A Level Computer Science:
Unit 3 - Programming Project Guidance
(20% of Course - 70 marks)

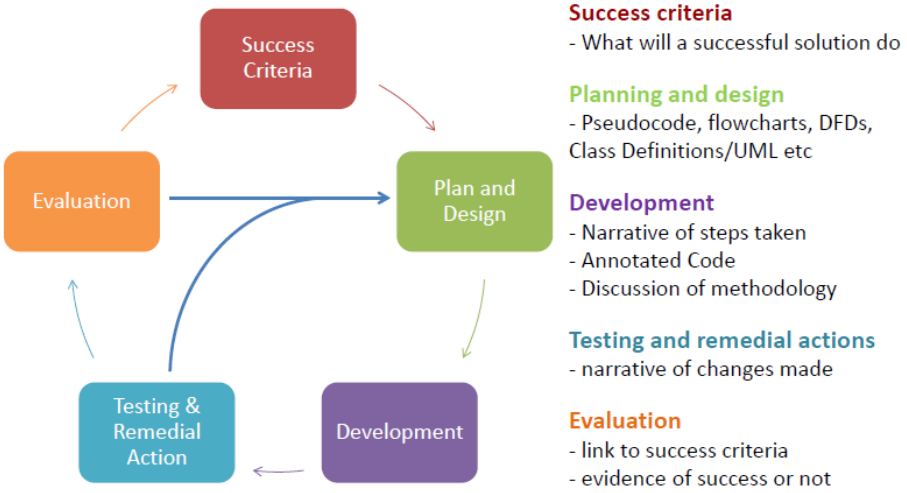
The purpose of this guide is to provide with the information and help needed to complete the A-level Computer Science Programming Project. It gives section/area headings, ideas of how to complete them and the different levels marking in criteria.

The information can be seen in more detail in the OCR Exam Board documentation. Also three exemplar papers are available (graded A-C for the old specification?) to give you an idea of what they are looking for.

By now you would have completed the analysis of your project and received valuable feedback from your teacher. From now until the end of summer your bridging work will be to complete the DESIGN, DEVELOPMENT AND TESTING of your project. It is very important to you complete most if not all development of your project during the summer otherwise you will not be able to meet the final deadline. This means that you must have a working project.
The next few pages detail that work that must be undertaken by you.

Iterative Development Process

It is expected that you will conduct the project using some form of iterative development, mainly (plan à create à test à re-plan à re-develop à re-test) as a way of refining the solution. it is important that this process happens and that evidence is kept of how the solution has evolved over time.



Design (15 marks)

1. Decompose Problem (this should have already been done by now)

- Break the problem down in to a series of smaller problems.
- Explain how these sub problems are linked and fit in the big picture (refer to ***Defined structure***).
- Justify why they are suitably sized.

	-
	Broken the problem down systematically into a series of smaller problems suitable for computational solutions describing the process.
	Broken the problem down systematically into a series of smaller problems suitable for computational solutions explaining the process.
	Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process.

2. Define Structure (this should have already been done by now)

- Produce a suitable diagram to show the structure of the solution and how the sub problems interlink.
  eg; Data Flow Diagram, Top Down Diagram, Class Diagram, other UML diagrams.

	-
	Defined the structure of the solution to be developed.
	Defined in detail the structure of the solution to be developed.

3. Algorithms: (due 5th July)

- Flow charts are optional but may help the design/modelling process.
- Pseudo code is required. *(the aim is that someone should be able to create the solution using your design)*
- Explain and justify the different parts of your algorithms and how they fit in the complete solution.

	Described elements of the solution using algorithms.
	Described the solution fully using appropriate and accurate algorithms.
	Described the solution fully using appropriate and accurate algorithms explaining how these algorithms form a complete solution to the problem.
	Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem.

Usability Features (due 13th July)

Usability is how easy to use a system is for the user. Areas to consider include: Navigation, consistency, user feedback, visual clarity, efficiency, flexibility and error prevention.

- Explain how you will consider and implement these areas in your solution.
- Justify the choices you have made.

	Described some usability features to be included in the solution.
	Described the usability features to be included in the solution.
	Described, explaining choices made, the usability features to be included in the solution.
	Described, justifying choices made, the usability features to be included in the solution.

