Python3 Programming Project (30%)

Mark Scheme

PROJECT REPORT (15%)

	Fail 0 12 25 32 38	Pass 42 45 48	Credit 52 55 58	High Credit 62 65 68	Distinction 74 78 82	High Distinction 88 94 100
INTRODUCTION	A problem of no merit OR Absent introduction: Work not submitted.	A short introduction to a problem which has little merit	A short introduction to a problem which has some merit	A detailed introduction to a computational problem that has some merit.	A detailed introduction to a substantial computational problem.	An excellent introduction to a substantial computational problem.
REQUIREMENTS	No attempt to gather requirements from users. Work of no merit OR Absent, work not submitted,	Short list of requirements identified without clearly identifying the users.	A list of requirements some, or all of which are identified as being from the users.	A description of the requirements of the software as identified by the student and potential users. Some user stories and methods of validation are evident.	Detailed user stories and methods of validation are evident. These artefacts are linked to the requirements gathered from the expected software users.	Excellent and unambiguous set of user stories and validation methods. The student has worked closely and effectively with the expected users to link requirements to these artefacts.
IMPLEMENTATION	No description of the process of implementing the program. Work of no merit OR Absent, work not submitted,	A very simple set of steps taken by the student to create a design and implement the program. No examples of debugging the program are given.	A good description of the steps taken by the student to design and implement the program. Some indication of a timeline of work is evident. Some debugging is evident.	A very thorough description of the steps to design and implement the program. Each step clearly describes the task and obstacles which were encountered. Debugging is used to identify problems and reiterate to improve the program.	An excellent description of the steps to design and implement the program. The objectives of each step are identified, along with the nature of the solution and how they relate to computational thinking. Debugging is used very well to identify design or usability issues and	In addition to the previous grade: The level of decomposition of the problem in order make the implementation more manageable is exemplary. The student has clearly spotted patterns in the problem and optimised them effectively The student can unambiguously

					reiterate to improve the program.	describe all elements of the program. Description of debugging the program shows it was effective in helping to overcome design and usability issues quickly.
PROGRAMMING TECHNIQUES IDENTIFIED	No attempt to identify the programming techniques used. Work of no merit OR Absent, work not submitted,	Basic set of programming techniques identified. "Sequence, Iteration, Conditions" Some errors in understanding are evident.	A good set of programming techniques identified. In addition to previous grading: "Validation, Annotation, Functions". Some errors in understanding are evident.	A suitably wide set of programming techniques identified. In addition to previous grading: "Recursion, Use of Modules/Libraries" Very little error in understanding is evident.	An extensive set of programming techniques identified. In addition to previous grading: "Algorithms, Data Structures" No errors in understanding are evident.	In addition to a clear understanding of techniques from the previous grades; in designing the solution, the student has clearly researched programming techniques that are viable methods for the problem. The selection of libraries, modules and other resources is appropriate to the problem.
INSTRUCTIONS & TESTING	No instructions to run or test the program are given. Work of no merit OR Absent, work not submitted,	A limited set of basic instructions to run the program are given. No methods of testing the program are evident. No plan to maintain or improve the program robustness is given.	Clear set of instructions to run the program. Evidence of testing but it is insubstantial. Some suggestions to maintain or improve the program robustness are given.	A clear set of instructions to run the program. Evidence of testing that sufficiently checks the validity of program. Good suggestions to maintain or improve the program robustness are given.	An excellent set of instruction to run the program. Testing is well described, and it is well planned, coherent and linked to the original requirements. The maintenance plan suggested can improve the robustness of the program. Some indication of how the solution might scale in the future	The instructions given to execute the program from a clean installation are excellent. Evidence of testing that was planned, coherent and linked to the original design and requirements specification. The maintenance plan is valid and comprehensive and

