Object Georiënteerde technieken

Basis

1. useful and correct class (explain why)
2. useful and correct abstraction (explain why)
3. useful and correct encapsulation (explain why)
4. useful and correct inheritance (explain why)
5. useful and correct polymorphism (explain why)
6. useful and correct object composition (explain why)
7. useful and correct base class
8. useful and correct abstract base class
9. useful and correct virtual function
10. no mistake in object-oriented programming

Aanvullend

Algemeen

1. clean main (i.e. nothing in the main that should be in a class)
2. no globals, but statics if needed
3. correct protections
4. maintainability by clean uniform code style and good function naming and/or comments everywhere
5. separate header files
6. one complete project that compiles and does not crash
7. fully working project
8. sufficient git commits (+/- weekly)
9. correct files on git
10. working build manual as readme on GitHub (project must be possible to build from scratch on a clean PC)

OOP

1. at least 2 default constructors
2. at least 2 parameterized constructors
3. at least 2 copy constructors
4. at least 2 destructors
5. member initialization in constructors (the stuff behind a colon)
6. constructor forwarding
7. useful proven (dynamic) polymorphism
8. useful usage of "this" (if the code does not work without it)
9. useful member function
10. default values in function definition
11. useful member variabel
12. useful getters and setters for member variables
13. correct usage of inline function
14. useful template function or class

15. useful friend function or class

C++

1. everything in one or more self-made namespace(s)
2. 2 useful unsigned chars or other better usage of memory efficient type
3. at least 4 useful const references for variables
4. at least 4 useful const references for functions
5. at least 4 useful bool
6. dynamic memory allocation (new)
7. dynamic memory removing (delete)
8. 2 useful (modern) call-by-references
9. useful string class usage
10. useful container class
11. useful usage of nullptr
12. useful usage of (modern) file-I/O
13. useful exception handling
14. useful usage of lambda function
15. useful usage of threads

Uitbreiding

1. useful Qt class
2. useful usage of signals/slots
3. test-driven development (= written test plan or unit tests)
4. solve bug ticket (with pull request or commit message issue link and issue branch)
5. report a bug ticket on another project
6. usage of a GUI
7. usage of OpenGL or other 3D engine
8. useful usage of an external library (not Qt)
9. project that communicates (e.g. UART, BT) with hardware
10. a nice extra that you think that should deserve grading (stuff you put time in and is not rewarded by an item above)