Appendix A: Major Programming Project

Project Specifications

Project Specifications				
CONTENT				
Summary			**2 MARKS**	
0 No summary or completely inadequate	1 Summary partially done		2 Summary encompasses all aspects of the problem	
Specifications of Program Function			**3 MARKS**	
0- No Functions listed	1- Function list is a single line statement / a list of 4 or less points	2 – Function list is a substantial list of appropriate outcomes	3 – Function List is complete and detailed	
Specifications of User Interface			**2 MARKS**	
0 – User Interface not specified or incorrectly specified	1 – one or two items are specified (inadequately)		2 – User Interface is completely specified	
Specification of Help			**2 MARKS**	
0 – Help not discussed	1 – Help partially discussed with omissions and/or errors		2 – Help completely detailed including menu options and types of help available	
Specifications of Data Storage			**3 MARKS**	
0 No information given on the data to be stored	1 Only a few items are incorrectly described	2 Many items are described with a few errors	3 All data to be stored has been described	
Hardware & Software requirements			**2 MARKS**	
0 – Hardware & Software not discussed	1 – Hardware & software is partially discussed for development, includes detail on what software is needed for what task		2 – Hardware & software is partially discussed for development, includes detail on what software is needed for what task	

System Design Document

Taking the template as an example, this document should be around 14 -20 pages (including title page & table of contents). Its main task is to detail the actual design elements of the program, namely:

- User interface design (what the screens look like & what happens on them)
- Program flow (how the program works – linked to the interface)
- Class design (what the classes are, their fields & methods)
- Database / Storage design (what the persistent storage structure is)

System Design Document			
User Interface Design NB: The GUI screen can be designed in a RAD environment (e.g. NetBeans / JBuilder / Delphi), on paper, or in a graphics program like Paint. Screen mock-ups are possible without writing code, therefore screenshots are acceptable as evidence of design. All data to be displayed on the screen must be listed. The action elements on the screen must be listed and clearly described.		**6 MARKS**	
0 / 1 – No screen design evident / Screen design is cursory	**2 / 4** – Screen design is evident but no consideration has been given to good design principles for an effective GUI or inadequate description of on-screen action elements	**5 / 6** – Screen design present, layout good, all on-screen action elements tabulated and described in detail.	
Sequencing - (also known as What Happens When) **In this section you describe the flow of events in the program – planning this can make your program easier / more logical to use (can help you decide on interface elements such as wizards, etc)** **NB:** The template contains flowcharts. You may use any algorithmic representation of sequencing of events in the flow of the program such as pseudocode.		**5 MARKS**	
0 / 1– No sequencing evident / Sequencing is rudimentary or cursory with little detail and large leaps of logic evident	2 / 3 – Sequencing is substantial but still shows leaps of logic / areas that have not been covered in appropriate detail	5 – Sequencing is broken into sections to cover all aspects of the functions and features in the Specification document. Flow is clear, well represented and easy to understand. No logic gaps are evident.	
Class Design **The candidates must provide their class design and explain their choice of classes, fields and methods.** **NB:** The template contains tables and this structure must be used for the class design where each field and method is explained.		**10 MARKS**	
0 / 3 – No class design evident / class design is rudimentary or cursory with little detail. Fields are incomplete, methods are minimal / not well thought out / not well described.	4 / 7 – Class design is substantial but still shows obvious gaps in missing fields / methods. Method descriptions more thorough but elements still missing.	8 / 10 – Class design is thorough – all fields and methods are present and well described. Sub-methods are present. Methods and fields clearly relate back to the Specification document.	

Persistent Storage Design **The candidate must provide their storage design** **NB:** Storage design should be done in tables. Screenshots of tables with record structure & field types from database software can be acceptable.			**6 MARKS**	
0 / 1 – No storage design evident / storage design is rudimentary or cursory (e.g. 'uses a database').	2 / 3 / 4 – Storage design is substantial. Some record design & description of fields are evident. Descriptions are, however, cursory and show evidence that they have not been completely thought through. Not all files / tables / relationships covered.	5 / 6 – Record structure is described – fields are listed, typed and described. Data structure for text / typed files is described. Storage design is appropriate to purpose and matches the Specification document requirements.		
Explanation of Storage Design **The candidate must provide an explanation of their storage design** For example a text file may have been a better solution than a database as the data to be stored is small in amount and simple. In this section you describe the way that data is stored so it can be re-accessed when the program is used again. Appropriate storage is what is required. What we need to see is that if, for example, a game is coded then high scores & save games are needed – and maybe other file handling to load appropriate data. DOES NOT have to be database.			**4 MARKS**	
0 / 1 – No explanation is given about storage or no understanding of the storage design is shown.	2 / 3 – Explanation is substantial but it is not completely justified and there are some areas of confusion or lack of understanding of the implication of the storage design.	4 – Explanation shows in depth understanding of the implications of the storage design and is completely justified.		

