OCR
RECOGNISING ACHIEVEMENT

AS/A Level GCE

GCE Computing

OCR Advanced Subsidiary GCE in Computing H047

OCR Advanced GCE in Computing H447

specification

version 4 – May 2015

QAN 500/2279/9
QAN 500/2191/6

Contents

Contents

Vertical black lines indicate a significant change to the previous printed version.

1 About these Qualifications

This booklet contains OCR's Advanced Subsidiary GCE and Advanced GCE specifications in Computing for teaching from September 2013.

These specifications encourage candidates to develop their knowledge and understanding of hardware and communications, software, applications and effects, and information; as well as skills in analysis, design, implementation and evaluation.

1.1 The Two-Unit AS

The Advanced Subsidiary (AS) GCE is both a 'stand-alone' qualification and also the first half of the corresponding Advanced GCE. The AS GCE is assessed at a standard appropriate for candidates who have completed the first year of study (both in terms of teaching time and content) of the corresponding two-year Advanced GCE course, ie between GCSE and Advanced GCE.

The AS GCE (from September 2013) is made up of **two** mandatory units, which are externally assessed and form 50% of the corresponding four-unit Advanced GCE.

This AS specification encourages candidates to develop their knowledge and understanding of computer systems, the principles of computing (including programming) and how these are applied to the solution of problems.

1.2 The Four-Unit Advanced GCE

The Advanced GCE (from September 2013) is made up of **two** mandatory units at AS and **two** further units at A2. Of the A2 units, F453 is externally assessed and F454 is internally assessed through coursework.

Additionally to the AS, this GCE specification encourages candidates to gain an understanding of systematic methods – such as the use of algorithms and test strategies, the maintenance of computer systems, and the skills associated with documenting solutions – and encourages candidates to further develop skills associated with applying this knowledge and understanding to producing computer-based solutions to real problems.

1.3 Qualification Titles and Levels

These qualifications are shown on a certificate as:

- OCR Advanced Subsidiary GCE in Computing.
- OCR Advanced GCE in Computing.

Both qualifications are Level 3 in the National Qualification Framework (NQF).

1.4 Aims

The aims of these specifications are to encourage candidates to develop:

- the capacity to think creatively, innovatively, analytically, logically and critically;
- an understanding of the organisation of computer systems, including software, hardware, data, communications and people;
- the ability to apply skills, knowledge and understanding of computing, including programming, in a range of contexts to solve problems;
- skills in project and time management;
- the capacity to see relationships between different aspects of the subject, and perceive their field of study in a broader perspective;
- an understanding of the consequences of using computers, including social, legal, ethical and other issues;
- an awareness of emerging technologies and an appreciation of their potential impact on society.

1.5 Prior Learning/Attainment

Candidates will have been assessed in IT skills at Key Stage 3 and many will have followed a course in IT at Key Stage 4. Whilst **not** assuming the full knowledge and understanding of the subject at Key Stage 4, these specifications assume that all candidates will have a basic understanding and knowledge of both the hardware and software of a standard, stand-alone computer system.

2 Summary of Content

2.1 AS Units

Unit F451: *Computer Fundamentals*

- Hardware
- Software
- Data: its presentation, structure and management
- Data transmission and networking
- Systems development life cycle
- Characteristics of information systems
- Implications of computer use

Unit F452: *Programming Techniques and Logical Methods*

- Designing solutions to problems
- The structure of procedural programs
- Data types and data structures
- Common facilities of procedural languages
- Writing maintainable programs
- Testing and running a solution

2.2 A2 Units

Unit F453: *Advanced Computing Theory*

- The function of operating systems
- The function and purpose of translators
- Computer architectures
- Data representation
- Data structures and data manipulation
- High-level language programming paradigms
- Programming techniques
- Low-level languages
- Databases

Unit F454: *Computing Project*

- Definition, investigation and analysis
- Design
- Software development and testing
- Documentation
- Evaluation
- The written report

3 Unit Content

3.1 AS Unit F451: *Computer Fundamentals*

3.1.1 Components of a Computer System

- Types of hardware
- Types of software

Candidates should be able to:

a. define the terms hardware, software, input device, storage device and output device;

b. describe the purpose of input devices, storage devices and output devices;

c. describe the different roles and functions of systems software and applications packages.

3.1.2 Software

- The systems development life cycle
- Generic applications software
- Operating systems
- User interfaces
- Utilities

Candidates should be able to:

a. describe the stages of the systems life cycle;

b. explain the importance of defining a problem accurately;

c. describe the function and purpose of a feasibility study;

d. explain the importance of determining the information requirements of a system and describe different methods of fact finding, including questionnaires, observation, and structured interviews, highlighting the advantages and disadvantages of each method;

e. describe what is involved when analysing the requirements of a system, explaining the nature of the requirements specification and its content, including current data structures, inputs, outputs and processing represented in diagrammatic form (data flow diagrams, system flowcharts), identify inefficiencies/problems in the current system;

f. describe a design specification including input design, diagrammatic depiction of the overall system, processing, data structure design and output design;

g. explain the importance of evaluating the system, and how to identify the criteria used for evaluation;

h. explain the content and importance of different types of documentation at different stages in the system life cycle, including the technical and user manuals;

i. explain the importance of system testing and installation planning;

j. explain the purpose of maintaining the system, and explain the need for system review and reassessment, understanding that software has a limited life span;

k. describe prototyping to demonstrate how a solution will appear;

l. describe the spiral and waterfall models of the systems life cycle;

m. identify the features of common applications found in business, commercial and industrial applications: eg stock control, order processing, payroll, process control, point-of-sale systems, marketing, computer-aided design (CAD), and computer-aided manufacture (CAM);

n. identify and justify generic applications software for particular application areas, eg word processing, spreadsheets, desktop publishing (DTP), presentation software, drawing packages;

o. identify and justify application areas for which custom-written applications software is appropriate;

p. describe the characteristics of knowledge-based systems;

q. describe the purpose of operating systems;

r. describe the characteristics of different types of operating systems and their uses: batch, real-time, single-user, multi-user, multi-tasking and distributed systems;

s. describe a range of applications requiring batch processing, and applications in which a rapid response is required;

t. identify and describe the purpose of different types of user interface: forms, menus, GUI, natural language and command line, suggesting the characteristics of user interfaces that make them appropriate for different types of user;

u. discuss the importance of good interface design;

v. identify and describe the purpose of a range of utilities, eg compression software, hardware drivers, anti-virus software, file handlers.

3.1.3 Data: Its representation, structure and management in information systems

- Number systems
- Data capture, preparation and entry
- Validation and verification of data
- Outputs from a system

Candidates should be able to:

a. express numbers in binary, binary-coded decimal (BCD), octal and hexadecimal;

b. describe and use two's complement and sign and magnitude to represent negative integers;

c. perform integer binary arithmetic, that is addition and subtraction;

d. explain the use of code to represent a character set (ASCII, EBCDIC and UNICODE);

e. describe manual and automatic methods of gathering and inputting data into a system, including form design, keyboard entry, voice recognition, barcodes, optical mark recognition (OMR), optical character recognition (OCR), magnetic ink character recognition (MICR), touch screens; image capture, chip and pin, sensors and remote data logging;

f. explain the techniques of validation and verification, and describe validation tests which can be carried out on data;

g. describe possible forms of output such as graphs, reports, interactive presentations, sound, video, images, animations, stating the advantages and disadvantages of each with reference to the target audience;

h. explain the procedures involved in backing up data and archiving, including the difference between data that is backed up and data that is archived.

3.1.4 Hardware

- Processor components
- Peripheral devices

Candidates should be able to:

a. describe the function and purpose of the control unit, memory unit and ALU (arithmetic logic unit) as individual parts of a computer;

b. explain the need for, and use of, registers in the functioning of the processor (Program Counter, Memory Address Register, Memory Data Register, Current Instruction Register and Accumulator);

c. explain the need for, and describe the use of, buses to convey information (Data, Address and Control buses);

d. describe the connectivity of devices (methods of hard wiring, and wireless connections);

e. describe the differences between types of primary memory and explain their uses;

f. describe the basic features, advantages, disadvantages and uses of secondary storage media;

g. describe the transfer of data between different devices and primary memory, including the uses of buffers and interrupts;

h. describe a range of common peripheral devices in terms of their features, advantages, disadvantages and uses: bar-code readers, MICR, OCR, OMR, scanners, printers, plotters, speakers, microphones, sensors, actuators (this list is indicative and any device which could be connected to the computer for input, output and storage should be considered as included);

i. describe and justify the appropriate peripheral hardware for a given application.

