CPU Algorithm Design

Exercise 3 **Students:** Vishal Mangukiya, Konstantin Benz

3.1 Adapting reduce and transform

The input containers in `reduce_LoopUnrolling_view.hpp` and `transform_LoopUnrolling_view.hpp` have been adapted as requested. For the reduction routines, `std::views::repeat(1.0f, N)` is used to replace the original memory-backed containers. This ensures that the workload is compute-bound rather than memory-bound. For the transform routines, `std::ranges::views::iota(0, N)` is used for the input range, and the output container `W` is a fixed-size `std::vector<Real>(256)` with modulo indexing.
All adapted benchmark functions were successfully compiled and tested using the executables `reduceVbenchmarkUnroll` and `transformVbenchmarkUnroll` on the target system.

3.2 Adapting benchTransformUnrollLoopPeelingDirective

3.3 Adapting benchReduceUnrollTreeDirective

3.4 Adapting benchReduceUnrollSimdXHorizontal and benchReduceUnrollSimdXVertical

3.5 Benchmarking