Part A -

Sr Num	Time	Count
1	0.000912	74427
2	0.000803	88270
3	0.000886	78930
Avg	**0.000867**	**80542**

The final value of count doesn't matches the value of N*K in any run.
(line 15)
This is because count++ is not a atomic machine instruction. This results in the following race condition.
A thread loads the value of count and is then context switched out and the values is incremented by some other thread. This thread now store's the old value after incrementing which is clearly wrong

Part B -

Sr Num	Time	Count
1	0.001118	75135
2	0.001005	78721
3	0.000982	82520
Avg	**0.001035**	**78792**

The final value of count does not match the value of N*K in any run.
This is because this software based lock do not guarantee mutual exclusion. This can happen when a process gets context switched out after it has broken out of the while loop, but before setting lock to 1, and some other process in the meantime also breaks out of the while as lock is still 0 now we are in a similar situation as part a where more than 1 threads may start executing (line 19) count++ , hence resulting in race conditions as in part a.

Part C -

Sr Num	Time	Count
1	0.019662	100000
2	0.010943	100000
3	0.013023	100000
Avg	**0.014543**	**100000**

The final value of count matches the value of N*K in all runs.
This is because mutexes were used and the increments to count were done in a mutually exclusive fashion (unlike part A).