Teacher Resource Bank

GCE Computing

- COMP4 Definition of problem types for projects

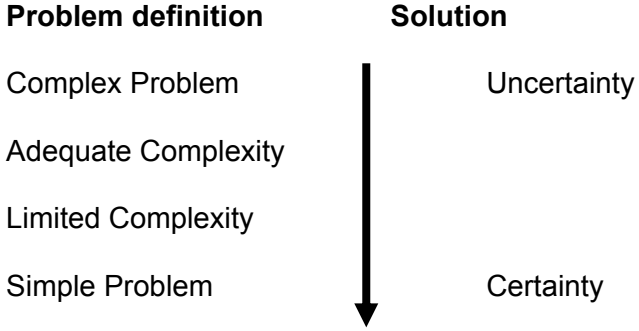
Complex, adequate, limited and simple problems, including advice about task setting

Important note about Simple problems: Projects based upon Simple problems may result in low marks which may or may not achieve a Grade E when the decisions about standards are made annually by the GCE Computing Awarding Committee each year immediately prior to results.

1. Identify the degree of complexity of a project at an early stage

As teachers move into the teaching of COMP 4[1], it becomes crucial to identify the complexity of the project at an early stage. This is because all sections for the marking criteria are related to the degree of complexity of the project except for System Testing and Quality of Communication. A simple project (e.g. one implemented in a database package with just a few lines of candidate code or one fully programmed by the candidate but where the coding is trivial) even if implemented and documented extremely well is likely to be a borderline project under the new specification. Good candidates need to be advised to undertake a Complex project in order to achieve a high grade. However, there is little point in a weaker candidate trying to implement a Complex project that is significantly beyond their capability.

One very crude way of determining the degree of complexity is to consider the number of possible solutions. In general, simple projects may have a single obvious solution but as we move to complex projects there are likely to be several possible solutions each with their own merit.



The key to the complexity also lies in the processing and/or the volumetrics. In the past, most candidates concentrated on finding a solution to an organisation's data-processing problem but AQA is keen to encourage diversity in coursework and the following problem areas could be used in the Computing specification:

- a data-processing problem of an organisation

[1] Use the latest version of the specification: you should be aware that the AQA printed version of the specification has been superseded by the online version and there have been minor changes to the specification including the assessment criteria for COMP4. **The current website version is to be considered as definitive.** You should read these Teacher Resource Bank notes in parallel with section 3.4 of the online version of the specification while advising your candidates as they begin the COMP4 course.

Many of the comments from previous legacy Principal Moderators' Reports on the Examination for CPT6 are still valid and should also be considered. These are available on the AQA Website at: http://www.aqa.org.uk/qual/gceasa/comp_exam.php

- a scientific or mathematical problem
- a simulation of a real-life situation
- a computer-aided learning system
- a software tool or utility
- a control system / robotics.

It is important to note that with regard to the assessment criteria given in the GCE Computing specification not all criteria will necessarily be applicable to each type of project.

The production of Computer Games is not ruled out but proposals should be considered carefully. If developing a Computer Games project it **must** have a genuine, independent end user (the client who specifies the problem) and the end user cannot be the candidate. A friend of the candidate is also unlikely to be an appropriate independent end user but this option is not ruled out; the judgement of the candidate's teacher must be brought to bear in this situation. A computer-aided learning system might involve a type of game, for example. Other examples might include the development of learning aids at KS1 or KS2, for SEN students in later key stages or for learning aids at KS4 and KS5. There is also scope for Xbox or PDA programming as listed later in the section on complex programming projects.

The Candidate Record Form has been modified so that in the Project Proposal section, the name of the End User and their Key role e.g. Job Title in relation to the project has to be clearly identified from the outset in the choice of the project. The supervisor also has to verify that they have authenticated the candidate's work, **including the involvement of the End User,** in the project when they sign off the form.

2. Defining the complexity of a project for a fully programmed solution

Defining complexity of the project is not a simple task. We have tried to exemplify the different levels of complexity for a programmed solution below to help you advise your candidates and later in this document for part-programmed, part-package solutions. You are advised that the following criteria should be applied using a 'best fit' method i.e. a Complex project may have a mix of both Complex and Adequate criteria.

At the Analysis stage, teachers need to make a judgement as to whether the problem posed by the candidate for their project is:

- Complex
- Adequately Complex
- Limited in Complexity
- Simple.

