1. We can also write a program to do matrix multiplication without using multithreading. What are the advantages and disadvantages of using multithreading? (5 points)

The advantages are that the work can be split among the cores and since results of one element in the resulting matrix do not depend on the others it is embarrassingly parallel.

The disadvantages of using multithreading are that you have an overhead for creating threads, and a wait time to synchronize the threads. If there isn't enough work these overheads could slow the program down.

2. Is your code still applicable for multiplying two 100-by-100 matrices? If yes, why? If no, how will you change your current code to make it applicable? You only need to write down the key changes and you can use pseudo code. (5 points)

I am taking this question to mean could my code provide a correct answer for multiplying two 100-by-100 matrices. The answer to that is yes, assuming A,B and M,N,K were all changed to match the values of those matrices.

3. Is your code efficient in the scenario of multiplying giant matrices, say one million by one million matrices? If yes, why? If not, what are the potential performance problems? (Hint: think about the overhead of creating thread and data locality issue). How to solve these problems? (10 points)

No. The overhead for creating that many threads would make the program very slow. To fix this instead of having 1 thread per element in the resulting matrix, a fixed number of threads should be responsible for a subsection of resulting elements.