Homework 4, due: 02/24

MATH 9830, Spring 2015 Timo Heister, heister@clemson.edu

0. Before you start:
   - Read the description of tutorial step-4, optionally watch the linked videos.
   - Familiarize yourself with `04_threads_hello` and `05_threads_ex1` and run them on your computer.
   - Submit your code via email. Print all output mentioned in programming exercises.
1. Think of an example in real life (like the "digging a hole" example in class) of a task that can be done using a certain number of workers N. Identify a serial part, a perfectly parallel part, and a third part that grows proportional with N.
   (a) Ignoring the third part, apply Amdahl's law and determine the maximum speedup for your example (make up numbers as you go).
   (b) Now include the third part and determine the maximum speedup (what value of N gives it)?
2. step-4
   (a) Write a member function `void mesh_info()` that prints the following information about the triangulation to the screen: 1) number of active cells, 2) number of active/used vertices, lines, quads, hexs (only if appropriate for the dimension!).
   (b) Use `VectorTools::compute_mean_value` (see step-3) and verify the convergence order of the mean in 2d and 3d.
   (c) Change the mesh to an L-shape, only apply boundary values to the faces adjacent to the center (see `set_boundary_indicator()` in the step-3 description), change the boundary values to be $1+\|x\|$ and the right-hand side to be 1. Finally, learn how to visualize your solutions in ParaView (or VisIt) and generate a 2d and a 3d picture (print or email).
3. Multithreading
   (a) Determine the number of (virtual) cores in your machine (hint: `/proc/cpuinfo`). Figure out if your machine uses hyperthreading.
   (b) Implement multithreaded vector addition based on `06_threads_ex2` and find the optimum number of threads for your machine (try anything between 1 and twice the number of cores in your system, the commandline tool `time` might help).