**Homework Wan Huzaifah bin Wan Azhar**

**Answer:**



* It points to line 8 and 15 as the code that can trigger data race, which is correct.
* It points out which variable is affected by data race.
* In this main-race, both threads try to access and write variable balance.
* As such, it is likely possible for each thread to increment balance to 1 and store 1 into balance instead of increments of 2.



* Removing one of offending code no longer trigger data race.
* When adding only one lock to protect the critical section (balance++), it does not change anything and helgrind still reports two data race error.
* This is because locking only one critical section will still trigger data race it depends on which thread get balance first.
* Adding locks on two thread will make data race disappear from helgrind analysis.



* The code will trigger deadlock because when first thread get m1 lock and second thread get m2 lock, both of the thread will wait for other to finish before they can get the next lock, m2 for 1st thread and m1 for 2nd thread.
* In the end, both thread cannot proceed and just waiting for each other.



* Helgrind reported lock order is violated.
* Which says that order is first m1 before m2, but another thread lock order is m2 before m1.
* Helgrind did not specifically reports deadlock however it reports the possibility of deadlock.



* It does not have the same problem as main-deadlock.c because it has global lock of g before a thread can acquire both lock.
* However, Helgrind still report the same problem as lock order is violated
* Helgrind function as stated in 4.



* This code is inefficient because the parent thread spends time doing nothing (busy waiting) until child thread signals that it is finished.
* As seen from (while done == 0)



* Helgrind report two possible data race
* One is printf
* Another one is shared variable done
* However, the program itself is correct as printf is printing first message before last message.



* Main-signal-cv.c is preferred because of both of performance.
* Previous program have the same correctness to main-signal-cv.
* Sleeping the thread until its condition variable change will improve the performance instead of wasting CPU cycle on busy waiting.



* Helgrind reports no error.
* This is because there is no shared variable to signal parents thread.