Teachers are asked to state on the Candidate Record Form the perceived complexity of a project. If you are in doubt, consult your Coursework Adviser. (Contact details for Coursework Advisers are sent to Heads of GCE Computing via their Examinations Officer in the autumn term.)

2.1 A Complex problem

A Complex problem is one that has the potential to involve one or more of the following to the depth indicated in columns 2 and 3 of Table 1, when *automated*.

- Non-trivial algorithms, standard or user-defined, e.g. a graph traversal algorithm, recursive algorithms

- Use of sophisticated features of programming language / complexity of programming language, e.g. sophisticated data structures, runtime created objects, user-defined OOP classes
- Time-based simulation
- Development of program solutions for portable devices / games consoles
- Complexity of non-computing field of the problem, e.g. 3-D vector manipulation
- Communication Protocols, e.g. TCP connections
- Image Processing / pattern recognition, e.g. steganography, use of regular expressions

2.2 A problem of Adequate Complexity

This is a problem in which columns 2 and 3 of Table 1 are interpreted in the following restricted ways:

- The algorithms will be non-recursive, linear time or quadratic time e.g. a For Loop or a nested For Loop comprising an inner and outer For Loop.
- Has a requirement for a mix of different data types to be stored together, e.g. a file of records.
- The range of processing tasks requires a modular approach e.g. user-defined subprograms.
- The time complexity of the problem is limited to linear or quadratic time.
- In the case of time-based simulation, the simulation is constrained by a relatively small number of objects for which the outcome can be predetermined by analysis.
- Complexity of non-computing field of the problem e.g. mathematics involved at no more than Key Stage 4 Level.
- The problem's solution will not involve communication protocols or image processing / pattern recognition or development of program solutions for portable devices / games consoles.

2.3 A problem of Limited Complexity

This type of problem is one in which columns 2 and 3 of Table 1 are interpreted in the following restricted ways:

- The solution to the problem will not necessarily require permanent storage of data and if it does, then the data stored is of one data type e.g. a file of integers or strings.
- It involves non-recursive algorithms carrying out simple mathematical calculations e.g. addition, subtraction, multiplication and division.
- The size of the input is limited in range so that time complexity is not an issue.
- In the case of time-based simulation, the simulation is constrained by a very small number of objects making analysis of the outcome very straightforward.

2.4 A Simple problem

A Simple problem is one that interprets columns 2 and 3 of Table 1 in the following restricted ways:

- The solution to the problem will not require permanent storage of data.
- The solution will employ trivial algorithms or will not have any algorithms.
- The solution will have very limited input / output and no significant processing.

The advice in Section 3 and Table 1 refers to entirely candidate programmed solutions. Candidates may still wish to use a database package **as part of their solution** e.g. a combination of Microsoft Access and Delphi / VB / VBA / VB.NET or some other package in combination with a programming language.

TABLE 1

Principles	Sub-sections	Exemplars
FC1[2] Algorithms	Standard– Non-recursive and recursive	Tree Traversal, List Traversal, Graph Traversal
	User defined	Route Planning
FC2 Use of Sophisticated features of programming language	Data Structures	2 – D arrays User Defined Records User defined types Lists, Linked or otherwise Graphs Queues Stacks Trees
	Creating objects at run time	x := TButton.Create

	User defined Classes / Objects	`TAccount = Class`
		`Public`
		`Function`
		`GetBalance : Integer;`
		`etc`
		`Private`
		`Balance : Integer;`
		`etc`
		`End;`

	Time complexity / Space complexity	Polynomial / Exponential Volumetrics e.g. scheduling / timetabling problems

[2] 'FC' stands for fully coded.

FC3 Complexity of programming language	OOP independent of GUI List processing languages, Functional programming languages, Logic programming, Simulation languages, OpenGL, Assembly language programming	Java, C++, Object Pascal, C# LISP, Scheme, PROLOG, Simula, F#
FC4 Time-based simulation	Discrete Event Modelling	Kinetic Theory Modelling of gases Modelling/simulation of some business activity e.g. queuing in a petrol station or supermarket
FC5 Development of program solutions for portable devices/ games consoles	Special Purpose processing systems / hardware, DreamSpark system	Mobile Phones, Smart phones, X Box
FC6 Complexity of non-computing field of the problem	Modelling of mathematical processes	Statistical analysis, quantum mechanics, 3D vector manipulation, matrix algebra, encryption techniques
FC7 Communication protocols	HTTP, FTP, TCP/IP, IRC, Bots	Network projects
FC8 Image processing/ pattern recognition	Processing bit patterns	Steganography (encoding text in bitmaps), compression techniques,

