Lab 4

Question 2

- Barrier option is used when we want each thread to wait until all the other threads arrive to the same point. Since barrier is implicitly set at the end of a block, nowait clause can be used to remove the barrier.
- By enabling 'nowait' it's evident that the program acts differently. On the original version the first two threads being initialized wait for the others two before running the code. With nowait, the first two don't need to wait by others to run, i.e., they execute the code block while the others are "sleeping".
- When with barrier enabled and nowait disabled, the program acts as in the original form, since two of the threads will wait for the others that are sleeping, arriving all the 4 at the same time to the barrier.
- With barrier and nowait enabled, the program also acts as in the original form. Although, the nowait is enabled, the waiting procedure will be done in the barrier clause.

References:

1. https://www.intel.com/content/dam/www/public/apac/xa/en/pdfs/ssg/Programming_with_OpenMP-Linux.pdf
2. https://docs.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-clauses?view=msvc-170
3. https://www3.nd.edu/~zxu2/acms60212-40212-S12/Lec-11-02.pdf

Question 3

All the results were taken with input "25":

reduction clause*

- Check ex3-reduction.c

locks

- Check ex3-lock.c

atomic directive

- Check ex3-atomic.c

critical directive

- Check ex3-critical.c

Conclusion

- After measuring the 4 options the best to take would be atomic option. Although we need to take into account that the number used was very small, so, to a better comprehension a sort of tests would be recquired.

Question 4

To test the best option for multiplying 2 matrixes i've tested 3 scenarios.

Scenario 0: with no paralelization

Execution time: 7.922122s

Scenario 1: with paralelization in external loop

Execution time: 3.080599s

Scenario 2: with paralelization in external and internal loop

Execution time: 3.680599s

By this comparison of execution times I would choose scenario 1. Nevertheless more scenarios could be tested, for example using schedule(static)/schedule(dynamic).