3.1.5 Data transmission

- Data transmission
- Circuit switching and packet switching
- Protocols
- Networking

Candidates should be able to:

a. describe the characteristics of a LAN (local area network) and a WAN (wide area network);

b. show an understanding of the hardware and software needed for a LAN and for accessing a WAN, eg the internet;

c. describe the different types of data transmission: serial and parallel; and simplex, half-duplex and duplex modes;

d. explain the relationship between bit rates and the time sensitivity of the information;

e. recognise that errors can occur in data transmission, and explain methods of detecting and correcting these errors (parity checks, the use of parity in data blocks to become self-correcting, check sums and echoes);

f. describe packet switching and circuit switching;

g. explain the difference in use of packet switching and circuit switching;

h. define the term protocol and explain the importance of a protocol to the transmission of data;

i. describe the need for communication between devices and between computers, and explain the need for protocols to establish communication links;

j. explain the need for both physical and logical protocols and the need for layering in an interface.

3.1.6 Implications of computer use

- Economic implications
- Social implications
- Legal implications
- Ethical implications
- Environmental implications

Candidates should be able to:

a. discuss changing trends in computer use and their economic, social, legal and ethical effects on society;

b. explain changes to society brought about by the use of computer systems, eg in changing leisure patterns and work expectations;

c. discuss the effects on privacy and confidentiality of data held in computer systems, and steps that can be taken to protect confidentiality;

d. understand the need for legislation governing computer use.

3.2 AS Unit F452: *Programming Techniques and Logical Methods*

In this unit candidates develop their knowledge, understanding and skills in creating computer programs to solve problems. This includes:

- creation of software designs;
- creation of actual software components using an appropriate programming language;
- methods of installing and testing software.

Candidates develop their ability to use logic to create and describe algorithms for solving a given problem, as well as interpreting and implementing designs for such algorithms.

3.2.1 Designing solutions to problems

This topic includes the specification and documentation of designs that solve a given problem. This includes the design of the input, output and user interfaces, but principally the design of the algorithms that convert the input into the output. (An awareness of the design of the data structures is also required, but is dealt with in detail in 3.2.3: *Data types and data structures*.)

- Design of the input, output and interface
- Use of structure diagrams to describe the modular nature of a solution
- Use of program flowcharts and pseudo-code to describe the steps of an algorithm
- Prototyping and Rapid Application Development (RAD)

Candidates should be able to:

a. discuss the importance of good interface design;

b. design and document data capture forms, screen layouts, report layouts or other forms of input and output (eg sound) for a given problem;

c. determine the data requirements of a program (relating to 3.2.3: *Data types and data structures*);

d. explain the advantages of designing a solution to a problem by splitting it up into smaller problems (top-down/modular design);

e. produce and describe top-down/modular designs using appropriate techniques including structure diagrams, showing stepwise refinement;

f. produce algorithms to solve problems;

g. describe the steps of an algorithm using a program flowchart;

h. describe the steps of an algorithm using pseudo-code;

i. understand, and implement algorithms and evaluate them by commenting on their efficiency, correctness and appropriateness for the problem to be solved;

j. describe the use of Rapid Application Development (RAD) as a design strategy, including prototyping and iterative development, and state its advantages and disadvantages.

3.2.2 The structure of procedural programs

This topic includes the general programming principles concerning the structure and flow of control in a procedural program.

- Basic programming constructs/control structures
- Use of subprograms/subroutines, including procedures and functions
- Recursion

Candidates should be able to:

a. define and correctly use the following terms as they apply to procedural programming: statement, subroutine, procedure, function, parameter/argument, sequence, selection, iteration/repetition, loop;

b. identify the three basic programming constructs used to control the flow of execution, ie sequence, selection and iteration;

c. understand and use selection in pseudo-code and a procedural programming language, including the use of IF statements and CASE/SELECT statements;

d. understand and use iteration in pseudo-code and a procedural programming language, including the use of count-controlled loops (FOR-NEXT loops) and condition-controlled loops (WHILE-ENDWHILE and REPEAT-UNTIL loops);

e. understand and use nested selection and iteration statements;

f. understand, create and use subroutines (procedures and functions), including the passing of parameters and the appropriate use of the return value of functions

g. identify and use recursion to solve problems; show an understanding of the structure of a recursive subroutine, including the necessity of a stopping condition;

h. trace the execution of a recursive subroutine including calls to itself;

i. discuss the relative merits of iterative and recursive solutions to the same problem.

3.2.3 Data types and data structures

Candidates should learn to select appropriate data types and design appropriate data structures (including files) to solve a given problem, and justify their choice. They should learn to use the facilities of a procedural programming language to manipulate files.

- Data types: integer, real, Boolean, character, string
- Data structures: array (one- and two-dimensional), record
- Storing, retrieving and searching for data in files

Candidates should, when writing a program in a procedural language, be able to:

a. define different data types, eg numeric (integer, real), Boolean, character and string; select and use them appropriately in their solutions to problems;

b. define and use arrays (one- and two-dimensional) for solving simple problems, including initialising arrays, reading data into arrays and performing a simple serial search on a one-dimensional array;

c. explain the advantages and disadvantages of different data types and data structures for solving a given problem;

d. design and implement a record format;

e. define different modes of file access: serial, sequential, indexed sequential and random; and justify a suitable mode of file access for a given example;

f. store, retrieve and search for data in files;

g. estimate the size of a file from its structure and the number of records;

h. use the facilities of a procedural language to perform file operations (opening, reading, writing, updating, inserting, appending and closing) on files of different access modes as appropriate.

3.2.4 Common facilities of procedural languages

This topic includes operations, built-in functions and other facilities that are common to most procedural languages.

- Assignment statements
- Arithmetic, relational and Boolean operations
- String manipulation
- Input and output facilities

Using an appropriate procedural programming language, candidates should be able to:

a. understand and use assignment statements;

b. understand arithmetic operators including operators for integer division (+, -, *, /, MOD and DIV) and use these to construct expressions;

c. understand a range of relational operators, eg =, <, <=, >, >= and <> and use these to construct expressions;

d. understand the Boolean operators AND, OR and NOT and use these to construct expressions;

e. understand the effects of the precedence of standard operators and the use of parentheses to alter the order of precedence;

f. evaluate expressions containing arithmetic, relational and Boolean operators and parentheses;

g. understand and use a range of operators and built-in functions for string manipulation, including location (LOCATE), extraction (LEFT, MID, RIGHT), comparison, concatenation, determining the length of a string (LENGTH) and converting between characters and their ASCII code (ASCII and CHAR);

h. understand that relational operations on alphanumeric strings depend on character codes of the characters and explain the results of this effect (eg why ‘XYZ’ < ‘abc’, ‘2’ > ‘17’ and ‘3’ <> ‘3.0’);

i. input and validate data;

j. output data onto screen/file/printer, formatting the data for output as necessary.

3.2.5 Writing maintainable programs

This topic includes basic techniques for writing code that will be easy to maintain.

- Declaring and using variables and constants
- Self-documented code, including identifiers, annotation and formatting
- Modularised code

Using an appropriate procedural programming language, candidates should be able to:

a. define, understand and use the following terms correctly as they apply to programming: variable, constant, identifier, reserved word/keyword;

b. explain the need for good program-writing techniques to facilitate the ongoing maintenance of programs;

c. declare variables and constants, understanding the effect of scope and issues concerning the choice of identifier (including the need to avoid reserved words/keywords);

d. select and use meaningful identifier names and use standard conventions to show the data types and enhance readability;

e. use declared constants to improve maintainability;

f. initialise variables appropriately, before using them;

g. create appropriately modularised programs (following a top-down/modular design as produced in 3.2.1: *Designing solutions to problems*) making effective use of subroutines to improve maintainability;

h. annotate the code with comments so that the logic of the solution can be followed;

i. use indentation and formatting to show clearly the control structures within the code.

