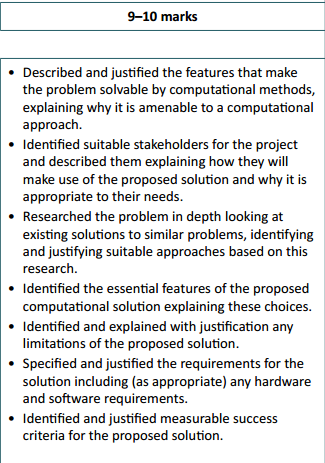
**A Level Computer Science Project – Marking criteria**

WHY is a computerised solution needed? JUSTIFY what it is about the problem that means it is suited to be solved by a computer. What significant PROCESSING is involved?

**Analysis**



WHO will use it? Multiple people? Who are they and what will they need it to do? DESCRIBE both of these in detail. Use names!

Start to break your proposed system down into sections. What exactly will it do? What are the essential features that it MUST do to be a successful project?

What options are there instead of creating a brand new system? Do other systems exist that do the same or similar things? Can off-the-shelf software (eg MS Office, databases) do part of the task?

JUSTIFY why you have decided to make an entirely new system and give a (brief) outline of what your new system will do and how it will do it.

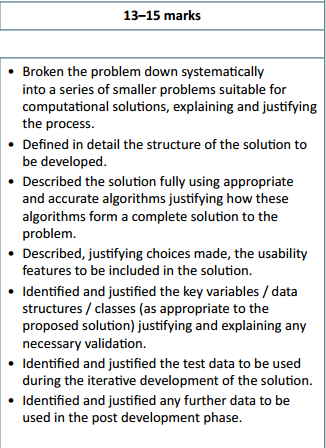
Limitations – what are you NOT going to do in your project? What areas will you not concentrate on (perhaps because of time, expertise, etc). Is it OK to miss these things out? Why are they not essential?

Measurable success criteria.

You must have, at the end of your analysis, a list of EVERYTHING that your proposed solution will do. This can be a bullet point list. These should be measurable (so “a login system that lets users log in with a username and password” is fine, “be easy to use” is not fine – how would you measure this?) If they were in some way numeric (ie “will store up to 500 customers”), even better.  
  
In the evaluation section, you will check these off to say whether you have done them or not.

What HARDWARE and SOFTWARE (including the operating system, programming language environment, etc if applicable) would be needed for the end user to run the completed program?

**Design**



DEFINE IN DETAIL the structure of the solution, in terms of which screens will be needed, what each of these will do, etc. How will the user access each screen?

USABILITY FEATURES… this means GUI, but also other things such as help features, instructions, etc. How will you make the system easy to use?

It would be wise here to include sketches / layout diagrams for each screen to show this.

ALGORITHMS…. How will each module work? What input will it take, what processing will be done and what output will be produced? This can be done in either pseudocode or flowcharts, but if using flowcharts be careful to explain HOW to do things, not just WHAT will be done. If at this stage you struggle to identify algorithms, this is a big clue that your project is not complex enough for A Level.

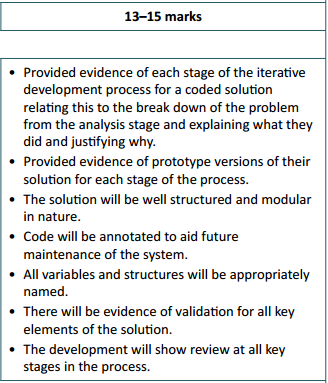
JUSTIFY how these algorithms form a complete solution (so link it back to your analysis!)

DECOMPOSE the problem, and do this as a series of steps (not just one final diagram. How can the problem be split down into smaller chunks? JUSTIFY why each module is important and stands on its own.

TEST data to be used is credited here but is probably best presented in the testing section

IDENTIFIED key variables / data structures – design which variables you will need (this links to the algorithms) and also what DATA STORAGE you need – text files, databases, etc. You need a design for what data these will store and in what structure.

**Development**



DEVELOPMENT is actually worth 25 marks – 15 marks here and 10 marks for testing during development (on next page).

Your evidence of HOW you developed the system will primarily be formed of your development log, showing screenshots of your progress and testing along the way.

Your notes explaining WHAT you did and JUSTIFYING why you did each thing are also just as important.

Finally, the finished product will be looked at for three things:

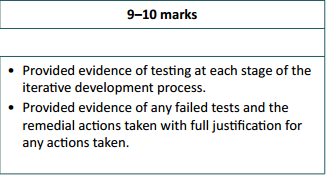
* Is code annotated (with **comments**) to explain what each section does, so that someone else can come in and maintain the system in future?
* Are **variables and data structures named appropriately**? You may wish to discuss this.
* Is **validation** used – ie can the used make the program crash by entering the wrong thing, or is this trapped?

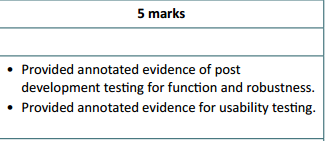
**Testing (during and post)**

You should have been testing each section during the development, with a test plan. For example, as you completed the login section, you should have been checking that it worked correctly, using a variety of data (normal, erroneous and borderline).

Any problems should have been fixed and re-tested, explaining what went wrong and what was fixed.

Don’t forget, the test data used here is credited in the marking for your design section.



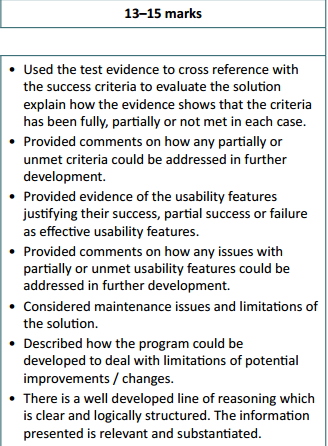


After the system is complete, there is now a need to test the system as a whole, again with a test plan. This should focus on “getting the job done”, but also robustness (ie checking that the program doesn’t break).

You also need to do some usability testing. There are many ways to do this, but getting real users involved and asking them to use it (and carry out specific tasks), then getting feedback from them is an easy way.

**Evaluation**

Last section! But 15 marks out of 70 is a big percentage.



The last bullet point simply means that the whole project is well organised, well written and logically presented.

It is important to remember that in the evaluation, you do not gain marks by pretending that everything worked perfectly! You can get 15 out of 15 for a program with lots of problems if you are able to discuss and explain all of the problems.

Limitations of the solution. Are there any? (Very sure that there will be). These could be the limitations that you mentioned in the analysis that you planned for, or limitations of things that you didn’t manage to get working perfectly. How could the program be changed to fix these, if you were given more development time?

Maintenance issues. What needs to be kept up to date or changed in the future? What would someone who is coming in to take over the system need to know about how it works.

Same as above but this time for the usability features. Look at what you said in your design section about usability features and provide evidence of these having been done (screenshots or a link to the page number in testing). Again, do you think these worked? You could link this to your usability testing (ie does your user think that your help text is actually helpful)? Anything that has NOT been met or only PARTIALLY met, how could you improve these with further development time?

Go through your success criteria from the analysis section and discuss, point by point, whether each point has been completed. If it HAS been completed, either show evidence (screenshots) of this section working, or refer to the page number for the testing evidence that shows this.   
If any bullet point has NOT been met or has only PARTIALLY been met, then you need to discuss why this was and how it could be fixed if you were to continue to develop the system.