Question 6 :

a. Why is threading useful on a single-core processor?

On a single -core processor, threading allows multiple instructions to be processed at a time.

b. Do more threads always mean better performance?

More threads do not always imply a better performance. If we have too many threads for a simple process, it could take more time. Or if a program is not designed to use multiple threads, having many threads will not improve the performance.

c. Is super-linear speedup possible? Explain why or why not.

Super-linear speedup is possible if the task given is embarrassingly parallelizable. As we split the data into smaller pieces, it is possible that the size of the data becomes small enough such that it fits into higher memory hierarchy component (e.g. from memory → L3 cache, or L3 cache → L2 cache etc), which often times the speeds up comes in chunks. Together with more cores, the speed up is possible to be super-linear.

d. Why are locks needed in a multi-threaded program?

Locks are abstractions provided by the operating systems to coordination shared resources across threads. They are used to avoid race-conditions in programs. For instance, if two threads increment the same variable, at the end it's not guaranteed that we will have the expected value. To avoid that, we have to use locks to control the access to the variable.

e. Would it make sense to limit the number of threads in a server process?

It makes sense to limit the number of threads in a server process because a processor core can have 2 threads at most normally. Introducing too many threads would incur a high overhead to the system, thus lowering the overall performance.