# Programming Techniques II
# Exercise 6



### Task 6.1 PennaLV

- The simulation class is now hardcoded on sheep and bears. Change it into a template that takes two arbitrary types `T1, T2` (which happen to behave like sheep and bears) and remove all occurences of `sheep` and `bear` from the simulation class hpp/cpp.
- We have chosen `std::list` as a natural container for the animal population, but now we want to investigate alternatives. Create a feature branch (using the `git branch` command) where you replace `std::list` with `std::vector`. Since vectors have different performance characteristics from lists, here are a few hints:
  - don't delete dead animals, just mark them (insertion/deletion in vectors is very expensive)
  - you may want to use a separate vector to store indices of dead animals
  - new animals should replace the earliest (lowest index) occurence of a dead animal (`push_back` if no space)
  - animals should not be updated (`progress`) in the iteration they are born
- Compare the performance of your branches by inserting MiB statements as shown in the lecture. Since the changes described alter the numerics of your simulation (statistics should be equivalent in the long run), you will have to relax some unit tests (e.g. final population counts).

### Task 6.2 Forward Challenge (optional)

Implement `challenge::forward` yourself without using anything from the `std` namespace (directly or indirectly). The challenge code can be found in the lecture repo. You will learn more if you don't copy the `std::forward` implementation of your libc.
The submission deadline for this challenge is Wednesday 28.10.15 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.

### Additional Notes:

The *lecture* repository holds the discussions we had on tools (git, cmake, catch, gdb/lldb) in the *wiki* directory (best viewed on GitLab), and the corresponding examples in the *tools* directory.