Parallel computing 12.3.2018
Exercise 8

Last lecture at Tue 27.2., a tutoring session at Fri 2.3. 10-12, these last exercises at **Monday 12.3. 14-16**. Course exam at Fri 16.3. 12:00-16:00, Joensuu: C2, Kuopio MD100, please register through WebOodi.

Submit the X3 task by Monday 12.3. 09:00 to Moodle. Submit each of the other solutions separately to Moodle by 12.3. 12:00 (2 hours before exercises). Take skeletons from course www-page.

The following X3 exercise replaces standard exercises 36 and 37.

The answers to X-exercises have to be unique for every student. No copies of the same answer are allowed. The answer has to be submitted to Moodle by Monday 09:00. Answers will be graded. The answer must also contain a short self-evaluation in which you describe whether the program works, nearly works, or does not probably work; how efficient it is, what speedups you got, etc. A correct and proper self-evaluation is worth one point (in case of a proper answer). As the answer is a C program, the self-evaluation must be included in the comments of the program.

The solution should be a compilable source code file with needed additional files (Makefile, etc.).

Take a skeleton from course www-page. Do not change the header (name, parameters) of the X-task method(s). Please make sure that the program is compilable as such.

X3. Parallelize the prime-finding program using CUDA (max 6p), OpenCL (max 6p), pthreads (max 4p), C++11 threads (max 4p), C11 threads (max 4p), Java threads (max 4p), (or OpenACC (max 4p), to be confirmed). Running time will impact the grading. If you wish, you can also just return the number of primes without returning the actual primes – this version gives you 2 points less.

38-39. Parallelize the prime finding algorithm using CUDA if you did't use it in X3. If you used CUDA, then using OpenCL, pthreads, C/C++11/Java threads, or Fortran.

40. Write an algorithm to count votes using Fortran 90. Input is an N-element array of integer elements, each from 0..M (vote for number) or -1 (empty vote). Result is the number having most votes and the number of votes. If there are several numbers with equal number of votes, print them all. Use vector operations where possible.