3.2.6 Testing and running a solution

This topic requires candidates to understand and be able to deal with the variety of errors that may occur during development, including testing their solution and ensuring that it is robust. They should also have an awareness of the process of translating, installing and executing their solution.

- Types of programming errors
- Testing strategies and test data
- Debugging
- Installation and execution

When developing software to solve a problem, candidates should be able to:

a. describe types of errors in programs (syntax, logic and run-time errors) and understand how and when these may be detected;

b. identify why/where an error may occur in an algorithm and state how the algorithm may be corrected;

c. describe testing strategies including white box testing, black box testing, alpha testing, beta testing and acceptance testing;

d. select suitable test data for a given problem, including normal, borderline and invalid data;

e. perform a dry run on a given algorithm, using a trace table;

f. describe the use of a range of debugging tools and facilities available in procedural programming languages including translator diagnostics, break points, stepping, and variable checks;

g. describe the purpose of an installation routine in the delivered version of the program.

3.3 A2 Unit F453: *Advanced Computing Theory*

In this unit, candidates acquire extensive knowledge of computing theory. Traditional computing forms a basis for knowledge which is developed further to include modern trends.
Note: Candidates are **not** expected to use any particular form to present algorithms, but should be able to write procedural algorithms in some form. A detailed knowledge of the syntax of programming languages is **not** required.

3.3.1 The function of operating systems

- Features of operating systems
- Interrupt handling
- Scheduling, job queues and priorities
- Memory management
- Spooling
- Modern personal computer operating systems

Candidates should be able to:

a. describe the main features of operating systems, for example memory management, and scheduling algorithms;

b. explain how interrupts are used to obtain processor time and how processing of interrupted jobs may later be resumed, (typical sources of interrupts should be identified and any algorithms and data structures should be described);

c. define and explain the purpose of scheduling, job queues, priorities and how they are used to manage job throughput;

d. explain how memory is managed in a typical modern computer system (virtual memory, paging and segmentation should be described along with some of the problems which could occur, such as disk thrashing);

e. describe spooling, explaining why it is used;

f. describe the main components of a typical desktop PC operating system, including the file allocation table (FAT) and how it is used, and the purpose of the boot file.

3.3.2 The function and purpose of translators

- Types of translators and their use
- Lexical analysis
- Syntax analysis
- Code generation and optimisation
- Library routines

Candidates should be able to:

a. describe the need for, and use of, translators to convert source code to object code;

b. understand the relationship between assembly language and machine code;

c. describe the use of an assembler in producing machine code;

d. describe the difference between interpretation and compilation;

e. describe the purpose of intermediate code in a virtual machine;

f. describe what happens during lexical analysis;

g. describe what happens during syntax analysis, explaining how errors are handled;

h. explain the code generation phase and understand the need for optimisation;

i. describe the use of library routines.

3.3.3 Computer architectures

- Von Neumann architecture
- Registers – purpose and use
- Fetch-execute cycle
- Other machine architectures

Candidates should be able to:

a. describe classic Von Neumann architecture, identifying the need for, and the uses of, special registers in the functioning of a processor;

b. describe, in simple terms, the fetch/decode/execute cycle, and the effects of the stages of the cycle on specific registers;

c. discuss co-processor, parallel processor and array processor systems, their uses, advantages and disadvantages;

d. describe and distinguish between Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC) architectures.

3.3.4 Data representation

- Floating point binary
- Normalisation of floating point binary numbers

Candidates should be able to:

a. demonstrate an understanding of floating point representation of a real binary number;

b. normalise a real binary number;

c. discuss the trade-off between accuracy and range when representing numbers.

3.3.5 Data structures and data manipulation

- Implementation of data structures, including stacks, queues and trees
- Searching, merging and sorting

Candidates should be able to:

a. explain how static data structures may be used to implement dynamic data structures;

b. describe algorithms for the insertion, retrieval and deletion of data items stored in stack, queue and tree structures;

c. explain the difference between binary searching and serial searching, highlighting the advantages and disadvantages of each;

d. explain how to merge data files;

e. explain the differences between the insertion and quick sort methods, highlighting the characteristics, advantages and disadvantages of each.

3.3.6 High-level language programming paradigms

- Types of languages and typical applications
- Features of different types of language

Candidates should be able to:

a. identify a variety of programming paradigms (low-level, object-oriented, declarative and procedural);

b. explain, with examples, the terms object-oriented, declarative and procedural as applied to high-level languages, showing an understanding of typical uses;

c. discuss the concepts and, using examples, show an understanding of data encapsulation, classes and derived classes, and inheritance when referring to object-oriented languages;

d. understand the purpose of the Unified Modelling Language (UML);

e. interpret class, object, use case, state, sequence, activity and communication diagrams;

f. create class, object, use case and communication diagrams;

g. discuss the concepts and, using examples, show an understanding of backtracking, instantiation, predicate logic and satisfying goals when referring to declarative languages.

3.3.7 Programming techniques

- Standard programming techniques
- Methods for defining syntax

Candidates should be able to:

a. explain how functions, procedures and their related variables may be used to develop a program in a structured way, using stepwise refinement;

b. describe the use of parameters, local and global variables as standard programming techniques;

c. explain how a stack is used to handle procedure calling and parameter passing;

d. explain the need for, and be able to create and apply, BNF (Backus-Naur form) and syntax diagrams;

e. explain the need for reverse Polish notation;

f. convert between reverse Polish notation and infix form of algebraic expressions using trees and stacks.

3.3.8 Low-level languages

- Use of computer architecture
- Features of low-level languages

Candidates should be able to:

a. explain the concepts and, using examples, demonstrate an understanding of the use of the accumulator, registers, and program counter;

b. describe immediate, direct, indirect, relative and indexed addressing of memory when referring to low-level languages;

c. discuss the concepts and, using examples, show an understanding of mnemonics, opcode, operand and symbolic addressing in assembly language to include simple arithmetic operations, data transfer and flow-control.

3.3.9 Databases

- Database design
- Normalisation and data modelling
- Methods and tools for analysing and implementing database design
- Database management system (DBMS)
- Use of Structured Query Language (SQL)

Candidates should be able to (in general and within a context of a scenario):

a. describe flat files and relational databases, explaining the differences between them;

b. design a simple relational database to the third normal form (3NF), using entity-relationship (E-R) diagrams and decomposition;

c. define and explain the purpose of primary, secondary and foreign keys;

d. describe the structure of a DBMS including the function and purpose of the data dictionary, data description language (DDL) and data manipulation language (DML);

use SQL to define tables and views, insert, select and delete data and to produce reports.

3.4 A2 Unit F454: *Computing Project*

In this unit, candidates develop their knowledge and understanding of computer systems and the skills studied in AS F451: *Computer fundamentals* and AS F452: *Programming techniques and logical methods* and use the high-level programming techniques studied in AS F452. This project is a substantial piece of work, requiring analysis and design over an extended period of time, which is organised, evaluated and presented in a report.

Candidates choose, in conjunction with their teacher, a well-defined user-driven problem of an appropriate size which enables them to demonstrate their skills in Analysis, Design, Software Development, Testing, Implementation, Documentation and Evaluation, and their interrelation; and to give a completed overall system that solves the problem.

3.4.1 Definition, investigation and analysis

Explanation of the problem to be solved, the user's requirements and how they were obtained.

Evidence of the development of a requirement specification and a clear statement of requirements.

- Define a task
- Research the task and needs of the end-user(s)
- Record findings
- Analyse findings
- Project plan, including specification of the requirements:
  – user
  – hardware
  – software

Candidates should be able to:

a. define the nature of the task to be carried out;

b. identify methods by which to investigate the problem, including questionnaires, observation and structured interviews;

c. record information/data and gather sample documents currently used;

d. identify the current processes and current data;

e. analyse the data and processes: candidates will be expected to use appropriate techniques such as structure diagrams/data flow diagrams/system flowcharts to illustrate their analysis;

f. specify any perceived problems and inefficiencies apparent from discussions with the user and the analysis work carried out;

g. derive the user and information requirements of a system;

h. specify and justify the required hardware;

i. specify and justify the required software.

