuestion1.2.xtc SemaphoreQuestion1.5.xtc

Q 1.1 (2 marks)

Draw behaviour diagrams (state machines) for the worker process and the semaphore process (processes "A" and "Sem") from the "SemaphoreQuestion1.2.xtc" example file.

Q 1.2 (1 mark)

Extend the XCircaI example file "SemaphoreQuestion1.2.xtc" to construct a system consisting of two worker processes (A and B) and the Semaphore, and to print out the behaviour of the system.

Q 1.3 (2 mark)

Define another version of your system process from Q 1.2, but with all events other than the critical section events hidden using the abstraction operator.

Q 1.4 (2 mark)

Use the CircaI System equivalence checker to show that the system constructed in 1.3 fulfills the mutual exclusion property.

Q 1.5 (2 marks)

Here is a definition of "fairness" for a semaphores:

If a process waits on a semaphore and is put to sleep
and
if the semaphore is signalled infinitely often
then
the process must eventually be woken.

Here is an extension of the worker process to handle a third worker process C (example file SemaphoreQuestion1.5.xtc) :

Process Sem, Empty, Full0, Full1, Full2, Wake0, Wake1

```
Empty      <- pA goA Full0 +
              pB goB Full0 +
              pC goC Full0 +
              vA Empty + vB Empty + vC Empty
Full0      <- pA sleepA Full1 +
              pB sleepB Full1 +
              pC sleepC Full1 +
              vA Empty + vB Empty + vC Empty
Full1      <- pA sleepA Full2 +
              pB sleepB Full2 +
              pC sleepC Full2 +
              vA Wake0 + vB Wake0 + vC Wake0
```

LabSheet1.txt

```
Full2    <- vA Wake1 + vB Wake1 + vC Wake1
Wake0    <- wakeA Full0 + wakeB Full0 + wakeC Full0
Wake1    <- wakeA Full1 + wakeB Full1 + wakeC Full1
```

```
Sem      <- Empty
```

According to the previous definition of fairness, this semaphore is unfair. Explain why. (in English - using diagrams or Circa if it helps you to explain your reasoning).

Q 1.6 (1 mark)

Lecture notes 4 gives a solution to the mutual exclusion problem using a test-and-set instruction (see Lecture6.xtc), while this exercises (SemaphoreQuestion1.2.xtc) gives a solution using a semaphore.

Which of these solution is a "spin-lock" ? What are some advantages and disadvantages of a spin-lock ?

Total: 10 marks