O Level Computer Studies 7010
Unit 4: Algorithm design

Recommended Prior Knowledge
Students need to have studied unit 3, systems analysis, before beginning this unit.

Context
The rest of the systems life cycle is covered in this unit.

Outline
The aim of this unit is to cover the design, development, implementation, maintenance and review principles, which include techniques and tools which relate to the solution to a problem. A study of these topics is reinforced through practical work and illustrated by a consideration of existing problem solutions in computer applications. Candidates should have experience of representing algorithms informally (as structure diagrams, flowcharts, step sequences, descriptions).

AO	Learning outcomes	Suggested Teaching activities	Learning resources
3.1	Defining the scope of separate modules Designing algorithms which relate clearly to the requirements of the system	Develop several examples to demonstrate what an algorithm is and how they are written. For example: • adding two numbers together • finding average of 2 or more numbers • finding largest and smallest numbers in an input set • sorting out ranges of numbers (e.g. if a series of temperatures were input how many were in the range -20 to 0, 0 to 20 and over 20) • use of formulae (e.g. convert ^{o}F to ^{o}C)	http://www.theteacher99.btinternet.co.uk/theteacher/gcse/newgcse/module6/task12.htm a basic introduction to the stages http://www.teach-ict.com/as_a2/topics/system_life_cycle/slc/index.htm provides a more in-depth look at the systems life cycle http://www.theteacher99.btinternet.co.uk/theteacher/gcse/newgcse/others/algorithms.htm provides a good introduction using a real life example C+W 9.2
	Explaining algorithms and how they relate to the system Explaining how hardware needs arise from the output required from the system Algorithm tools	The above could be further extended to look at more complex problems from real life situations. All this links into 3.2 (unit 5) where pseudocode could be produced as part of the algorithmic design. The use of flow charts as an algorithmic tool shouldn't be overlooked – it is a very useful exercise to develop the solution to a problem using a flow chart and then convert into pseudocode later on.	http://www.theteacher99.btinternet.co.uk/theteacher/gcse/newgcse/others/algorithms.htm provides a good introduction using a real life example http://www.smartdraw.com/tutorials/flowcharts/whatis.htm provides a tutorial on how to draw flowcharts http://www.sharewareorder.com/WizFlow-Flowcharter-screenshot-2401.htm some free

AO	Learning outcomes	Suggested Teaching activities	Learning resources
	Interpreting and testing algorithms	But for more complex problems, sometimes the use of a flow chart only may be entirely appropriate (also links into systems flowcharts – unit 3). This should lay the foundations for the student's project work.	flowcharting software http://www.ictgcse.com/sub_projects/ictgcse_th_sysflow.htm an introduction to flowcharts C+W 9.2, 9.3 and 9.4
		Use examples of algorithms and practice dry running, use some algorithms that work and some that don't, use different sets of test data. Once algorithms (both in flow chart and pseudocode form) have been developed it is essential to test them out. Carrying out a dry run with various sets of test data is essential here. The test data should be carefully chosen to cover: • examples where the final output is known so the test data merely demonstrates the correctness of the algorithm • examples of test data which shouldn't work (e.g. inputting negative numbers into a wages calculation) – this may be testing validation rules • examples of test data which tests the extremes (e.g. input somebody's age as 110 or 1) • finally input test data as a genuine run of the program once all the above testing has indicated the robustness of the algorithm	C+W 10 provides useful practical examples.