March 2022 Mock COMPUTER SCIENCE

Paper 1			
			Revision pages
1.1 Systems Architecture	1.1.1: Architecture of the CPU What actions occur at each stage of the fetch-execute cycle. The role/purpose of each component and what it managers, stores, or controls during the fetch-execute cycle. The purpose of each register, what it stores (data or address). The difference between storing data and an address.	Purpose of the CPU: (The fetch-execute cycle) Common CPU components and their functions (ALU, CU, cache, registers) Von Neumann architecture (MAR- Memory Address Register), MDR (Memory Data Register), Program Counter and Accumulator.	Pg 2-3
1.2 Memory and Storage	1.2.1: Primary Storage (Memory) Why computers have primary storage (how this usually consists of RAM/ROM). The difference between RAM and ROM. The purpose of RAM, ROM, Virtual memory.	The need for primary storage RAM, ROM, Virtual memory	Pg 6-7
	1.2.2 Secondary Storage Why do computers have secondary storage? Differences between each type of storage device/medium. Compare advantages/disadvantages for each storage device.	Common types of storage • Optical • Magnetic • Solid state Suitable storage devices The advantages and disadvantages of different storage devices and storage media relating to these characteristics. (Capacity, speed, portability, durability, reliability, cost)	Pg 8-9
	1.2.3: Units Why data must be stored in binary format. Calculate required storage capacity for a given set of files. Calculate file sizes of sound, images and text files. Sound file size = sample rate x duration (s) x bit depth image file size = colour depth x image height (px) x image width (px) text file size = bits per character x number of characters	The units of data storage; bit, nibble, byte, kilobyte, megabyte, gigabyte, terabyte, petabyte). How data needs to be converted into a binary format to be processed by a computer. Data capacity and calculation of data capacity requirements.	Pg 11
	1.2.4: Data Storage	Numbers:	Pg 12-20

	Conversion of any numbers in these ranges to another number base. The differences between and impact of each character set. The effect on an image size and quality when changing colour depth and resolution.	• Convert denary to binary and vice versa. • Binary addition • Convert denary to hexadecimal numbers and vice versa. • Binary shift Characters: • Use of binary to represent characters • Character set • ASCII • Unicode Images: • Images are represented by pixels, in binary • Metadata • Effect of colour depth and resolution Sound • How sound can be sampled and stored in digital form • The effect of sample rate, duration and bit depth on (the playback quality, the size of a sound file)	
	1.2.5: Compression Advantages and disadvantages of each type of compression. Effects on the file for each type of compression.	Need for compression. Lossy and Lossless compression	Pg 21
1.3 Computer networks, connections and protocols	1.3.1: Networks and topologies The tasks performed by each piece of hardware. DNS (Domain Name Server's role in the conversion of a URL to an IP address.	Types of networks (LAN, WAN) Factors that affect the performance of networks. The hardware needed to connect stand-alone computers into Local Area Networks. The internet as a worldwide collection of computer networks.	Pg 23-25
	1.3.2: Wired and wireless networks, protocols and layers Compare benefits and drawbacks of wired versus wireless connection. The principle of encryption to secure data across network connections. That different types of protocols are used for different purposes.	Modes of connection (wired- ethernet), (wireless- wifi, Bluetooth) Encryption IP addressing and MAC addressing Standards Common protocols (TCP/IP, HTPP, HTTPS, FTP, POP, IMAP, SMTP)	Pg 28-32
1.4 Network Security	1.4.1: threats to computer systems and networks	Forms of attack (malware, social engineering brute force attacks, denial of service attacks,	Pg 34

	Threats posed to devices/systems.	data interception and theft, the concept of SQL injection).	
	1.4.2: Identifying and preventing vulnerabilities Understanding of how to limit the threats posed in 1.4.1. Understanding methods to remove vulnerabilities.	Common prevention methods (penetration testing, anti-malware software, firewalls, user access levels, passwords, encryption, physical security).	Pg 35
1.6 Ethical, legal, cultural and environment al impacts of digital technology	1.6.1: Ethical, legal, cultural and environmental impact Technology introduces ethical, legal, cultural, environmental and privacy issues. The purpose of each piece of legislation and the specific actions it allows or prohibits. Features of open source and proprietary.	Impacts of digital technology on wider society. (ethical, legal, cultural, environmental, privacy) Legislation relevant to Computer Science. (The Data Protection Act, Computer Misuse Act, Copyright Designs and Patents Act, Software licenses (is open sources and proprietary)	Pg 40-44

