Question 2

For this question, I implemented a C program that counts the number of numbers between 1-10,000 that are evenly divisible by 3 or 7, or by both. Posix threads were used for generating threads and semaphores for synchronization. The code was run and tested for both 8 threads and 16 threads. More information regarding how to run and outputs are provided in the readme and log file respectively.

The platform used for this question is **COE System**.

Number of numbers between 1-10,000 that are evenly divisible by 3 or 7, or by both is **4285**.

The time taken to run the program for 8 and 16 threads are provided below. These observations were averaged over 5 runs.

Number of threads	Time taken (millisecond)
8	0.523
16	0.741

Observation:

From the above table, the time taken to run the program using 16 threads is higher compared to 8 threads. Following factors play a role in the above observation.

- Time is spent in initialization of threads. There is an increase in time spent in initializing 16 threads compared to 8 threads.
- There is also time spent in synchronization of threads. With increase in the number of threads, time spent in synchronization also increases.
- As the input load (10,000 numbers) is low, there is little performance benefit observed in increasing the number of threads.

In summary, the increase in time spent in initialization and synchronization of threads from 8 to 16 overshadows the performance benefit obtained in parallelization of work load.

- With increase in workload, the performance benefit obtained in parallelization of workload overshadows increase in initialization time and synchronization time.
- Below table shows the reduction in time taken to run the program for an input size of $\mathbf{10^7}$

Largest Number	Number of threads	Time taken (milliseconds)
10^7	8	27
10^7	16	23
10^7	32	20
10^7	64	19