3.4.2 Design

Detailed system design including:

- data structures
- input-output format
- processes involved.

A clear test strategy including test data needs to be developed to ensure the system meets the design objectives.

A clear design specification:

- Specify the objectives relating them to the requirements specification
- Input design
- Output design
- Design and test plan, and test data
- Data structures/variables
- Algorithms
- Test algorithms

Candidates should be able to:

a. specify the objectives of the proposed system and relate them to the requirements specification;

b. design and document data capture forms and/or screen layouts, drawing up detailed mock-ups of the proposed interface;

c. design and document report layouts, screen displays and/or other forms of output (for example, audio output), drawing up detailed mock-ups of the proposed interface;

d. identify, develop and document a test strategy for the design;

e. select suitable test data for the design;

f. design and document, using appropriate techniques (for example, data flow diagrams), the data structures necessary to solve the inefficiencies/problems indicated in the requirements specification;

g. design and document an algorithm/pseudo-code/top-down diagram or other form of process model;

h. using appropriate techniques, test that the algorithms meet the design objectives.

3.4.3 Software development and testing

Develop a software solution and, using the test plan developed in 3.4.2: *Design*, show that the system works with valid, invalid and borderline data (or, if it does not, under which circumstances it fails).

Test plan clearly cross-referenced to evidence that the system has been tested during development and implementation.

Evidence of user testing.

- Software development
- Alpha testing
- Response to the results of testing
- Beta testing
- Response to the results of beta testing
- Modularisation of code
- Code documentation
- Use of modules, data structures and objects
- In-code documentation
- Code structure
- Applying the test plan and data

Candidates should be able to:

a. develop the rules/methods/algorithms of a design using a programming language;

b. develop the data structures of the design using the appropriate features of a programming language;

c. develop inputs/outputs using the features of a programming language;

d. test and refine the software solution during development, illustrating how the software solution evolves;

e. test the software solution with the user, providing documented evidence that the solution works;

f. produce suitable modular code with full annotation and a description of how the modules combine to create the solution;

g. produce detailed output from the testing, cross-referencing as appropriate with the test plan.

3.4.4 Documentation

The software solution should be self-documenting, with suitable on-screen help and support for the end user(s).

All necessary supporting documentation required to enable the end user to make effective use of the solution.

- User documentation

Candidates should be able to:

a. develop detailed and appropriate user documentation.

3.4.5 Evaluation

Discussion of the degree of success in meeting the original objectives as specified in the requirements specification, including an evaluation of the project management – to include an evaluation of the ease of the use of the package, acceptability to the users, the choice of task, the project plan and desirable extensions/modifications.

- Evaluate results against the requirement specification showing that the objectives in 3.4.2: *Design* have been satisfied
- Evaluate the project management
- Evaluate user responses
- Identify the good and bad points of the final system, including any limitations and necessary extensions or modifications to the system

Candidates should be able to:

a. evaluate the final system against the criteria described in the requirements specification;

b. evaluate the users' responses to testing the system;

c. critically evaluate the project management;

d. identify the good and bad points of the final system highlighting any limitations and necessary extensions or modifications to the system, indicating how these could be carried out.

The written report

See Appendix B for the assessment criteria for this report.

4 Schemes of Assessment

4.1 AS GCE Scheme of Assessment

AS GCE Computing (H047)	
AS F451: *Computer Fundamentals*	
50% of the total AS GCE marks 1.5 h written paper 100 marks	Candidates are required to answer **all** questions.
AS F452: *Programming Techniques and Logical Methods*	
50% of the total AS GCE marks 1.5 h written paper 100 marks	Candidates are required to answer **all** questions.

4.2 Advanced GCE Scheme of Assessment

Advanced GCE Computing (H447)	
AS units as above, being 25% of the total Advanced GCE marks.	
A2 F453: *Advanced Computing Theory*	
30% of the total Advanced GCE marks 2 h written paper 120 marks	Candidates are required to answer **all** questions.
A2 F454: *Computing Project*	
20% of the total Advanced GCE marks Coursework 80 marks	The project is a substantial piece of work, requiring analysis and design over an extended period of time, which is organised, evaluated and presented in a report. Candidates choose, in conjunction with their teacher, a well-defined user-driven problem of an appropriate size which enables them to demonstrate their skills in Analysis, Design, Software Development, Testing, Implementation, Documentation and Evaluation, and their interrelation, and to give a completed overall system that solves the problem. **Assessment Criteria**: Please refer to Appendix B.

4.3 Unit Order

The normal order in which the unit assessments could be taken is AS Units F451 and F452 in the first year of study, leading to an AS GCE award; then A2 Units F453 and F454, leading to the Advanced GCE award.

It should be noted that AS Units F451 and F452 **must** be delivered before A2 Units F453 and F454.

Also, AS F451: *Computer fundamentals* includes some of the theory which underpins AS F452: *Programming techniques and logical methods*, including the software life cycle and data representation, an awareness of this theory is assumed in F452. Thus F452 **must** be delivered after, or in parallel with, the relevant sections of F451 so that candidates can use their practical experience of programming to reinforce the material studied.

Similarly, some sections of A2 F453: *Advanced computing theory* need to be studied before undertaking A2 F454: *Computing project*, although it is envisaged that A2 F454 is conducted in parallel with the study of A2 F453.

4.4 Unit Options (at AS/A2)

There are no optional units in the AS GCE specification; for AS GCE Computing candidates must take AS Units F451 and F452.

There are no optional units in the Advanced GCE specification; for Advanced GCE Computing candidates take AS Units F451 and F452, *and* A2 Units F453 and F454.

4.5 Synoptic Assessment (A Level GCE)

Synoptic assessment is included in both A2 units. It draws on both assessment objectives and is designed to test candidates' understanding of the connections between different elements of the subject.

Synoptic assessment in computing requires candidates to make connections between different areas of computing represented in the Advanced GCE specification.

In particular, candidates are required to draw on their knowledge and understanding of information, software, hardware, and communications, when demonstrating the skills associated with analysis, design, implementation and evaluation of computer-based systems. For example, this could be applying knowledge and understanding of the methods of organising and structuring information when designing a system.

4.6 Assessment Availability

There is one examination series each year in June.

From 2014, both AS units and A2 units will be assessed in June only.

4.7 Assessment Objectives

There are two assessment objectives, AO1 and AO2.

Candidates are expected to demonstrate the following in the context of the content described.

AO1 Demonstrate knowledge and understanding

- describe and explain the purpose and characteristics of a range of computing applications and show an understanding of the characteristics of computer systems (hardware, software and communication) which allow effective solutions to be achieved;
- describe and explain the need for, and the use of, various forms of data organisation and processing to support the requirements of a computer-based solution;
- describe and explain the systematic development of high quality solutions to problems and the techniques for implementing such solutions, including the use of a programming language where appropriate;
- comment critically on the consequences of current uses of computing, including economic, social, legal and ethical issues.

AO2 Demonstrate skills

- analyse a problem and identify the parts that are appropriate for a computer-based solution;
- select, justify and apply appropriate techniques and principles to develop data structures and algorithms for the solution of problems;
- design, implement and document an effective solution using appropriate hardware and software, including the use of a programming language.

AO weightings in AS GCE

Unit	% of AS GCE		
	AO1(%)	AO2(%)	Total(%)
AS F451: *Computer Fundamentals*	35–45	10–15	50
AS F452: *Programming Techniques and Logical Methods*	20–25	20–30	50
	55–70	30–45	100

AO weightings in Advanced GCE

Unit	% of Advanced GCE		
	AO1(%)	AO2(%)	Total(%)
AS F451: *Computer Fundamentals*	15–18	7.5–10	25
AS F452: *Programming Techniques and Logical Methods*	10–15	9–15	25
A2 F453: *Advanced Computing Theory*	17.5–20	10–12.5	30
A2 F454: *Computing Project*	0–4.5	16–20	20
	42.5–57.5	42.5–57.5	100

4.8 Quality of Written Communication

Quality of Written Communication is assessed in all units and credit may be restricted if communication is unclear.

