Programming Techniques II
Exercise 11

Task 11.1 Penna HDF5

In order to run large sweeps and create millions of datasets, we want to change the output format from text files to HDF5. Most languages have HDF5 libraries and support, so it is an ideal format to comunicate between different programs. You can find multiple examples of HDF5 here:
(http://www.hdfgroup.org/HDF5/examples/intro.html)
Add a switch to your PennaLV simulation to allow choosing text files or HDF5 archives as output. You need to install libhdf5 and link against it. We recommend using the C-API.

Task 11.2 Ackerman / boost::serialization

On the repository (exercise/ackerman) you can find an implementation of the Ackerman function, which scales tetrationally (faster than exponential) and makes heavy use of recursion. To speed up the calculation, we want you to come up with a way to cache previous results s.t. they don't need to be recomputed every time.
Since we do not want to recalculate the cached values every time we start the program, use boost::serialization to save the cache at the end of the run and read it at the start. You have to install the boost headers and the libboost_serialization and link against it. If you get an invalid signature runtime error in your implementation, just delete the archive file you generated. It happens every time you change the serialization layout.

Task 11.3 Functional Programming

Pick one of the following problems and implement a solution in a functional programming language of your choice. This language must be installable/runnable with reasonable effort on a Ubuntu 14.04 system (if the process is non-obvious, document it).
Let `L` be a list of 200 randomly generated integers. You may create the list with another program and hard-code (paste) it into your codes, as long as they work correctly for arbitrarily modified lists.

1. Find the maximum value of `L`.
2. Sort `L` in descending order.
3. Check if a manually provided value is in `L`.
4. Compute the average of `L` without using built-ins.
5. Remove the element at a manually provided index in `L` (return a new list).
6. Print every second element of `L`.
7. Compute the dot product of two different randomisations of `L`.
8. Calculate the first 100 numbers of the Fibonacci sequence $F_n = f_{n-1} + F_{n-2}$:
   0 1 1 2 3 5 8 13 21 ...
9. Check if a manually provided number is prime.
10. Compute the square root of a manually provided real number approximatively without using built-ins.

Task 11.4 Functional Marathon Challenge (optional)

Repeat the previous Functional Programming task. Don't pick the same problem twice. Don't pick the same language twice.
The highest count of correct solutions wins.
Since we are doing functional programming, elegance will be rewarded. The problems are fairly simple, we highly recommend to not just copy solutions from the internet.
The submission deadline for this challenge is Wednesday 2.12.15 at 05:00 in the morning. Notify us via the mailing list (pt2_hs15_ta@lists.phys.ethz.ch) with a link to your uploaded solution on your PT2 repository on GitLab.