* Static Scheduler

Q1 The measurements seem erratic for a low value of 'n', but become less erratic as intensities increase. The creation of multiple threads and locking and unlocking them seems to increase the total work that is being done. This seems to be substantial amount of work for such problem.

Q2 As the intensities increase, the total work being done for locking the threads is overshadowed by the repeated calls made to the function due to the intensities.
However, as the intensities decrease, the total work being done for locking the threads is much more notable. Thus, speedup of low intensity runs with iteration-level synchronisation the way it is.

Q3 In the iteration-level-locking method, the shared resources are locked after each computation. Thus the shared resources are locked many more times comparatively.
In the thread-level-locking method, the shared resources are locked as many times as number of threads. Thus the shared resources are locked for less times comparatively.
Hence, the speedup for iteration level is lower.

* Dynamic Scheduler

Q1 Thread-level method and Chunk-level method perform quite similarly overall. Seems like thread-level method outperforms some times due to less times locking of the shared resources. Additionally, runs become very close to sequential when granularity tends to get closes to n. Hence in this case speedup decreases. and vice-versa.

Q2 Speedup increases when value of 'n' increases. But speedup tends to '1' when 'granularity' increases. Hence, speedup is more optimal for smaller values of granularity. It is because the distribution of tasks across the threads is even. And as the intensities increases, speedup becomes more optimal due to fewer locks.