CAB403 Assignment 2

Criteria-Referenced Assessment

Marking Guidelines

We will assess your submitted source code, your written report and your demonstration video. The marks you receive will depend on how much of the specification you have implemented and the level of quality of the implementation and the level of quality of your written report. Note that the demonstration video **does not** have criteria attached to it, because it is used to mark the other parts of the assignment. **We will only mark aspects of your submitted project demonstrated in the video**, whether those aspects are present in your submitted code or not. If you want to be given marks for implementing a certain area of functionality, ensure that your video covers it. However, you may still lose marks for a feature if we find flaws in it when running your code, whether it is running flawlessly in the video or not.

For any given criterion, you may receive a mark that is somewhere between the different grade levels, e.g. if you have met all the requirements for a 6 and some of the requirements for a 7.

Your entire team will receive identical marks for the assignment.

Unit learning outcome #4: Construct low-level systems programs to carry out authentic systems programming tasks

Marks	**7 (85-100%)**	**6 (75-84%)**	**5 (65-74%)**	**4 (50-64%)**	**3 (40-49%)**	**2 (25-39%)/1 (<25%)**
Implementation of systems software following provided specification (15 marks)	The implementation meets all requirements for a grade of 6 and is thoroughly professional in coding quality, implementation and performance. No issues or limitations that compromise adherence to the specification	The implementation meets all requirements for a grade of 5 and is mostly professional in coding quality, implementation and performance. Some minor issues or limitations	The implementation meets all requirements for a grade of 4, except that all functionality described in the specification has been implemented and all features have been demonstrated in the submitted video	The implementation contains most of the functionality contained in the specification, with the software performing reliably and with a respectable level of code quality. All of this is demonstrated in the submitted video	The implementation only contains some of the functionality contained in the specification, or the code quality/reliability is unacceptably low, or the demonstration video fails to successfully demonstrate the submitted assignment	The implementation is either missing or non-functional, or the demonstration video is either absent or misleading as to the completion/quality of the submitted assignment
Use of low-level systems programming constructs (e.g. system calls, shared memory, threads, synchronisation primitives etc.) to meet project requirements (5 marks)	The use of low-level systems programming constructs meets all requirements for a grade of 6 and all use of systems programming constructs is appropriate and correct, considering all potential exceptional circumstances	The use of low-level systems programming constructs meets all requirements for a grade of 5 and systems programming constructs are used correctly and appropriately for the task at hand. All race conditions and undocumented/ unspecified behaviour are avoided.	The use of low-level systems programming constructs meets all requirements for a grade of 4 and all systems programming constructs required to complete the task are used	The implementation uses low-level systems programming constructs with only occasional omissions or incorrect use thereof	The implementation uses low-level systems programming constructions to meet the requirements of the specification, but with significant omissions or errors that cause poor reliability	The implementation does not use or barely uses systems programming constructs

Unit learning outcome #2: Articulate industry standards and critically apply best practice for developing safety-critical systems

Marks	**7 (85-100%)**	**6 (75-84%)**	**5 (65-74%)**	**4 (50-64%)**	**3 (40-49%)**	**2 (25-39%)/1 (<25%)**
Critical appraisal and evaluation of safety-critical software component (5 marks)	The appraisal and evaluation of the safety-critical software component in the report meets all requirements for a grade of 6 and provides a well-reasoned and insightful investigation into the issues with the software from a safety-critical perspective	The appraisal and evaluation of the safety-critical software component in the report meets all requirements for a grade of 5 and correctly identifies all issues with the software component from a safety-critical perspective	The appraisal and evaluation of the safety-critical software component in the report meets all requirements for a grade of 4 and provides a well-reasoned analysis of these safety-critical issues and the problems associated with them	The appraisal and evaluation of the safety-critical software component in the report identifies the majority of issues where the software falls short of meeting safety-critical software development guidelines and explaining why these issues are problematic from a safety-critical perspective	The appraisal and evaluation of the safety-critical software component in the report identifies multiple issues where the software falls short of meeting safety-critical software development guidelines and explains why these make the software component unsafe	No appraisal and evaluation of the safety-critical software component, or the appraisal has serious omissions of severe safety-critical problems, or no explanation as to why identified issues make the software unsafe
Applying best practices for developing safety-critical software (5 marks)	The submitted safety-critical component meets all requirements for a grade of 6 and follows best practices for safety-critical software development to the greatest extent possible, with all exceptions fully documented in the report	The submitted safety-critical component meets all requirements for a grade of 5, except there are either no or only very minor departures from best practices, and these do not realistically compromise the safety of the software. Any flaws or concessions are fully documented in the report	The submitted safety-critical component meets all requirements for a grade of 4, but with very few flaws, and the component runs reliably, with no observed deviations from the specification	The submitted safety-critical component is fully functional but has flaws from a safety-critical perspective. These flaws or concessions are mostly documented in the report. The component runs reliably with few deviations from the specification	The submitted safety-critical component has serious flaws from a safety-critical perspective, or is not 100% reliable, or the software	No safety-critical component was submitted, or the submitted component is non-functional or not demonstrated in the demonstration video

Marks	7 (85-100%)	6 (75-84%)	5 (65-74%)	4 (50-64%)	3 (40-49%)	2 (25-39%)/1 (<25%)
Applying best practices for software design (10 marks)	The architecture of the submitted software meets all requirements for a grade of 6 and shows that best practices for software design have been applied to a professional standard consistently across all submitted software	The architecture of the submitted software meets all requirements for a grade of 5 and shows that best practices for software design have been applied to a professional standard	The architecture of the submitted software meets all requirements for a grade of 4 and shows that best practices for software design have been applied consistently across all submitted software	The architecture of the submitted software shows the application of best practices for software design in terms of modularity, cohesion, coupling, separation of concerns etc.	The architecture of the submitted software is flawed in serious ways that has implications for maintaining the software, analysis of the software for correctness, performance of the software etc. Many best practices for software design are ignored	Submitted software is non-functional or highly incomplete, or there are very serious flaws in the architecture and design of the software