Exercise sheet 9 2023-01-12

Due date: 2023-01-19 16:59

The goal of this exercise sheet is to get you used to templates, classes, and move semantics in C++ .

In this exercise we will create a simple templated `Vector<T>` class similar to `std::vector<T>`. To avoid explicit memory management we will be using smart pointers (`std::unique_ptr<T[]>`).

Templates are implemented in header files. Complete the implementation of `Vector<T>` in `vector.h`.

- Constructors
- Copy and Move Assignment
- `calculate_capacity`: returns the required `capacity` given a `new_size`
  - current capacity is zero: requires `new_size` capacity
  - current capacity is large enough for `new_size`: no resizing necessary
  - current capacity is too small: double the capacity to fit `new_size`
    When in doubt, compare your size and capacity with std::vector's.
- `resize`: Resizes the vector to `new_capacity`. If resizing is necessary moves all elements to a new array.
- `push_back`:
  If adding an element to the vector would exceed its capacity we double (`growth_factor`) the vector's capacity.
- `pop_back`: For simplicity, removing elements does not reduce the capacity.
- Access using `operator[]` or `at` with bounds-checking. Throw an `std::out_of_range` exception if the requested position is out of range.