Rubrics & Grading criteria

Criterium	Insufficient	Sufficient	Good	Excellent
Student describes algorithms using e.g. flowcharts & pseudocode	There are no demonstratable flowcharts, logbook, pseudocode for any of the implemented algorithms, or they don't explain the actual implementation.	Student has kept a (digital or analog) notebook with drawings (sketches/flowcharts) **or** pseudocode, showing thought and (upfront) design for **at least one** of the assignments.	Student has kept a (digital or analog) notebook with drawings **and** pseudocode, showing thought and (upfront) design for **multiple** assignments.	Student has clearly invested a lot of time in up front design, documenting algorithms in drawings/pseudocode with a very clear design demonstrating insight for **most** assignments.
Student uses/creates the right data structure to implement a given algorithm	Student has not chosen the correct data structure for the problem at hand or uses a functional instead of an Object Oriented approach.	Student has used the applicable (standard) data structures for each algorithm, explaining all the required changes.	Student has used the applicable (standard) data structures for each algorithm, explaining all the required changes and **motivation behind these decisions**.	Student has **researched data structures outside of the scope of the course successfully improving** an algorithm's implementation.
Student implements list, dictionary and graph based algorithms iteratively and recursively	• Student solved the requested algorithms by trial and error, but cannot adequately reproduce this feat. • Student copied code without any real understanding. • Student cannot adequately explain control/looping constructs and other operations applied in the code. • Student has not prepared any presentation for the assessment • Student did not use the GXPEngine or the provided base classes	All the 'sufficient' assignment criteria have been implemented. With a little bit of nudging, student provides **an adequate explanation** of the implemented algorithms.	All the 'good' assignment criteria have been implemented. Student has no problems explaining the implemented algorithms **by answering all of the assessment questions**.	All the 'excellent' assignment criteria have been implemented. Student's presentation provides an excellent explanation of the implemented algorithms that requires hardly any (additional questions for) clarification. Student clearly went the extra mile.
Student tests and debugs an algorithm	Student cannot demonstrate or explain how the step by step execution of an algorithm is performed.	Student can demonstrate the step by step execution of an algorithm through extensive debug logs.	Student can demonstrate the step by step execution of an algorithm through extensive debug logs and **perform a step by step execution of a requested algorithm using the debugger**.	Student can demonstrate the step by step execution of an algorithm through extensive debug logs, perform a step by step execution of a requested algorithm using the debugger **and is able to demonstrate at least one algorithm step by step visually in the provided test environment or by providing a detailed analysis using value tables.**
Student explains algorithmic complexity	Student has no idea what is meant by algorithmic complexity and how it compares to performance.	Student is able to explain the O notation and how it relates to the differences between complexity and performance Student is able to highlight and explain performance bottlenecks in the implemented algorithms.	Student is able to explain the O notation and how it relates to the differences between complexity and performance Student is able to highlight and explain performance bottlenecks in the implemented algorithms **and has valid suggestions for improvements**.	Student is able to explain the O notation and how it relates to the differences between complexity and performance Student is able to highlight and explain performance bottlenecks in the implemented algorithms and **can demonstrate implemented improvements**.
Overall code quality	Code is undocumented. No code conventions have been applied. Code quality is poor and messy.	Code has minimal documentation. Code conventions are there. With some time code is readable.	Every method is documented. Code conventions consistently applied. Readable code.	Classes and methods documented. Code conventions consistently applied. Readable, clear code: • Single Responsibility Principle applied • Don't repeat yourself principle applied • Self documenting

Grading table / Exam matrix

	I	S	G	E	%
Student describes algorithms using e.g. flowcharts & pseudocode	-10	6	9	12	12
Student uses/creates the right data structure to implement a given algorithm	-10	6	9	12	12
Student implements list, dictionary and graph based algorithms iteratively and recursively (A1)	-30	8	10	12	
Student implements list, dictionary and graph based algorithms iteratively and recursively (A2)	-30	8	10	12	40
Student implements list, dictionary and graph based algorithms iteratively and recursively (A3)	-30	8	12	16	
Student tests and debugs an algorithm	-10	6	9	12	12
Student explains algorithmic complexity	-10	6	9	12	12
Overall code quality	-10	6	9	12	12
MAX		54	77	100	