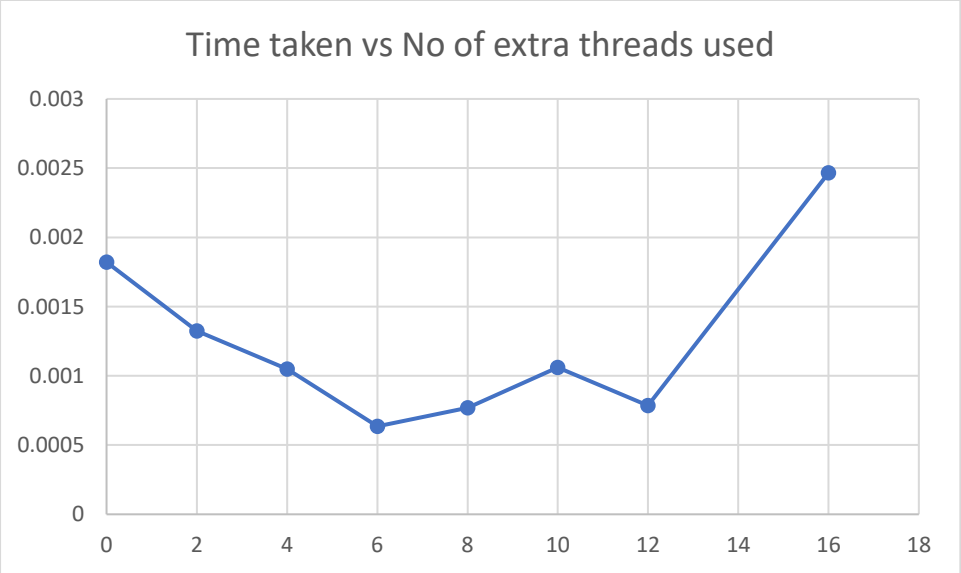
Q1.)

```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.00246574 seconds while using 16 threads.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.000786275 seconds while using 12 threads.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.00106063 seconds while using 10 threads.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.000768395 seconds while using 8 threads.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.000635366 seconds while using 6 threads.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.00104862 seconds while using 4 threads.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.00132401 seconds while using 2 threads.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$
```

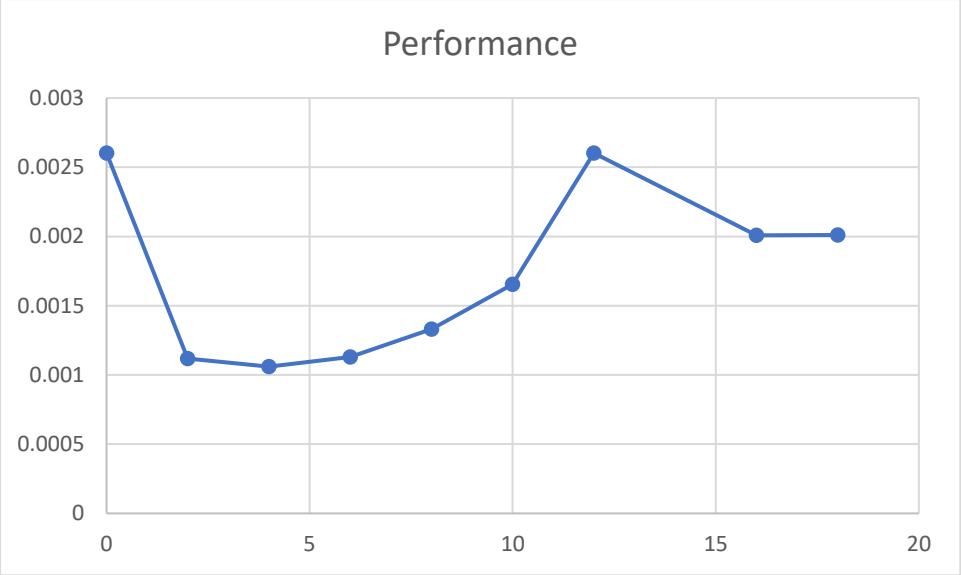
```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ1-ArnabMandal.cpp -o q1 -pthread
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q1
Total Sum: 500000500000 found in time: 0.00182235 seconds while processing Sequentially
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$
```



We observe that minimum time is needed when 6 extra threads are created and used. The processing time after this increases due to the increasing overhead and context switching.

Additionally at multiples of 6 number of threads, there is observed to be a relative boost in performance. This is likely due to CPU hyperthreading

Q2.)




```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00298051
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00201101 using 18 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00200788 using 16 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00260198 using 12 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00165439 using 10 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
^[[Aeros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00133178 using 8 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00112946 using 6 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00106006 using 4 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ^C

-bash: /q2: No such file or directory
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ /q2
-bash: /q2: No such file or directory
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Multithread sort is achieved in time: 0.00111878 using 2 threads
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q2
Sequential sort is achieved in time: 0.00260262
```

Performance is best while using 2-6 threads, with peak performance at 4 threads. After that, there are diminishing returns due to the increased overhead, explaining the later plateau in performance as well.

Q3.)

```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ3-ArnabMandal.cpp -o q3
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q3
Final Counter Value (without usage of lock): 100
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ3-ArnabMandal.cpp -o q3
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q3
Final Counter Value (without usage of lock): 1053794187
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$
```

Firstly, we observe the race condition only is observed for high values, during en masse repeated increments. For smaller values, it cannot be observed.

Here initially with usage of n=50, the actual and predicted value were same.
But with the usage of n=1000000000, the actual value predicted far from the expected 2000000000 with a different variation of value on each run.

```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q3
Final Counter Value: 2000000
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ g++ Assgn4SrcQ3-ArnabMandal.cpp -o q3
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$ ./q3
Final Counter Value: 1119912
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab4/Assgn4-ArnabMandal$
```

Using the mutex lock, with a value of n for 1000000 solves the race condition we found with higher values of n, by synchronizing the threads.

It should be noted that the lock increases the run time, as for the previous large value of n, the computation took a long time to finish.