1. It depends. If the thread mapping it many-to-one, that means there is only one kernel per processor. There wouldn't be better performance in this case.
2. Line C would be 5 because it only runs the C_runner function which sets the value to 5. Line P would be 2 because the C_runner function isn't called and P_runner sets the value to 2.
3.
    a. When the number of kernel threads allocated to the program is less than the number of processing cores, each kernel thread would have its own core, but each kernel thread would have to balance more than 1 user thread, resulting in worse performance.
    b. When the number of kernel threads allocated to the program is equal to the number of processing cores, reach core would be used by a kernel thread, but it would only be marginally better as there would need to more than 1 user thread per kernel thread.
    c. When the number of kernel threads allocated to the program is greater than the number of processing cores but less than the number of user-level threads, the kernel threads would need to interleave. This could be more efficient, but not as efficient as a one-to-one mapping.