LOCK USAGE

Q1:

Hàm ***gettimeofday()*** trả về thời gian hiện tại tính theo giây (s) và micro giây (µs) kể từ thời điểm ***00:00:00, 01/01/1970***

Khoảng thời gian nhỏ nhất mà nó có thể đo chính xác được là 1 µs

Q1.c cung cấp một ví dụ căn bản về việc dùng hàm ***gettimeofday()*** để tính toán thời gian thực thi của chương trình

Q2:

Chương trình yêu cầu được hiện thực trong ***Q2.c***, với tham số truyền vào là số lượng thread của chương trình.

Mặc định, khi không có tham số truyền vào, chương trình sẽ thực thi với 1 thread (*./Q2*)

```
quockhanh@Khanh-VirtualBox:~/homework$ make Q2
gcc -pthread Q2.c -o Q2
quockhanh@Khanh-VirtualBox:~/homework$ ./Q2
counter = 1000000
Execution time: 0.015892 sec
```

Thực thi tiếp với số lượng thread lần lượt là 2, 3, 4:

```
quockhanh@Khanh-VirtualBox:~/homework$ ./Q2 2
counter = 2000000
Execution time: 0.106857 sec
```

```
quockhanh@Khanh-VirtualBox:~/homework$ ./Q2 3
counter = 3000000
Execution time: 0.136556 sec
```

```
quockhanh@Khanh-VirtualBox:~/homework$ ./Q2 4
counter = 4000000
Execution time: 0.170421 sec
```

Như đã thấy, việc tăng số lượng thread đồng thời cũng làm tăng thời gian thực thi của chương trình do để tránh xảy ra race condition, chỉ có duy nhất 1 thread được phép thực thi tại critical section ở một thời điểm

nhất định, các thread còn lại phải chờ đến lượt. Điều này làm tăng đáng kể thời gian thực thi của chương trình với nhiều thread.

Chương trình được chạy trên CPU với 6 lõi (6 core). Con số này không làm ảnh hưởng đáng kể đến thời gian thực thi của chương trình trên do các thread có thể chạy song song trên các lõi khác nhau, tuy nhiên phần lớn câu lệnh của các thread đều nằm trong critical section, do đó tại một thời điểm chỉ có 1 thread được thực thi các câu lệnh trong vùng này. Sự song song gần như không xảy ra.