1. Implement a method to count the number of system ticks taken by the high priority

task within its critical section. How much time in seconds does it take to run the high

priority task critical section? (1 Mark)

1. ms
2. Which tasks are sharing the semaphore and which task pre-empts the low-priority task while it is holding the semaphore? (1 Mark)

The highest priority task and the lowest priority task share the semaphore.

The middle priority task pre-empts the low-priority task

1. A white board with colorful lines

   AI-generated content may be incorrect.Show your understanding of this problem by drawing a timing diagram highlighting each task and its status and when they switch contexts. See example diagrams in the lectures on the concurrent access model (1 Mark)
2. Which task is potentially at risk for missing a critical deadline and why? (1 Mark)

The highest priority task, because if a low priority task takes the semaphore and is then pre-empted by the medium priority task the high priority task will have to wait for both to finish before it can run

**Task B**

**1. Does the time taken by the high-priority task depend on the computation time for the**

**low or medium priority task? Why? (1 Mark)**

Task\_factor (0.1): 230ms

Task\_factor (0.5): 342ms

Task\_factor (1): 292ms

Task\_factor (2): 298ms

Task\_factor(4): 810ms

Task\_factor (8): 1835ms

Task\_factor (20): 4907ms

The time taken by the high-priority is increased as the medium or low priority tasks are increased. As it runs into priority inversion where if the low priority tasks locks the semaphore, the higher priority task has to wait for the medium task to finish which will then let the low priority finish and release the semaphore

**3. Show your results and show if more time was spent in the high priority critical section with the semaphore or with the mutex. Explain why this is the case? (1 Mark)**

(change in medium task)

Mutex high priority task delay\_ms,

* task\_factor(0.5) 184ms,
* task\_factor(1) 139, 208ms,
* task\_factor(2) 139, 190ms,
* task\_factor(4) 139, 95ms,
* task\_factor(8) 95ms

The time spent by the high priority task was drastically reduced by using a mutex instead of a semaphore, as the low priority task could now use priority inheritance through the mutex on the high priority task. However, when the low priority task was increased it still drastically affected the time taken, as even with inheritance it still was required to finish first.

**4. Show the output and explain what you observed. Which task is the priority inherited**

**from? (1 Mark)**

When the highest priority task tries to pre-empt the low priority task and take the mutex, the low priority task inherits the priority of the highest task that tries to pre-empt.

**5. Why does the medium priority task not pre-empt the high priority task when a mutex**

**is used as it did when a semaphore was used? (1 Mark)**

Because the low task inherits the high priority tasks priority.