Assignment 3

3-0-1

ForwardIterator vs. RandomAccessIterator

Pointer arithmetic is not possible with Forward Iterators. Also, only post and preincrement ++ are available. RAIs have valid decrement operations and give random access to the underlying data - operator[];

InputIterator vs. OutputIterator

InputIterator only lets us read the dereferenced value. - std::iterator_traits::value_type == const smth OutputIterator lets us write to the dereferenced value.

3-0-2

No questions

3-1-1

Unter `gottschlin_list.h` zu finden. `iterator_traits` nachträglich hinzugefügt.

3-1-2

Alle tests aus der Vector test suite sind erfolgreich gelaufen. In den letzten Zeilen von `list_test.cpp` auch `insert_after` - falls dies nicht die richtige Funktionsweise ist, sehr froh auf Hinweise...

3-2

No questions. Tests passed.

Assignment 4

4-0

No questions.

Talks watched, enjoyed both.

4-1-1

No questions.

Implementation is more efficient for RAIs because of the use of std::advance().
https://en.cppreference.com/w/cpp/iterator/advance

> Complexity
>
> linear.
>
> However, if InputIt additionally meets the requirements of RandomAccessIterator, complexity is constant.

4-1-2

No questions.

4-2-1

No questions.

Still working on it.

4-2-2

No questions.

Solved after hints in e-mail exchange.

Assignment 5

5-1-0

No questions.

Watched both talks, NRVO sticked after the first one. Small string optimization after the second one. Andrei Alxandrescu's talks are way more entertaining.

5-1-1

No questions.

Included move constructor and assignment to `gottschling_list.h` and godbolt link with test.

5-1-2

No questions.

Sparse array has no ownership, default move semantics work. Will only be able to validate with working `sparse_array.h`

5-2

Not being color-blind makes it possible to see if move semantics work as expected, poor dogs.