Paper 2			
2.1.1 Computational thinking	Understanding these principles and how they are used to solve problems	Principles of computational thinking • Abstraction • Decomposition • Algorithmic thinking	Pg 47
2.1.2 Designing, creating, and refining algorithms	Produce simple diagrams to show: • The structure of a problem • Subsections and their links to other subsections	Identify the inputs, processes, and outputs for a problem • Structure diagrams • Create, interpret, correct, complete, and refine algorithms using: • Flowcharts	Pg 49-53

	• Complete, write or refine an algorithm using the techniques listed • Identify syntax/logic errors in code and suggest fixes • Create and use trace tables to follow an algorithm	• Reference language/high-level programming language • Identify common errors • Trace tables	
2.1.3 Searching and sorting algorithms	Understand the main steps of each algorithm Understand any pre-requisites of an algorithm Apply the algorithm to a data set Identify an algorithm if given the code	Standard searching algorithms: • Binary search • Linear search Standard sorting algorithms: • Bubble sort • Merge sort • Insertion sort	Pg 54-60
2.2.1 Programming fundamentals	Understanding of each technique Recognise and use operators	The use of variables, constants, operators, inputs, outputs and assignments The use of the three basic programming constructs used to control the flow of a program: • Sequence • Selection • Iteration (count- and condition-controlled loops) The common arithmetic operators The common Boolean operators AND, OR and NOT	Pg 61-65
2.2.2 Data types	Ability to choose suitable data types for data in each scenario Understand that data types may be temporarily changed through casting, and where this may be useful	The use of data types including: • Integer • Real • Boolean • Character and string • Casting	Pg 66
2.2.3 Additional programming techniques	Ability to manipulate strings, including: • Concatenation • Slicing Arrays as fixed length or static structures Use of 2D arrays to emulate database tables of a collection of fields, and records	The use of basic string manipulation The use of basic file handling operations: • Open • Read • Write • Close The use of records to store data The use of SQL to search for data The use of arrays (or equivalent) when solving problems, including	Pg 67-74

	The use of functions The use of procedures The use of the following within functions and procedures: • local variables/constants • global variables/constants • arrays (passing and returning) SQL commands: • SELECT • FROM • WHERE Be able to create and use random numbers in a program	both one-dimensional (1D) and two-dimensional arrays (2D) How to use sub programs (functions and procedures) to produce structured code Random number generation	
2.3.1 Defensive design	Understanding of the issues a programmer should consider ensuring that a program caters for all likely input values Understanding of how to deal with invalid data in a program Authentication to confirm the identity of a user Practical experience of designing input validation and simple authentication (e.g. username and password) Understand why commenting is useful and apply this appropriately	Defensive design considerations: • Anticipating misuse • Authentication • Input validation Maintainability: Use of sub programs Naming conventions Indentation Commenting	Pg 78-79
2.3.2 Testing	The difference between testing modules of a program during development and testing the program at the end of production Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated Logic errors as errors which produce unexpected output	The purpose of testing Types of testing: • Iterative • Final/terminal Identify syntax and logic errors Selecting and using suitable test data: • Normal • Boundary • Invalid/Erroneous • Refining algorithms	Pg 80

	Normal test data as data which should be accepted by a program without causing errors Boundary test data as data of the correct type which is on the very edge of being valid Invalid test data as data of the correct data type which should be rejected by a computer system Erroneous test data as data of the incorrect data type which should be rejected by a computer system Ability to identify suitable test data for a given scenario Ability to create/complete a test plan		
2.4.1 Boolean Logic	Knowledge of the truth tables for each logic gate Recognition of each gate symbol Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios Ability to work with more than one gate in a logic diagram	Simple logic diagrams using the operators AND, OR and NOT Truth tables Combining Boolean operators using AND, OR and NOT Applying logical operators in truth tables to solve problems	Pg 82-83
2.5.1 Languages	Knowledge of the truth tables for each logic gate Recognition of each gate symbol Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios Ability to work with more than one gate in a logic diagram	Characteristics and purpose of different levels of programming language: • High-level languages • Low-level languages The purpose of translators The characteristics of a compiler and an interpreter	Pg 84
2.5.2 The Integrated Development Environment (IDE)	Knowledge of the tools that an IDE provides How each of the tools and facilities listed can be used to help a programmer develop a program	Common tools and facilities available in an Integrated Development Environment (IDE): • Editors • Error diagnostics • Run-time environment • Translators	Pg 85

	Practical experience of using a range of these tools within at least one IDE		