3. Database-related projects

In a database-related project, it is inappropriate to ask the question “How many tables do I need for it to be a complex project”. Using as many as five or seven linked tables might involve just data storage / retrieval and possibly trivial processing by the package and the candidate might have developed virtually no code of their own and thus it might be a simple project. By contrast, a scheduling project might just involve 2 or 3 tables but have high data volumes and many constraints so that it would become a complex project with a great deal of candidate developed code. By comparison, even with just 2 or 3 files, an entirely programmed package that uses programming language native file handling operations to link files, store and retrieve data **and** which contains complex candidate-written code could be a complex project. In fact, a project that has the potential to be solved using a complex programmed interlinked file structure can classify as a complex project if this approach is used to solve the problem. This could quite easily be a data processing project. Indeed, data processing projects are welcome.

A project which simply uses a package to collect data and produces simple reports with little or no data transformation is not complex enough for A2 and cannot score a good mark however attractive the implementation looks and however conscientious the candidate was and despite how well the report is written. Candidates who use packages and carry out basic customisation of the package with simple self-written macros are unlikely to achieve high marks.

Exemplar extracts from project reports which illustrate good practice for different sections of the report will be published in the Teacher Resource Bank from time to time.

Complex queries / reports produced entirely in a database package such as Access are not relevant in determining the overall complexity of the projects unless the candidate has demonstrably written the SQL code.

4. Determining the complexity of a project for a part-programmed, part- package solution

To determine the degree of complexity of a project involving both a package and a programming language, you would need to apply from Table 1, above, Principle FC1 (Algorithms), Principle FC2 (Use of sophisticated features of the programming language), Principle FC3 (Complexity of programming language), Principle FC6 (Complexity of non-computing field of the problem) and Principle FC7 (Communication Protocols). **In addition**, you will need to apply these Principles in the context shown in Table 2, below, which has been designed specifically for classifying package-programming language solutions. The table also has two additional principles, Principles PP9 and PP10, which apply to part-programmed, part-package solutions.

TABLE 2

Principles	Sub-sections	Exemplars
PP1[3] Algorithms	User defined	Manipulating data extracted from a database such as extracting data from one or more tables in order to update other tables or to aggregate data in order to perform data analysis

PP2 Use of sophisticated features of programming language	Use of methods in database objects to connect to a local or remote database, and to extract, store, update and delete data in a database	`IBQuery1.SQL := 'Select From T1, T2 Where (T1.Id = T2.Id) And (T1.Surname = ' + '''BOND''');`
		`IBQuery1.Open;`
		
		`<?php`
		`$con = mysql_connect("localhost","peter","abc123");`
		`if (!$con)`
		`{`
		`die('Could not connect: ' . mysql_error());`
		`}`
		`// some code`
		`?>`

PP3 Complexity of programming language	OOP independent of GUI. Use of networking components. Use of database, query, transaction components	Object Pascal (Delphi), PHP, Python, ASP, VB/VBA/VB.Net

PP6 Complexity of non-computing field of the problem	The number of entities and their attributes in the problem domain poses a significant identification and resolution challenge when generating the conceptual model. Potential volume of data is significant	`Calf (StatutoryTagID, Gender, DateOfBirth, TimeOfBirth, PlaceOfBirth, DateOfDeath, PhysicalDescription, AssistanceToCalving, Breed, Remarks, Horns, TagCheck, BullStatutoryTagID, CowStatutoryTagID)`
		
		`Cow (CowStatutoryTagID, CowManagementTagID, NumberOfCalves)`
		
		`Bull (BullStatutoryTagID, BullManagementTagID)`
		
		`CowVaccinations VaccinationInstanceID, DateOfApplication, Details,`

[3] 'PP' stands for package-programming

		StatutoryTagID, VaccinationID) Vaccination (VaccinationID, VaccinationName) Slaughter (StatutoryTagID, DateOfSlaughter, LiveWeight, DeadWeight, Price, Grade)
PP7 Communication protocols	HTTP, TCP/IP, remote database networking protocols	Client-server projects involving server-side scripting/web server extension/remote database access
PP9 Complexity of data modelling	Derivation of non-trivial conceptual model of a real world problem possessing several non-trivial many-to-many relationships and composite keys/entity identifiers	Entity-Relationship diagram with intersection entities and fully normalised entities, e.g. Order, OrderLine, Customer, Stock, StockItem, Invoice entities for an invoicing system
PP10 Use of SQL commands and queries	DDL script Range of different Select statements involving joining two or more relations, Order By Insert, Update, Delete SQL created by candidate	Use of Case tool to generate DDL script, use of DDL script to generate database tables in, for example MySQL or Interbase; dynamically parameterised SQL (parameters set at runtime) embedded in programming language statements written by candidate

