Cartesian Vectors Variable

This exercise is somewhat similar to the one of last session. It is also concerned with the implementation of a vector like class. The main difference is that now the dimension of the vector is no longer known at compile time. This means that your class is a RAII class, a class which holds some ressource, in this case memory. In order to teach you good C++ reflexes you must use a suitable smart_pointer in order to internally manage the resource.

Important

Operations (+/-/*) between Vectors of different size are forbidden and must throw a std::runtime_error with the message "Incompatible size".

Restrictions

You are not allowed to use std::array, std::vector, std::list, etc (Anything that is already a container) You are not allowed to use new/new[] (delete/delete[]) in your code. (Do not even think about using malloc/free!!!)

Hints

std::initializer_list is a somewhat weird construct. You can imagine it like a wrapper around an array constructed from a braced list of values, like {1, 2, 3}. To access the underlying array, you should can std::data.

Challenge

Implementations faster than our (suboptimal but not bad) reference implementation get extra points.

Bonus

Another bonus is given to an implementation if it is at least 10% faster than any of the other implementations (including our reference implem)