HOMEWORK 1: SEQUENTIAL ALGORITHMS

DEADLINE: OCTOBER 24

1. Implement in the C programming language the following algorithms:
   a. Matrix multiplication;
   b. Gaussian elimination algorithm for solving a linear system of equations assuming that the pivot element cannot be zero: http://en.wikipedia.org/wiki/Gaussian_elimination;
   c. Dijkstra's algorithm for computing the shortest path between nodes in a graph stored as an matrix: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm;
   d. Sieve of Eratosthenes for finding all prime numbers to a given limit: https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes;
2. Implement in the C programming language the following sorting algorithms:
   a. Bubble sort: http://en.wikipedia.org/wiki/Bubble_sort;
   b. Bucket sort: http://en.wikipedia.org/wiki/Bucket_sort;
   c. Counting sort: http://en.wikipedia.org/wiki/Counting_sort;
   d. Insertion sort: http://en.wikipedia.org/wiki/Insertion_sort;
   e. Selection sort: http://en.wikipedia.org/wiki/Selection_sort;
   f. Quick sort: https://en.wikipedia.org/wiki/Quicksort.
3. Initialise the algorithms with uniformly distributed random numbers;
4. Choose for each algorithm one large problem size (i.e. array dimension) and execute it on the `stud1.itec.aau.at` parallel machine using the `Slurm` workload manager;
5. Use the GNU `gprof` profiler to measure the execution time of each algorithm;
6. Report and explain the results 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).