4.1 A Complex data processing problem

A Complex data processing problem involving a part-programmed, part-package solution
is one that is based on a complex non-computing field as indicated in columns 2 and 3 of Table 2 and which will generate a complex data model as indicated in columns 2 and 3 of Table 2 and will have the potential to involve the use of sophisticated SQL queries and commands if it is a database project as indicated in columns 2 and 3 of Table 2. In addition, it will have the potential to involve one or more of the following to the depth indicated in columns 2 and 3 of Table 2, when *automated*.

- Non-trivial algorithms, standard or user-defined e.g. data extracted by SQL queries must be processed further to obtain desired result
- Use of sophisticated features of programming language / complexity of programming language, e.g. embedding of SQL statements within the programming language and assigning query parameter values at runtime; web-based programming; has the potential to be multi-user; modular approach, i.e. use of procedures/functions/separate units.

- Communication Protocols, e.g. use of remote database connections
- The solution will have an extensive range of input/output and processing.

4.2 A data processing problem of Adequate Complexity

This type of problem is one in which columns 2 and 3 of Table 2 are interpreted in the following restricted ways:

- The algorithms will relate to applying just single SQL query /insert/update/delete statements.
- Complexity of non-computing field of the problem – The number of entities and their attributes in the problem domain poses a non-trivial but straightforward identification and resolution challenge when generating the conceptual model. The conceptual model has to model many-to-many relationships. The project has the potential to store a significant volume of data.
- The complexity of data modelling is such that a non-trivial conceptual model of a real world problem is modelled (therefore possessing many-to-many relationships) but resolution of many-to-many relationships into one-to-many relationships is straightforward.
- Use of sophisticated features of programming language / complexity of programming language, e.g. embedding of SQL statements within the programming language and assigning query parameter values at runtime; modular approach, i.e. use of procedures/functions/separate units.
- The solution will have a significant range of input/output and processing.
- The problem's solution will not involve communication protocols.

4.3 A data processing problem of Limited Complexity

This type of problem is one in which columns 2 and 3 of Table 2 would be interpreted in the following restricted ways:

- Complexity of non-computing field of the problem is limited; the number of entities and their attributes in the problem domain make generating the conceptual model an exercise on a par with an exercise used to introduce data modelling to candidates.
- The complexity of data modelling is such that the conceptual model of a real world problem can be generated in a formulaic way and without significant mental challenge. The data volume that needs to be stored is not likely to be significant.
- SQL statements are not used, or if they are, they are not parameterised for assignment of values at runtime.
- The use of a programming language is limited.
- The solution will have a limited range of input/output and processing.
- The problem's solution will not involve communication protocols.

4.4 A Simple data processing problem

A Simple problem is one in which columns 2 and 3 of Table 2 are interpreted in the following restricted ways:

- Complexity of non-computing field of the problem is simple; the number of entities and their attributes in the problem domain make generating the conceptual model trivial. The conceptual model has no many-to-many relationships.
- The complexity of data modelling is such that the conceptual model of a real world problem can be generated by applying little or no thought. The data volume that needs to be stored is not likely to be significant.
- SQL statements are not used or are trivially used.
- The problem's solution will not involve communication protocols.
- The solution will employ trivial or no algorithms
- The use of a programming language is very limited.
- The solution will have very limited input / output and processing.

5. Summary

In summary, as candidates are starting to choose their projects if they are very able candidates, you should consider advising them to opt for a complex project. You should consider advising a candidate not to attempt a project that is too complex if the candidate is less able or has limited programming aptitude (possibly evidenced by their COMP1 performance).

All candidates must have a genuine, independent end user and the candidate cannot be their own end user.

Standardising Meetings will be offered in late summer and autumn 2009. Exemplar projects and commentaries will be provided for those attending. Extracts from other exemplars will appear in the Teachers Resource Bank in due course.