CODING & Technical Documentation

Taking the template as an example, this document can be anything from 10 – 100+ pages depending on the complexity and extent of the code that the candidate has written. Emphasis must be placed upon:

- Comments for all the methods (these can be copied & pasted from the Design Document)
- Separation of UI from working code
- Communication using typed methods (functions) and parameters
- Good general programming techniques (naming, indentation, appropriate data structures, etc)
- Good use of persistent storage
- Defensive programming
- Fulfilment of Design Specifications
- User Experience

CODING **NB:** This is assessed by examining the actual code – no attention need be paid to documentation / layout / etc.			
COMMENTS		**6 MARKS**	
0 – code has not been commented.	1/ 3 - Code contains some comments. Not all methods are commented. Comments are brief and contain little relevant detail. Scale the mark on the detail and quantity of appropriate comments.	4 / 6 – Code has been commented (4). All methods have comments describing what they do (5). Comments include the data they return (for typed methods) and the data they receive (parameters) (6). Steps in algorithms and solutions are commented as well. Variations of this will produce a different grade .	
Separation of UI from working code		**5 MARKS**	
0 – No separation – all code in the interface class / unit.	1/ 3 – Some separation. There are separate classes / units but work is still done in the UI. Insufficient further breakdown and separation of different aspects of the engine. This includes SQL statements for database centric programs (SQL is separation – you are passing off complex data handling to the database engine).	4 / 5 – Complete separation. Different classes are separated as well as the engine from the UI. The engine can be 'plugged into' a different UI that uses all the methods appropriately and will work without any issues.	
Inter-Code communication (Typed Methods and Parameters)		**6 MARKS**	
0 – No inter code communication (no typed methods (functions) or parameters.	1/ 4 – Some use of parameters / functions. Marks can be deducted (-1 per error type – multiple instances of the same error do not accumulate deductions) Errors include: Shows errors in comprehension of the concepts– unnecessary use of parameters, incorrect parameter types, parameters specified but not used, incorrect function types, failing to return values in functions, failing to use the results returned by functions, using variables / fields where the value is best returned by a function.	5 / 6 – effective and conceptually correct use of parameters and typed methods (functions).	
Good General Techniques		**6 MARKS**	
0 – No techniques.	1/ 5 – Errors in techniques (-1 per error type – multiple instances of the same error do not accumulate deductions). Errors include: No indentation, single level indentation, inconsistent or inaccurate indentation, variable names do not clearly indicate what the variable is used for, multiple variables used instead of arrays, multiple if statements instead of switches/cases, repetition of code (instead of using a procedure / function), code extending beyond the edge of the readable printed page.	6 – Technically perfect. Indentation immaculate. Variable names, data structures, etc all correct.	

Persistent storage / Querying			**5 MARKS**	
0 – No persistence. Data is not saved or retrieved.	1 / 3 - Storage is utilised but is either minimal or badly implemented. Inappropriate storage structure selected, poor implementation, storage intrudes on workflow of user.	4 - Storage effective and appropriate to the nature of the task. Storage management is not intrusive and does not disrupt the workflow of the user.	5 – Storage includes i/o exception handling and meaningful error messages.	
Defensive Programming (data validation, exception handling, error messages)			**4 MARKS**	
0 – No data validation.	1 / 2 – cursory data validation / error trapping. Only focussed on limited areas of code (file handling). UI elements are poorly selected (i.e. do not include UI that prevents incorrect data entry). Important data type checking not implemented.	3 – Aspects are complete – potential major IO errors protected with exception handling, GUI elements used and potential maths errors trapped. There are, however, a few areas where the candidate has not implemented defensive programming. Error messages not as clear as they could be.	4 – All appropriate data is controlled & validated using code and appropriate GUI elements and exception handling. All error messages are descriptive and easy to understand.	
Fulfilment of Specifications **NB:** this can only be assessed by running the compiled program.			**6 MARKS**	
0 – Not achieved.	1 / 3 – Basic implementation of specifications. Obvious omissions in missing functions / significant amount of functions do not work as specified.	4 – 90% of specification achieved. Perhaps all functions are there but do not all work correctly OR almost all functions are there but those that are there work 100%.	5 / 6 – All specifications complete and working 100%.	
User Experience **NB:** this can only be assessed by running the compiled program.			**4 MARKS**	
0 – Program does not execute.	1 – The user is lost – does not know where to start or how to achieve anything when using the program.	2 / 3 – Most of the program has a good user experience but navigating to some screens / functions is unnecessarily complex / impossible. Any aspect of the design / interaction is confusing or unsatisfying.	4 – An easy to use, completely easy to understand and navigate program: a wonderful user experience.	

Technical Documentation			
CONTENT			
Externally Sourced Code **NB:** This must be present even if the candidate only declares that no external code has been used.		**1 MARK**	
0 – Not Present.	1 – Candidate has declared used code. Confirm this with interview incorporating oral review of code and techniques.		
Explanation of Critical Algorithms **NB:** The core algorithms that are critical to the correct functioning of the program. There may be only a few (or even only 1) of these.		**2 MARKS**	
0 – Not Present.	1 – Algorithm present with no description of significance / poor flowchart / pseudocode.	2 – A good, clear description of why these algorithms are critical. Good flowchart / pseudocode.	
Advanced Techniques **NB:** This must be present even if the candidate only declares that no advanced techniques have been used.		**5 MARKS**	
0 / 1 Not Present.	2 / 3 – Spurious – candidate is manufacturing advanced concepts / not clearly explained.	4 / 5 – a good explanation of what the candidate regards as the 'extra mile' that they included in the project.	

TESTING Document

TESTING Document		
TEST PLAN AND RESULTS	**5 MARKS**	
Testing has been planned using well selected or generated data	2	
Table of tested data and results	3	
	100 MARKS	