Candidates need to:

- ensure that text is legible and that spelling, punctuation and grammar are accurate so that the meaning is clear;
- select and use a form and style of writing appropriate to the purpose and complex subject matter;
- organise information clearly and coherently, using specialist vocabulary when appropriate.

5 Technical Information

5.1 Making Unit Entries

Please note that centres must be registered with OCR in order to make any entries, including estimated entries. It is recommended that centres apply to OCR to become a registered centre well in advance of making their first entries. Centres must have made an entry for a unit in order for OCR to supply the appropriate forms or moderator details for coursework.

It is essential that unit entry codes (the four-figure alpha-numeric codes given in brackets at the end of the unit title) are quoted in all correspondence with OCR. See Sections 4.1 and 4.2 for these unit entry codes.

5.2 Making Qualification Entries

Candidates must enter for qualification certification separately from unit assessment(s). If a certification entry is **not** made, no overall grade can be awarded.

Candidates may enter for:

- AS GCE certification (entry code H047).
- Advanced GCE certification (entry code H447).

A candidate who has completed all the units required for the qualification, and who did not request certification at the time of entry, may enter for certification either in the same examination series (within a specified period after publication of results) or in a later series.

AS GCE certification is available from June 2014.
Advanced GCE certification is available from June 2014.

5.3 Grading

All GCE units are awarded a–e. The Advanced Subsidiary GCE is also awarded on the scale A–E. The Advanced GCE is awarded on the scale A-E with access to A*. To be awarded an A*, candidates will need to achieve a grade A on their full A level qualification and an A* on the aggregate of their A2 units.. Grades are reported on certificates. Results for candidates who fail to achieve the minimum grade (E or e) will be recorded as unclassified (U or u) and this is **not** certificated.

A Uniform Mark Scale (UMS) enables comparison of candidates' performance across units and across series and enables candidates' scores to be put on a common scale for aggregation purposes. The two-unit AS GCE has a total of 200 *uniform* marks and the four-unit Advanced GCE has a total of 400 *uniform* marks.

OCR converts the candidate's *raw* mark for each unit to a *uniform* mark. The maximum *uniform* mark for any unit depends on that unit's weighting in the specification. In these Computing specifications, the four units of the Advanced GCE specification have UMS weightings of 25%, 25%, 30% and 20% (and the two units of the AS GCE specification have UMS weightings of 50% and 50%). The UMS totals are 100, 100, 120 and 80 respectively. Each unit's *raw* mark grade boundary equates to the *uniform* mark boundary at the same grade. Intermediate marks are converted on a pro-rata basis.

Uniform marks correspond to *unit* grades as follows.

(Advanced GCE) Unit Weighting	Maximum Unit Uniform Mark	Unit Grade					
		a	b	c	d	e	u
30%	120	120–96	95–84	83–72	71–60	59–48	47–0
25%	100	100–80	79–70	69–60	59–50	49–40	39–0
20%	80	80–64	63–56	55–48	47–40	39–32	31–0

OCR adds together the unit *uniform* marks and compares these to pre-set boundaries (see the table below) to arrive at *qualification* grades.

Qualification	Qualification Grade					
	A	B	C	D	E	U
AS GCE	200–160	159–140	139–120	119–100	99–80	79–0
Advanced GCE	400–320	319–280	279–240	239–200	199–160	159–0

5.4 Result Enquiries and Appeals

Under certain circumstances, a centre may wish to query the grade available to one or more candidates or to submit an appeal against an outcome of such an enquiry. Enquiries about unit results must be made immediately following the series in which the relevant unit was taken.

For procedures relating to enquires on results and appeals, centres should consult the *Administration Guide for General Qualifications* and the document *Enquiries about Results and Appeals: Information and Guidance for Centres* produced by the Joint Council. Copies of the most recent editions of these papers can be obtained from OCR.

5.5 Shelf-life of Units

Individual unit results, prior to certification of the qualification, have a shelf-life limited only by that of the qualification.

5.6 Unit and Qualification Resits

There is no restriction on the number of times a candidate may resit each unit before entering for certification for an AS GCE or Advanced GCE.

Candidates may enter for the full qualifications an unlimited number of times.

5.7 Guided Learning Hours

AS GCE Computing requires **180** guided learning hours in total.
Advanced GCE Computing requires **360** guided learning hours in total.

5.8 Code of Practice/Subject Criteria/Common Criteria Requirements

These specifications comply in all respects with the current *GCSE, GCE, GNVQ and AEA Code of Practice,* as available on the QCA website; the subject criteria for GCE Computing; and *The Statutory Regulation of External Qualifications 2004*.

5.9 Arrangements for Candidates with Particular Requirements

For candidates who are unable to complete the full assessment or whose performance may be adversely affected through no fault of their own, teachers should consult the *Access Arrangements and Special Consideration: Regulations and Guidance Relating to Candidates who are Eligible for Adjustments in Examinations* produced by the Joint Council. In such cases advice should be sought from OCR as early as possible during the course.

5.10 Prohibited Qualifications and Classification Code

Candidates who enter for the OCR GCE specifications may not enter for any other GCE specification with the certification title *Computing* in the same examination series.

Every specification is assigned to a national classification code indicating the subject area to which it belongs.

Centres should be aware that candidates who enter for more than one GCE qualification with the same classification code will have only one grade (the highest) counted for the purpose of the Schools and College Achievement and Attainment Tables.

The classification code for these specifications is 2610.

5.11 Coursework Administration/Regulations

Supervision and Authentication

As with all coursework, teachers must be able to verify that the work submitted for assessment is the candidate's own work. Sufficient work must be carried out under direct supervision to allow the teacher to authenticate the coursework marks with confidence.

Submitting Marks to OCR

Centres must have made an entry for a unit in order for OCR to supply the appropriate forms or moderator details for coursework. Coursework administration documents are sent to centres on the basis of estimated entries. Marks may be submitted to OCR either on the computer-printed Coursework Mark Sheets (MS1) provided by OCR (sending the top copy to OCR and the second copy to their allocated moderator) or by EDI (centres using EDI are asked to print a copy of their file and sign it before sending it to their allocated moderator).

Deadline for the receipt of coursework marks is:
15 May for the June series.

The awarding body must require centres to obtain from each candidate a signed declaration that authenticates the coursework they produce as their own. For regulations governing coursework, centres should consult the *Administration Guide for General Qualifications.* Further copies of the coursework administration documents are available on the OCR website (www.ocr.org.uk).

Standardisation and Moderation

All internally-assessed coursework is marked by the teacher and internally standardised by the centre. Marks must be submitted to OCR by the agreed date, after which postal moderation takes place in accordance with OCR procedures.

The purpose of moderation is to ensure that the standard for the award of marks in internally-assessed coursework is the same for each centre, and that each teacher has applied the standards appropriately across the range of candidates within the centre.

The sample of work that is submitted to the moderator for moderation must show how the marks have been awarded in relation to the marking criteria.

Minimum Coursework Required

If a candidate submits no work for a unit, then the candidate should be indicated as being absent from that unit on the coursework mark sheets submitted to OCR. If a candidate completes any work at all for that unit then the work should be assessed according to the criteria and marking instructions and the appropriate mark awarded, which may be zero.

6 Other Specification Issues

6.1 Overlap with other Qualifications

There is a small degree of overlap between the content of these specifications and those for Advanced GCE ICT.

6.2 Progression from these Qualifications

In today's workplace, those with knowledge and skills in computing have the opportunity to pursue new and exciting careers and to be instrumental in the conception of computer systems that increasingly shape work and leisure activities.

To meet these career challenges, students must be self-reliant as well as good communicators and problem solvers. They require interpersonal, academic and technical skills, and must demonstrate an ability to work independently and as part of a team. They also need to develop an ethical approach to the use of computers. These specifications provide a focus to develop these skills, while ensuring that students acquire a sound knowledge of computing.

It is envisaged that students will utilise the skills and knowledge of computing in one of three ways. Firstly, to provide a general understanding of the use of computer technology and systems, which will inform their decisions and support their participation in an increasingly technologically dependent society. Secondly, to provide the necessary skills and knowledge to seek employment in areas that utilise computing, where they may develop their skills and knowledge further through practical experience and training. Thirdly, students may choose to continue to develop their knowledge and understand of computing through entry to higher education, where this qualification will provide a useful foundation for further study of computing or more specialist aspects of computing.