Variables, Data Structures, Validation (due 18th July)

- Identify and justify Key variables.
- Identify and justify Data structures (eg; arrays, 2D arrays, text files, csv files, databases).
- If using OOP identify and explain Classes.
- Explain and justify any validation required.

	Identified the key variables / data structures / classes (as appropriate to the proposed solution).
	Identified the key variables / data structures / classes (as appropriate to the proposed solution) and any necessary validation.
	Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) explaining any necessary validation.
	Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation.

Iterative Development Test Data (due 25th July)

The best way to record tests is through the use of a test table, headers may include:

Test No.	What testing	How testing	Test data	Good/Bad/ Boundary	Expected result	Actual result	Actions needed

- Identify test data for each module/sub problem and what is expected to happen as you are developing the solution.
- Justify why this data was chosen.

	Identified some test data to be used during the iterative or post development phase of the process.
	Identified the test data to be used during the iterative development of the solution.
	Identified and justified the test data to be used during the iterative development of the solution.

Post Development Test Data (due 25th July)

- Identify test data that will check to see if all of the success criterions have been met at the end of the development stage.
- Justify why this data was chosen.

	-
	Identified any further data to be used in the post development phase.
	Identified and justified any further data to be used in the post development phase.

*Initial designs/tests are allowed to be modified/adapted at a later date and for higher marks this is expected.

Development (15 marks)- Monday 6th September-Whole development section

This is where you make the software and show evidence of it. Remember to test your project as you go along and record any failed tests.

- ITERATIVE DEVELOPMENT - Develop the project in the chunks/ sub problems identified in the design.
- VERSIONS - Save different versions of the solution that progressively improves/alters over time (eg; v1, v2, v3).
- MODULES - Divide the code into modules and make sure you are trying to be efficient with algorithms.
- COMMENTS - Comment code during development and describe sections in the write up.
- TESTING – Make sure you are testing (and getting print screens) as you go along using your test data.
- STAKE HOLDERS – Allow stake holders to regularly review solution and use feedback to inform any changes.

Development Evidence

- Use print screens to show how the development has been an iterative process.
- Explain what was done at each stage and justify why choices were made.
- Highlight unique or complex features in the code.
- Relate the different stages chunks/ sub problems identified in the design.

	Provided evidence of some iterative development for a coded solution.
	Provided evidence for most stages of the iterative development process for a coded solution describing what they did at each stage.
	Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did at each stage.
	Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did and justifying why.

Prototyping

- Include evidence of different versions of the solution that progressively improves/progress over time.

	Solution may be linear.
	-
	Provided evidence of some prototype versions of their solution.
	Provided evidence of prototype versions of their solution for each stage of the process.

Code Modules

- Try to ensure code is efficient and doesn't have needless steps.
- Divide code into the modules identified in designs.

	Code may be inefficient.
	Solution will have some structure.

	The solution will be modular in nature.
	The solution will be well structured and modular in nature.

Code Comments

- Comment code in detail so as others would be able to understand the structure. *This may not require every line.*

	Code may not be annotated appropriately.
	Code will be briefly annotated to explain key components.
	Code will be annotated to explain all key components.
	Code will be annotated to aid future maintenance of the system.

Variables/Structure Names

- Decide on a naming convention and use it throughout eg; `myFirstVariable` or `my_first_variable`
- Be sure to use appropriate and meaningful names.

	Variable names may be inappropriate.
	Some variable and/or structure names will be largely appropriate.
	Most variables and structures will be appropriately named.
	All variables and structures will be appropriately named.

Validation

- Include validation on use inputs and evidence it. *(Checks could be; Range, Presences, Format, Length)*

	There will be little or no evidence of validation.
	There will be evidence of some basic validation.
	There will be evidence of validation for most key elements of the solution.
	There will be evidence of validation for all key elements of the solution.

Reviewed

- Include regular opportunities for you and stake holders to review the solution at different stages against the original success criteria.
- This could be a chance to implement black box testing.

	There will be little evidence of review during the development.
	There will be evidence that the development was reviewed at some stage during the process.
	The development will show review at most key stages in the process.
	The development will show review at all key stages in the process.