Assignments P1.9 - Part 2

N-Body LJMD Refactoring, Optimization and Parallelization Project

You will work in this assignment in small groups with 3 or 4 members. The evaluation will be based in part (1/3rd) on the performance of the group as a total and also based on the individual contributions as evidenced by the commits to the project (2/3rd). Please add a list of the group members and how their names will show up in the commit messages to the README file, so that your commits will be properly attributed.

Tasks for the entire group (to be completed within the first ½ day)

Set up a **shared** git repository using the code from https://bitbucket.org/akohlmey/ljmd-c.git so that all members of your group have write access (i.e. can push into this repository). Then each member clones this repository. You will work in your local repository on temporary branches, merge them into your local master when done and push your changes (after pulling changes from other group members, if needed).

- break down the single file ljmd.c into multiple files (force compute, verlet time integration (split into two parts), input, output, utilities, header for data structures, prototypes); update the make process accordingly so that dependencies are properly applied.
- set up some simple unit tests (write C code that fills data structures, calls the respective functions and outputs the changed data. Compare that against reference) For example:
  a) compute force for a few 2- or 3 particle systems with atoms inside/outside the cutoff
  b) compute one full step time integration for given forces and velocities (no call to force())
  c) compute kinetic energy for given velocities and mass
  d) verify that input parameter data is read correctly.

Tasks for **individual** group members (to be completed by friday)

a) Create a python interface so that top-level operations are implemented in python and only time critical steps are performed in C. This can be done incrementally. Minimal goal is to replace input (of parameters) and do unit testing in python.
b) Optimize the force computation: refactor the code for better optimization and to avoid costly operations or redundant work. Adapt data structures as needed. Document improvements with profiling/benchmark data.
c) Add MPI or OpenMP parallelization. Document the parallel efficiency of changes.
d) (for 4-member groups) Add second parallelization method.
It is crucial that the individual tasks are implemented in small self-contained steps, so that they can be merged into the master branch frequently. Do not wait until it is too late.

Bonus tasks for extra credit (for individuals or pairs)

1) Implement additional force function using Morse potential instead of Lennard-Jones
2) Implement cell list for better scaling with system size
3) Implement force kernel with CUDA/OpenCL/SIMD-vector intrinsics.