						can improve the program's robustness. Strong indications given of how the solution might scale in the future
REFLECTION and NEXT STEPS	No reflection on the software development process or the next steps is provided. Work of no merit OR Absent, work not submitted,	Attempt at reflecting upon the software development process but the conclusions are not linked to the project. They may contradict the student's project experience with no explanation.	Reflects upon the software development process but to no depth. Links to the student's experience yet any contradictions are only explained in a basic way.	The software development process is compared to the student's experience with this project. Any contradictions are explained.	The software development process is compared to the student's experience with this project. Any contradictions are well argued.	The final evaluation reflected extremely well upon the software development process, what was learned from the experience and the student gave some concrete next steps.
References/Opensource	No references given. No use of academic sources. Open source software is not credited.	Some use of academic sources; citation is poor. Open source software use is poorly credited.	EITHER: mostly used academic sources but citation is often inaccurate OR some academic sources but mostly accurate citation. Open Source software is credited mostly.	Mostly academic sources used; citation is largely accurate. Open source software is credited consistently.	Mostly Academic sources used and accurately/ consistently cited. Open source software is consistently and accurately credited.	The academic references, citations and the declaration of open source software is extremely well written and comprehensive.

PYTHON3 PROGRAM (15%)

The adoption of opensource code that provides a template for the student's program will not be marked. Only the student's own code in addition to the template will be marked.

	Fail 0 12 25 32 38	Pass 42 45 48	Credit 52 55 58	High Credit 62 65 68	Distinction 74 78 82	High Distinction 88 94 100
Annotation and comments	None	Some but misleading or inaccurate	Some but accurate and helpful	Many accurate and helpful but inconsistent	Accurate, helpful and consistent	Very thorough, accurate, helpful and consistent comments.
Choice and Use of Libraries	No libraries used	Libraries used but poorly or inappropriately	Libraries used poorly but they are appropriate to the task	Libraries used well and appropriately for the task	Good choice of Libraries used efficiently and effectively for the task	Excellent choice of libraries for the task which are used efficiently and efficiently.
Naming of variables	No attempt at giving meaningful variable names	Suitable names given to some variables	Most variables are named consistently	All variables are named consistently	All names are consistent, and they benefit readability and understanding	All names are consistent, and they benefit readability and understanding. They make use of a recognised naming standard.
Algorithms evidenced	No algorithms are recognisable or identifiable.	Some attempt at recreating a standard algorithm is evident	A good attempt at recreating a standard algorithm	Multiple algorithms are evident in the program. OR a single algorithm which is well implemented and identified.	Multiple algorithms such as search, sort, tree/graph traversal are evident and clearly identified by the programmer.	Algorithms outside the scope of the course are evident and clearly identified by the programmer.

Evidence of data structures	No data structure is identified to support an algorithm.	Some basic data structure is used to support an algorithm	An appropriate data structure is used well to support an algorithm. Eg. Stack/Queue	One or more data structures are used and identified to support the related algorithms.	Multiple data structures are used effectively and efficiently to support the related algorithms.	Data structures outside the scope of the course are used effectively and efficiently to support the related algorithms.
Use of Data Types	No data types are identified by the programmer.	On basic data types are used by the programmer and these are not identified.	Basic data types are used by the programme, some of which are identified. Some more complex data types may be used without identification.	Complex or library provided data types are used. Some of these are identified and annotated by the developer.	Complex (e.g. List, dictionary) or library data types are used and clearly annotated by the programmer	Uncommon and complex Data Types are used and annotated by the programmer.
Program works with expected output from specified input	The program does not work. Or does not produce an output as expected from the input. No data validation attempted	The program works with the test data provided; or under very controlled situations. One point of data validation is evident without explanation. It does not successfully improve software robustness.	The program works with any valid data but fails when unexpected data is provided to it. Multiple points of data validation are evident without explanation. They partially improve the program's robustness.	The program is responsive with any data, but it can be made to produce incorrect output. Multiple points of data validation are evident with explanation of their requirement. They partially improve the program's robustness	The program is responsive with any data and always produces the correct output. Multiple points of data validation are evident with explanation of their requirement. Software robustness is achieved	Program works with expected output from any specified input; but it does this efficiently, robustly and accurately. Multiple points of data validation are evident with explanation of their requirement. Software robustness is efficiently achieved