6.3 Key Skills Mapping

These specifications provide opportunities for the development of the Key Skills of *Communication*, *Application of Number*, *Information Technology*, *Working with Others*, *Improving Own Learning and Performance* and *Problem Solving* at Levels 2 and/or 3. However, the extent to which this evidence fulfils the Key Skills criteria at these levels will be totally dependent on the style of teaching and learning adopted for each unit.

The following table indicates where opportunities *may* exist for at least some coverage of the various Key Skills criteria at Levels 2 and/or 3 for each unit.

Unit	C				AoN			IT			WwO			IOLP			PS		
	.1a	.1b	.2	.3	.1	.2	.3	.1	.2	.3	.1	.2	.3	.1	.2	.3	.1	.2	.3
F451	✓	-	✓	✓	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F452	✓	-	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F453	✓	-	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F454	✓	-	✓	✓	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

6.4 Spiritual, Moral, Ethical, Social, Legislative, Economic and Cultural Issues

These specifications offer opportunities that can contribute to an understanding of these issues in the following ways.

These specifications encourage candidates to explore the spiritual, moral, ethical, social, legislative and cultural aspects of the introduction of computer-based solutions to problems through a study of their effects on society.

Through candidates' study of Units F451 and F452, they have an opportunity to develop their understanding of spiritual, moral, ethical, social, legal and cultural issues. These units consider issues such as changing leisure patterns and work practices, privacy and confidentiality of data held in systems, opportunities for access to information, and environmental issues.

6.5 Sustainable Development, Health and Safety Considerations and European Developments

These specifications support these issues, consistent with current EU agreements.

6.6 Avoidance of Bias

OCR has taken great care in the preparation of these specifications and assessment materials to avoid bias of any kind.

6.7 Language

These specifications and associated assessment materials are in English only.

6.8 Disability Discrimination Act Information Relating to these Specifications

AS/A levels often require assessment of a broad range of competences. This is because they are general qualifications and, as such, prepare candidates for a wide range of occupations and higher level courses.

The revised AS/A level qualification and subject criteria were reviewed to identify whether any of the competences required by the subject presented a potential barrier to any disabled candidates. If this was the case, the situation was reviewed again to ensure that such competences were included only where essential to the subject. The findings of this process were discussed with disability groups and with disabled people.

Reasonable adjustments are made for disabled candidates in order to enable them to access the assessments. For this reason, very few candidates will have a complete barrier to any part of the assessment. Information on reasonable adjustments is found in *Access Arrangements and Special Consideration Regulations and Guidance Relating to Candidates who are Eligible for Adjustments in Examinations* produced by the Joint Council (refer to Section 5.9 of this specification).

Candidates who are still unable to access a significant part of the assessment, even after exploring all possibilities through reasonable adjustments, may still be able to receive an award. They would be given a grade on the parts of the assessment they have taken and there would be an indication on their certificate that not all of the competences have been addressed. This will be kept under review and may be amended in the future.

Appendix A: Performance Descriptions

Performance descriptions have been created for all GCE subjects. They describe the learning outcomes and levels of attainment likely to be demonstrated by a representative candidate performing at the A/B and E/U boundaries for AS and A2.

In practice most candidates will show uneven profiles across the attainments listed, with strengths in some areas compensating in the award process for weaknesses or omissions elsewhere. Performance descriptions illustrate expectations at the A/B and E/U boundaries of the AS and A2 as a whole; they have not been written at unit level.

Grade A/B and E/U boundaries should be set using professional judgement. The judgement should reflect the quality of candidates' work, informed by the available technical and statistical evidence. Performance descriptions are designed to assist examiners in exercising their professional judgement. They should be interpreted and applied in the context of individual specifications and their associated units. However, performance descriptions are not designed to define the content of specifications and units.

The requirement for all AS and A level specifications to assess candidates' quality of written communication will be met through one or more of the assessment objectives.

The performance descriptions have been produced by the regulatory authorities in collaboration with the awarding bodies.

AS performance descriptions for computing

	Assessment Objective 1	Assessment Objective 2
Assessment Objectives	**Knowledge and understanding** Candidates should be able to: • Describe and explain the purpose and characteristics of a range of computing applications and show an understanding of the characteristics of computer systems (hardware, software and communication) that allow effective solutions to be achieved • Describe and explain the need for and the use of various forms of data organisation and processing to support the requirements of a computer-based solution • Describe and explain the systematic development of high-quality solutions to problems and the techniques for implementing such solutions, including the use of a programming language Comment critically on the consequences of current uses of computing, including economic, social, legal and ethical issues.	**Skills** Candidates should be able to: • Analyse a problem and identify the parts that are appropriate for a computer-based solution • Select, justify and apply appropriate techniques and principles to develop data structures and algorithms for the solution of problems • Design, implement and document an effective solution using appropriate hardware and software, including the use of a programming language.

	Assessment Objective 1	Assessment Objective 2
AS A/B boundary Performance Descriptions	Candidates characteristically: • understand the purpose and characteristics of a range of computing applications • demonstrate knowledge of the characteristics of the main hardware, software and communication components of computer systems and how they allow effective solutions to be achieved • understand the need to organise data appropriately and process it efficiently in order to solve problems using computers • understand the need to adopt a systematic approach when developing high quality solutions to problems • show knowledge of appropriate techniques to implement solutions, including the use of a programming language • demonstrate a critical understanding of the consequences of current uses of computing, including economic, social, legal and ethical issues.	Candidates characteristically: • use subject-specific terminology appropriately and accurately • analyse a complex problem and identify the parts that are appropriate for a computer-based solution • derive most of the user and information requirements of a system to solve a problem • select and use appropriate techniques to develop a solution with suitable data structures and algorithms • choose and justify appropriate hardware and software with which to solve a problem, including the use of a programming language • design an effective solution and document it appropriately • implement a workable solution, testing and documenting it appropriately.
AS E/U boundary Performance Descriptions	Candidates characteristically: • demonstrate some understanding of the purpose and characteristics of a limited range of computing applications • show a limited knowledge of the characteristics of the main hardware, software and communication components of computer systems • have some understanding of the need to organise data appropriately and process it efficiently in order to solve problems using computers • demonstrate some understanding of the need to adopt a systematic approach when developing high-quality solutions to problems • show a limited knowledge of appropriate techniques to implement solutions, including the use of a programming language • have a limited understanding of the consequences of current uses of computing, including some economic, social, legal and ethical issues.	Candidates characteristically: • use subject-specific terminology • analyse a problem and identify parts that are appropriate for a computer-based solution • derive some of the user and information requirements of a system to solve a problem • select and use some appropriate techniques to develop a solution with generally suitable data structures and algorithms • choose hardware and software with which to solve a problem, including the use of a programming language • design a simple solution, and document it to a limited extent • produce a solution, with limited testing and documentation.

A2 performance descriptions for computing

	Assessment Objective 1	Assessment Objective 2
Assessment Objectives	**Knowledge and understanding** Candidates should be able to: • Describe and explain the purpose and characteristics of a range of computing applications and show an understanding of the characteristics of computer systems (hardware, software and communication) that allow effective solutions to be achieved • Describe and explain the need for and the use of various forms of data organisation and processing to support the requirements of a computer-based solution • Describe and explain the systematic development of high-quality solutions to problems and the techniques for implementing such solutions, including the use of a programming language Comment critically on the consequences of current uses of computing, including economic, social, legal and ethical issues.	**Skills** Candidates should be able to: • Analyse a problem and identify the parts that are appropriate for a computer-based solution • Select, justify and apply appropriate techniques and principles to develop data structures and algorithms for the solution of problems • Design, implement and document an effective solution using appropriate hardware and software, including the use of a programming language.

