Group 1

Assignment 1

Week 1

1.1.1) No, it's close but missing around 30.000 (3,333%)

1.1.2)

a) Yes, it is very likely.
b) It is not guaranteed, but likely dependent on CPU speed. Since the number of missing counts does not seem to be based linear on count (as it would be 197ish) maybe it is because only a single thread is activated as the task of counting 200 is too fast to activate multiple Threads.

1.1.3) No we don't think it will. The issue is the fact the operation is not atomic, which is also true for the other semantics.

1.1.4) It locks the critical section (count++), preventing multiple Threads from executing it in random interleavings, to be executed sequentially by the Thread holding the lock, so each Tread execution has a happens-before relation.

1.1.5) Yes, it includes the least number of lines. It gets the value from the shared memory and increments it. If one part is outside 2 threads could read the same value creating a race condition.

1.2.2) Running the 2 Threads in parallel creates a race condition, where we can have random interleavings such as: T1(|), T2(|), T1(-), T2(-) -> “||--”

1.2.3) Using locks ensures an interleaving where 2 threads can never be in the critical section at the same time, eliminating the race condition. So we can never have -- or || as this would require both Threads to be in the print method at the same time. The only exception is if the method crashes after printing - but before printing |, as the other Thread would then also print -, creating “--”.

1.3.2) By using locks we ensure an interleaving where the 2 Threads can never be in the critical section at the same time, and the only way to have more than 15.000 is an interleaving where they both operate in the critical section. Fx if both Threads checks if the counter is under 15.000 and see the counter is 14.999, they could both increment the total counter to 15.001 depending on the interleaving of counter++.

Group 1

```
T1(read and check), T2(read and check), T1(increment), T2(increment)
```

So it will always be 15.000, as the threads run in happens-before relation.

1.4.1)

1a) Resource utilization and Inherent: No such thing as they overlap

1b) Fairness and Hidden: No such thing as they overlap

1c) Convenience: Android apps being able to navigate between each other to perform tasks, such as the share button taking the user to the social media app, instead of implementing this share functionality in the original app.

1d) Exploitation: Data base systems…

1.4.2)

Inherent:

- .setOnClickListener{...} (android)
- GUI thread/view model
- Tesla car

Exploitation:

- website server asynchronous call
- Premiere pro video encoding
- Games with no loading screen (game asset swapping)

Hidden:

- Phone storage
- Google Drive
- Websites