Tutorial 7

Problem 1
Write a subroutine creates a matrix consisting of NxN floating point values, with a given size N. The values shall be randomly distributed in the range of [-1;1].

Store the matrix in a one-dimensional array in row-major order. This means, the array index of a matrix element at a given row and column (col) shall be given as:

\[ \text{index} = \text{row} \times N + \text{col} \]

Problem 2
Write a subroutine that multiplies two matrices of size NxN.

(Note: \( c_{ik} = \sum_j a_{ij} \times b_{jk} \))

If you code in C, the function declaration might look like this:

```c
void mtxmul (int n, float * c, float * a, float * b);
```

Problem 3
Measure the time that it takes to multiply two matrices on your computer (or maybe various different computers), dependent on the size N.

In order to obtain meaningful results, it is best to perform several rounds of computation not measuring the time. The first times are much slower, until the program code is allocated inside the cache.

In order to study the effect of the data cache, use a set of operand matrices so that they cannot come from the cache.

Determine the number of MFLOPS (million floating-point operations per second), depending on size N.

(Note: from the formula in Problem 2, it can be seen that the computation of each of the NxN elements in the result matrix \( c_{ik} \) requires N multiplications and N-1 additions. So, in total, the matrix multiplication requires \( 2N^3-N^2 \) floating-point operations.)

Do not forget to document the configuration of hardware and software with which you obtained the results.

Problem 4
a) Rewrite the problem to take care of cache architecture of your system.
b) Use prefetch primitives to preload data into caches.