HOMEWORK 3: PARALLEL PROGRAMMING WITH OPENMP (1)

DEADLINE: JANUARY 9

WRONG PROGRAM PARALLELISATION

1. Parallelise the outermost loop of the bubble sort algorithm using a **`parallel for pragma`**.
2. Run the parallel algorithm using a high number of threads.
3. Check the sorting results on a large randomly generated input array and explain.

OPENMP PARALLELISATION

1. Parallelise the following algorithms using OpenMP:
   a. Matrix multiplication.
   b. Brick sort (odd-even sort): https://en.wikipedia.org/wiki/Odd%E2%80%93even_sort.
   c. Bucket sort: http://en.wikipedia.org/wiki/Bucket_sort.
   d. Counting sort: http://en.wikipedia.org/wiki/Counting_sort.
   e. Insertion sort: http://en.wikipedia.org/wiki/Insertion_sort.
   f. Selection sort: http://en.wikipedia.org/wiki/Selection_sort.
2. Choose for each algorithm one large problem size (i.e., array dimension), initialise the array with uniformly distributed random numbers.
3. Execute the parallel algorithm on the `stud1.itec.aau.at` parallel machine using the `Slurm` workload manager with 1, 2, 4 and 8 parallel threads.
4. Report the execution time, speedup, and efficiency metrics in a simple PDF file.

IMPORTANT REQUIREMENTS

1. Measure only the core execution time of each algorithm without random number generation, array/matrix initialisation, and any I/O operations.
2. Declare the main array or matrix data structures as static global variables. Do not use dynamic memory allocation using `malloc`.
3. Implement the algorithms as short (few lines) and as simple as possible, focused on the core functionality (e.g., without safety checks, small optimisations, or redundant tests).