	Assessment Objective 1	Assessment Objective 2
A2 A/B boundary Performance Descriptions	Candidates characteristically: • demonstrate a thorough understanding of the purpose and characteristics of a wide range of computing applications • show an extensive knowledge of the characteristics of a wide range of hardware, software and communication components of computer systems • have a thorough understanding of the need to organise data appropriately and process it efficiently in order to solve problems using computers • demonstrate a thorough understanding of the need to adopt a systematic approach when developing high quality solutions to problems • show an extensive knowledge of appropriate techniques to implement solutions, including the advanced use of a programming language • have an in-depth understanding of the consequences of current uses of computing, including a wide range of economic, social, legal and ethical issues.	Candidates characteristically: • use subject-specific terminology appropriately and accurately • analyse a complex problem and identify the parts that are appropriate for a computer-based solution • derive the user and information requirements of a system to solve a problem • select and use appropriate techniques to develop an effective solution with suitable data structures and algorithms • choose and justify the most appropriate hardware and software with which to solve a problem, including the use of a programming language • design an effective and efficient solution and document it thoroughly • implement an efficient solution, testing and documenting it thoroughly.
A2 E/U boundary Performance Descriptions	Candidates characteristically: • demonstrate a basic understanding of the purpose and characteristics of some computing applications • show a basic knowledge of the characteristics of a range of hardware, software and communication components of computer systems • understand the need to organise data appropriately and process it efficiently in order to solve problems using computers • understand the need to adopt a systematic approach when developing solutions to problems • demonstrate a basic knowledge of appropriate techniques to implement solutions, including the advanced use of a programming language • show some understanding of the consequences of current uses of computing, including a range of economic, social, legal and ethical issues.	Candidates characteristically: • use a basic range of subject-specific terminology • analyse a fairly straightforward problem and identify the parts that are appropriate for a computer-based solution • derive some of the user and information requirements of a system to solve a problem • select and use appropriate techniques to develop a solution with suitable data structures and algorithms • choose and justify some appropriate hardware and software with which to solve a problem, including the use of a programming language • design a workable solution and document it to some extent • implement an workable solution, testing and documenting it to some extent

Appendix B: Guidance on Setting and Marking
A2 Unit F454: *Computing Project*

Guidance on Setting Computing Projects

A project should:

- allow candidates to demonstrate their knowledge and understanding of computer systems and the skills in the assessment objectives;

- encourage the sensible use of computers to produce a system, using an appropriate programming language, which is non-trivial, has a substantial coded element and will solve a given problem sensibly within the constraints of the resources available to the candidate.

Note: It may be difficult to quantify the scale of a problem fully. Indeed, before the analysis and design stages have been completed it is not possible to know the depth of the solution. Candidates should note that the emphasis should be on the choice of a real-life problem. Centres are reminded that if they are in any doubt about the suitability of a problem on the grounds of degree of rigour, type of problem or any other criteria, they should contact the board at the earliest opportunity for advice.

- show the successful completion of a whole task from its definition, involving a third party user, to its acceptance and evaluation by that user. Projects that involve much repetitive design, analysis or especially implementation, leading to unwieldy reports, are to be discouraged;

- involve all elements of the skills of definition, analysis, design, development, testing, implementation, documentation and evaluation. The project must provide sufficient opportunity for the candidate to demonstrate the programming skills developed as part of the AS syllabus and in the A2 F451. Projects need not be 'stand alone': the enhancement or modification of an existing system are acceptable, provided that all these elements are covered; this may in fact lead to work that is more likely to reflect a real-world situation;

- involve a third party user, who will provide information for the analysis, use the implemented solution and contribute towards its evaluation. The third party is likely to be a user (or potential user) of a computer system for a designated purpose. Whilst a teacher could act as the third party user, this arrangement is far from ideal. Candidates should be encouraged to look beyond school life into either the businesses and companies in the community of the surrounding area or to focus groups. The emphasis is on analysing an existing system or area for development, and producing a computer-based solution to fit the needs of the user.

Candidates should make the final choice of context for the project, although the supervisor should give guidance about project suitability. This should include guidance on the appropriateness of implementing a stand-alone or networked small computer system or other available facility. In a well-organised project, the candidate focuses on the production of an overall system analysis and design. The solution implementation must include the use of a high-level programming language. The additional use of pre-written modules and toolkits, applications software and programmable packages may be appropriate. Brief descriptions of the programming languages and any additional software packages used, together with reasons for their selection, should be included in the report.

For the coded element of the solution, the candidate should:

- annotate listings;
- explain each selection of the program with appropriate algorithm descriptions, which should be language independent;
- define variables by name, type and function where appropriate;
- define clearly, and identify the purpose of, functions subroutines and procedures.

Where the solution has used additional software packages not involving programming, candidates should:

- explain each section of the solution with appropriate algorithm descriptions;
- define the purpose and interrelationship of modules within the system;
- clearly annotate the results produced.

Test data should be devised and used systematically to test the package thoroughly. The choice of test data used, and the reason for choice, should be included. A description of the methods of testing should also be included, together with evidence of testing.

The projects should contain the title, a contents list, a description and justification of investigation, analysis, design and methods used, an evaluation and bibliography. Pages should be clearly numbered. Appropriate evidence of development, testing and implementation must support the report, for example screen dumps or photographs of screen layouts and printouts, paper-based user documentation and suitable evidence from the third party user to show that the system has been developed satisfactorily. Any evidence submitted to demonstrate the development of the solution must be able to be assessed without the use of any specific hardware or software.

Candidates should choose a well-defined user-driven problem of an appropriate size, which enables them to demonstrate their skills in Analysis, Design, Development, Testing, Implementation, Documentation and Evaluation. The project should involve the skills attained by studying the other modules of this specification, specifically the programming skills studied in the other units.

The computing projects must involve programming, and in some cases may also involve the selection and installation of hardware.

Guidance on Marking Computing Projects

Computing projects are assessed as follows:

(a)	Definition, Investigation and Analysis	**[14 marks]**
(b)	Design	**[16 marks]**
(c)	Software Development and Testing	**[30 marks]**
(d)	Documentation	**[10 marks]**
(e)	Evaluation	**[10 marks]**

(a) Definition, Investigation and Analysis [14 marks]

(i) Definition – nature of the problem to be investigated [3 marks]

A candidate should not expect the examiner to be familiar with the theory and practice in the area of the chosen system. There should be a brief description of the end user (for example, firm or business) involved; and the current methods used or details of the area for development that may form the basis of the project. A clear statement of the origins and form of any relevant data should be given. At this stage, the exact scope of the project may not be known and it may lead to the arrangement of an interview with the user.

3 marks	Excellent description with all elements present.
2 marks	Some description of both the stages of study and end user involved.
1 mark	Vague description of the end user or area for development.

(ii) Investigation and Analysis [11 marks]

This section is the 'systems analysis'. The question is not how a system performs detailed tasks, but rather how the project progresses from the original data to the results. The candidate should describe how the user requirements were ascertained (including detailed planning of the investigation). The results of the investigation should be recorded accurately and analysed carefully to show how the candidate has arrived at the requirements specification. The specification must be detailed and should include the user, hardware and software requirements of the proposed solution.

9–11 marks	Excellent user involvement with detailed recording of the user's requirements. All other items must be present, showing a thorough analysis of the system to be computerised. A detailed requirements specification, including full justification for the approach and hardware and software requirements, has been produced.
6–8 marks	Good user involvement and recording of the data collection methods. Most of the necessary items have been covered. However, one or two items have been omitted. A requirements specification is present with some attempt to justify the approach based on the results of the investigations but with some omissions, eg hardware and software requirements.
3–5marks	Some evidence that an attempt has been made to identify the end-user requirements and some recording of it has been made. Attempts at some of the other items have been made. An attempt has been made to develop a requirement specification but with little attempt to justify this based on the results of the investigation.
1–2 marks	Some elements have been discussed but with little or no user involvement.

(b) Design [16 marks]

(i) Nature of the solution [6 marks]

A detailed systems design (including appropriate diagrams) should be produced and agreed with the users. Proposed record, file and data structures should be described and design limitations should be included. Design of data capture forms, input formats (with examples of screen layouts) and output formats should be included. A detailed summary of the aims and objectives should also be included. These items are the design specifications, which should be agreed with the user.

