fortismere

Computing

AS

Contents

Specification followed

OCR Level 3 Advanced Subsidiary GCE in Computer Science (H046)

Specification

Version 1: First Assessment 2016

Content of Computing principles (Component 01)

This component will introduce learners to the internal workings of the Central Processing Unit (CPU), the exchange of data and will also look at software development, data types and legal and ethical issues. It is expected that learners will draw on this underpinning content when studying computational thinking and developing programming techniques.

You will be expected to apply the criteria, in the tables below, in different contexts including current and future uses of the technologies.

1.1. The characteristics of contemporary processors, input, output and storage devices

Components of a computer and their uses	**Confident**	**Developing**	**With difficulty**
1.1.1 Structure and function of the processor a) The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: How this relates to assembly language programs. b) The fetch-decode-execute cycle, including its effect on registers. c) The factors affecting the performance of the CPU, clock speed, number of cores, cache. d) Von Neumann, Harvard and contemporary processor architecture.			
1.1.2 Types of processor a) The differences between, and uses of, CISC and RISC processors. b) Multicore and parallel systems.			
1.1.3 Input, output and storage a) How different input output and storage devices can be applied as a solution of different problems. b) The uses of magnetic, flash and optical storage devices. c) RAM and ROM. d) Virtual storage.			

1.2. Software and software development

Types of software and the different methodologies used to develop software	Confident	Developing	With difficulty
1.2.1 Operating Systems a) The need for, function and purpose of operating systems. b) Memory management (paging, segmentation and virtual memory). c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the fetch decode execute cycle. d) Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time. e) Distributed, embedded, multi-tasking, multi-user and real time operating systems. f) BIOS. g) Device drivers. h) Virtual machines, any instance where software is used to take on the function of a machine including executing intermediate code or running an operating system within another.			
1.2.2 Applications generation a) The nature of applications, justifying suitable applications for a specific purpose. b) Utilities. c) Open source vs Closed source. d) Translators: interpreters, compilers and assemblers.			
1.2.3 Introduction to programming a) Procedural programming language techniques: program flow variables and constants procedures and functions arithmetic, Boolean and assignment operators string handling file handling. b) Assembly language (including following and writing simple programs with Little Man Computer).			

1.3. Exchanging data

How data is exchanged between different systems	Confident	Developing	With difficulty
1.3.1 Databases a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling. b) Methods for capturing, selecting, managing and exchanging data.			
1.3.2 Networks a) Characteristics of networks and the importance of protocols and standards. b) Internet structure: The TCP/IP Stack. DNS Protocol layering. LANs and WANs. Packet and circuit switching. c) Client-server and Peer to peer.			
1.3.3 Web Technologies a) HTML, CSS and JavaScript. b) Lossy v lossless compression.			

1.4. Data types, data structures and algorithms

How data is represented and stored within different structures. Different algorithms that can be applied to these structures	**Confident**	**Developing**	**With difficulty**
1.4.1 Data Types a) Primitive data types, integer, real/floating point, character, string and Boolean. b) Represent positive integers in binary. c) Use of sign and magnitude and two's complement to represent negative numbers in binary. d) Addition and subtraction of binary integers. e) Represent positive integers in hexadecimal. f) Convert positive integers between Binary Hexadecimal and denary. g) Positive and negative real numbers using normalised floating point representation. h) How character sets (ASCII and UNICODE) are used to represent text.			
1.4.2 Data Structures a) Arrays (of up to 3 dimensions), records, lists, tuples. b) The properties of stacks and queues.			
1.4.3 Boolean Algebra a) Define problems using Boolean logic. See appendix 5e. b) Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. c) Use logic gate diagrams and truth tables.			

1.5. Legal, moral, ethical and cultural issues			
The individual (moral), social (ethical) and cultural opportunities and risks of digital technology. Legislation surrounding the use of computers and ethical issues that can or may in the future arise from the use of computers	**Confident**	**Developing**	**With difficulty**
1.5.1 Computing related legislation a) The Data Protection Act 1998. b) The Computer Misuse Act 1990. c) The Copyright Design and Patents Act 1988. d) The Regulation of Investigatory Powers Act 2000.			
1.5.2 Ethical, moral and cultural issues a) The individual (moral), social (ethical) and cultural opportunities and risks of digital technology: Computers in the workforce Automated decision making Artificial intelligence Environmental effects Censorship and the Internet Monitor behaviour Analyse personal information Piracy and offensive communications Layout, colour paradigms and character sets.			

Content of Algorithms and problem solving (Component 02)

This component will incorporate and build on the knowledge and understanding gained in the Computing principles component. In addition, you should:

- understand what is meant by computational thinking
- understand the benefits of applying computational thinking to solving problems
- be able to use algorithms to describe problems.

2.1. Elements of computational thinking

Understand what is meant by computational thinking	**Confident**	**Developing**	**With difficulty**
2.1.1 Thinking abstractly a) The nature of abstraction. b) The need for abstraction. c) The differences between an abstraction and reality. d) Devise an abstract model for a variety of situations.			
2.1.2 Thinking ahead a) Identify the inputs and outputs for a given situation. b) Determine the preconditions for devising a solution to a problem. c) The need for reusable program components.			
2.1.3 Thinking procedurally a) Identify the components of a problem. b) Identify the components of a solution to a problem. c) Determine the order of the steps needed to solve a problem. d) Identify sub-procedures necessary to solve a problem.			
2.1.4 Thinking logically a) Identify the points in a solution where a decision has to be taken. b) Determine the logical conditions that affect the outcome of a decision. c) Determine how decisions affect flow through a program.			

2.2. Problem solving and programming			
How computers can be used to solve problems and programs can be written to solve them	**Confident**	**Developing**	**With difficulty**
2.2.1 Programming techniques a) Programming constructs: sequence, iteration, branching. b) Global and local variables. c) Modularity, functions and procedures, parameter passing by value and reference. d) Use of an IDE to develop/debug a program.			
2.2.2 Software Development a) Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. b) The relative merits and drawbacks of different methodologies and when they might be used. c) Writing and following algorithms. d) Different test strategies, including black and white box testing and alpha and beta testing. e) Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.			

2.3 Algorithms			
The use of algorithms to describe problems and standard algorithms	**Confident**	**Developing**	**With difficulty**
2.3.1 Algorithms a) Analysis and design of algorithms for a given situation. b) Standard algorithms (bubble sort, insertion sort, binary search and linear search). c) Implement bubble sort, insertion sort. d) Implement binary and linear search. e) Representing, adding data to and removing data from queues and stacks. f) Compare the suitability of different algorithms for a given task and data set.			