Exercises:

1. Create a UDT that can not be copied. Trying to copy an object of that UDT should result in a compile time error.
2. Explain in one sentence: When are two objects equivalent?
3. Create a UDT *Complex* (http://en.wikipedia.org/wiki/Complex_number and https://www.nussschale-podcast.de/imaginaere-zahlen-ep013) that implements the reasonable essential operators as well as all the reasonable arithmetic operators. Add a couple of tests.
4. Based on the overloaded operators in *Complex*, prove that the operator+ is left associative and that the operator= is right associative in C++. Document how you proved it.
5. Revisit the type *DynamicIntArray* and implement the big three.
6. Revisit the type *DynamicIntArray* and implement operator[] to mimic the behavior of an array exactly.
   a) You will have to provide two overloads of operator[]! – Why?
7. Revisit the type *DynamicIntArray* and implement operator+ and operator+=. The operators shall create a new *DynamicIntArray,* which just contains all elements of the source objects.
8. Revisit the type *DynamicIntArray* and implement operator<< for stream output.
   a) Also show how it works with file streams!
   b) Also show how it works with string streams (more exactly *std::ostringstream*)! – When are strings streams useful?
9. Write an operator* for *std::string*, that turns the expression *std::string("xy")* * 3 into *std::string("xyxyxy")*.

Remarks:

- Everything that was left unspecified can be solved as you prefer.
- In order to solve the exercises, only use known constructs, esp. the stuff you have learned in the lectures!
- The usage of goto, C++11 extensions, as well as #pragmas is not allowed. The usage of global variables is not allowed.
- **Please obey these rules for the time being:**
  - **The usage of goto, C++11 extensions, as well as #pragmas is not allowed.**
  - **The usage of global variables is not allowed.**
  - **You mustn't use the STL, because we did not yet understood how it works!**
  - **But *std::string*, *std::cout*, *std::cin* and belonging to manipulators can be used.**
- Only use classes for your UDTs. The usage of public fields is not allowed! The definition of inline member functions is only allowed, if mandatory!
- Do not put class definitions and member function definitions into separate files (we have not yet discussed separated compilation of UDTs).
- Your types should apply const-ness as far as possible. They should be const-correct. Minimize the usage of non-const&!
- The results of the programming exercises need to be runnable applications! All programs have to be implemented as console programs.
- The programs need to be robust, i.e. they should cope with erroneous input from the user.
- You should be able to describe your programs after implementation. Comments are mandatory.
- In documentations as well as in comments, strings or user interfaces make correct use of language (spelling and grammar)!
- Don't panic: In programming multiple solutions are possible.

- Don't send binary files (e.g. the contents of debug/release folders) with your solutions! Do only send source and project files.
- If you have problems use the Visual Studio help (F1) or the Xcode help, books and the internet primarily.
- Of course you can also ask colleagues; but it is of course always better, if you find a solution yourself.