5–6 marks	A clear set of objectives with a detailed and complete design specification, which is logically correct. There is evidence to show that the end user has seen and agreed these designs. There are also detailed written descriptions of any processes/modules and a clear, complete definition of any data structures. The specification is sufficient for someone to pick up and develop an end result using the software and hardware specified in the requirements specification.
3–4 marks	The major objectives of the new system have been adequately summarised, *but omissions have been made*. There is a brief outline of a design specification, including mock-ups of inputs and outputs, and the process model has been described (including a diagram: structure diagram, data flow diagram or system flowchart). There is some evidence that the end user has seen these designs. However, there is a lack of completeness with omissions from the process model, inputs and outputs. Data structures have been identified but there may be inadequate detail.
1–2 marks	Some vague discussion of what the system will do, with a brief diagrammatic representation of the new system.

(ii) Algorithms [5 marks]

Detailed language-independent algorithms should be developed together with evidence that the algorithms have been tested to ensure they meet the design objectives.

5 marks	A complete set of algorithms with evidence to show that they have been assessed by the candidate to show that they will meet the design specification. (Evidence should show how these algorithms form a complete solution and that they have been tested for functionality using appropriate techniques.)
3–4 marks	A complete set of detailed algorithms covering the system as specified.
1–2 marks	Some vague algorithms detailing how the system will be developed.

(iii) Test strategy [5 marks]

A detailed test strategy and plan, together with appropriate test data, should be developed and documented. It is vital to produce test cases and to show that they work. To do this, it is necessary not only to have test data, but to know what the expected results are with that data.

5 marks	A detailed test strategy and plan covering all aspects of the system with data to test under normal, extreme and abnormal circumstances.
3–4 marks	A detailed test strategy and a plan covering several aspects of the system but with inadequate data to effectively test the system, eg data covers only normal circumstances or covers only a limited part of the design specification.
1–2 marks	A vague discussion of how the system might be tested.

(c) Software Development and Testing [30 marks]

(i) Software Development [16 marks]

A technical description of how the solution relates to the design specification produced and agreed with the user should be included. It is the responsibility of the candidate to produce evidence of their development work. This section must show how the candidate tested each section during development and the responses to this alpha testing. The code must be documented adequately to explain its function and there must be clear evidence of how modular code has been used to develop the final solution.

13–16 marks	There is complete evidence showing how the solution was developed using suitable alpha testing at each stage to inform the process. The modular code is fully annotated indicating clearly the purpose of each section and the interrelationship between the sections. The developed solution fulfils all of the design specification.
9–12 marks	Program listings are provided in the form of printouts. Data structures are illustrated as part of the listings where appropriate, detailing their purpose. There is sufficient annotation evident to illustrate how the solution was developed for a particular purpose and indicate the purpose of sections of code. The code will be modular and there will be good evidence to show how testing was used during the development process to inform each stage. The developed solution fulfils the design specification but there are some minor flaws in the solution.
5–8 marks	Program listings are provided in the form of printouts. Data structures are illustrated as part of the listings where appropriate, detailing their purpose. There is some annotation evident to illustrate how the solution was developed and some limited evidence that some testing took place during development. The developed solution has significant flaws and only partially fulfils the design specification. The code may be linear but with some annotation indicating how the code relates to the problem and some limited evidence of alpha testing.
1–4 marks	Program listings are provided in the form of printouts but with no annotation or evidence of alpha testing. The developed solution does not fulfil the design specification. There is some evidence of system development.

(ii) Testing [14 marks]

An attempt should be made to show that all parts of the system have been tested, including those sections dealing with unexpected or invalid data as well as extreme cases. Showing that many other cases of test data are likely to work – by including the outputs that they produce – is another important feature. Evidence of testing is essential. The beta testing should cover all aspects of the test plan produced in the design section, which should cover all aspects of the design specification. The examiner must be left in no doubt that the system actually works in the target environment. This evidence may be in the form of hardcopy output (possibly including screen dumps), photographs or any format that does not require access to any specific hardware or software. The end user(s) must be involved in this process and evidence of end-user testing is required.

11–14 marks	The testing covers as many different paths through the system as is feasible, including valid, invalid and extreme cases. The testing covers all aspects of the design specification and the test plan from the design section. There is clear evidence of end-user testing.
8–10 marks	There is evidence of testing covering most aspects of the design specification but with omissions, eg test data does not include erroneous data for all tests or there is limited evidence of end-user testing.

5–7 marks There is limited evidence of testing based on a badly developed test plan with clear omissions. There is no description of the relationship between the test plan and the testing in evidence.

1–4 marks A collection of hardcopy test run outputs with no clear link to the test plan and covering few aspects of the system. No evidence of end-user testing.

(d) Documentation [10 marks]

Quality of Written Communication is assessed in this documentation. Much of the technical documentation will have been produced as a by-product of design and development work and also as part of writing up the report to date. The software solution should also include sufficient on-screen help to enable the end user to make use of the system. Some additional supporting documents will be necessary including initial set-up, getting started and troubleshooting guides, to ensure the end user can implement the solution.

8–10 marks Candidates will provide detailed and accurate documentation. The documentation will be well presented, in a structured and coherent format. The documentation will cover all aspects of the system, with no omissions, including installation, typical use, troubleshooting, and backup. The on-screen help and supplementary documentation makes a complete guide to the solution and is well presented and easy to follow. Subject-specific terminology will be used accurately and appropriately. There will be few, if any, errors of spelling, grammar and punctuation.

4–7 marks Candidates will provide clear documentation. The documentation will be well presented. There is clear on-screen support to enable the end user to use the system. The supporting documentation and on-screen help is well presented and covers most aspects of the system operation with only one or two omissions, eg troubleshooting or backup. Some subject-specific terminology will be used. There may be occasional errors of spelling, grammar and punctuation.

1–3 marks Candidates will provide superficial documentation, with weak supplementary user documentation covering few aspects of the system. The information will be poorly expressed and limited technical terms will be used. Errors of grammar, punctuation and spelling may be intrusive.

(e) Evaluation [10 marks]

(i) Discussion of the degree of success in meeting the original objectives [4 marks]

This discussion should demonstrate the candidate's ability to evaluate the effectiveness of the completed system. The original objectives stated in the requirements specification should be matched to the achievements, taking into account the limitations. User evaluation is also essential and should arise from direct user evaluation.

3–4 marks A full discussion, taking each objective mentioned in **(b) (i)** and explaining the degree of success in meeting them (indicating where in the project evidence can be found to support this), or reasons why they were not met.

1–2 mark Some discussion about a number of objectives, but some omissions or inadequate explanation of success or failure.

0 marks No discussion present.

(ii) Evaluate the user's response to the system [3 marks]

It is important that the user is not assumed to be an expert in computer jargon, so some effort must be made to ensure that the system is user-friendly. It will be assumed that the user will have considerable knowledge of the underlying theory of the business or area being computerised. Clarity of menus, clear on-screen help and easy methods of inputting data are all examples of how the system can be made user-friendly. Here marks are awarded for the degree of satisfaction that the user indicates in the acceptance procedure. Could the system or its results be used? Was the system specification achieved? Do any system faults still exist? The candidate should evaluate the user's response to the final version of the system.

3 marks	The user indicates that the system could be used but there are some faults which need to be rectified. The candidate provides a detailed discussion of how these inadequacies may be dealt with. OR A fully functional user-friendly system has been produced. The user indicates that the system fully meets the specification given in section **(a)**, and there are no known faults in the system.
2 marks	The system is, in the main, user-friendly, but there is room for improvement (eg no on-screen help has been provided). The user indicates that the system could be used but there are some faults which need to be rectified. The candidate has made some limited attempt to discuss how these inadequacies may be dealt with.
1 mark	The system does not meet the design specification and the end user is not able to make use of the system. The candidate briefly discusses these issues in terms of their project management.

(iii) Desirable extensions [3 marks]

As a result of completing the system, the candidate should identify the good and bad points of the final system, highlighting any limitations and necessary extensions to the system, and indicating how the extensions could be carried out.

3 marks	The candidate clearly portrays the good and bad points of the system indicating the limitations, possible extensions and how to carry out the extensions.
2 marks	The candidate clearly identifies good and bad points and any limitations.
1 mark	The candidate identifies the obvious good points of the system and possibly some bad points or limitations.