A Level Computer Science

Curriculum Intent 2021-2022

Core aims of the subject at Key Stage 5

Computer Science is a demand subject in a globally competitive world. It has become an ever-growing part of human life, affecting many aspects of a person's day. Computer systems are embedded ubiquitously in everyday devices, smart phones, washing machines, heating systems and vehicles, as our world embraces "The Internet of Things". Computer scientists have an impact on how our society advances by developing and maintaining these systems: whether it be for our home, work, learning or entertainment environments. Computer Science is an exciting and rapidly evolving subject that offers excellent employment prospects and well-paid careers.

The curriculum ensures learners have sufficient knowledge to stay safe online and use computers safely in life. We want students to not only understand how to use technology effectively, safely and responsibly, but also how technology is developed and constantly redeveloped into new and exciting tools. The curriculum continues to focus on developing resilient learners who are able to recover from mistakes and effectively solve problems. This will help develop a lifelong effect of learning and how to develop themselves further and prepare for the future.

The course is theoretical content, however there is also a lot of opportunities to build projects and challenging opportunities to develop programming skills.

The curriculum is developed so that students are taught the principles of problem solving and computation, which prepares you to solve the problems of tomorrow, by developing learner's knowledge, skills and understanding through key computational concepts and experience. Develop understanding for all the technology that surrounds them by not just understanding how computer systems work, but how to put this knowledge to use through programming and problem solving. Building on this knowledge and understanding, students are equipped to use information technology to create programs, systems and a range of content. Students will also analyse problems in computational terms and devise creative solutions by designing, writing, testing and evaluating programs. This also ensures that students become digitally literate – able to use, and express themselves and develop their ideas through, information technology – at a level suitable for the future workplace and as active participants in a digital world. We endeavour to make the curriculum as fun and interesting as possible with a high level of challenge by offering breadth and depth of experiences for the students. Our aim is to ensure that you develop and achieve ICT capability that is directly transferable, not only to other subjects, but also beyond, developing a wide range of digital skills that will prepare you for the future.

The rationale of the KS5 curriculum is for students to develop the mind-set of a computer scientist built upon the foundations at KS3 and KS4. Learners have the opportunity to develop their capability, creativity and knowledge in computer science. The topics have been chosen based

on the GCSE specification and are planned to dedicate time to each of them, allowing plenty of time for revision and future preparation for the exams.

This course is aimed to prepare students with the knowledge, skills and confidence to be ready for the next stage of their life either in further education or career.

The curriculum provides challenges and new experiences in computing, digital literacy and digital media (regardless of their prior knowledge of using computers). Over the 2 years, students will continue in the development of programming skills and effectively apply the knowledge learnt in earlier Algorithm and Programming units. Throughout refences to key events and developments through the history of technology using role models from all aspects of society to be inspirational and motivational for students.

We aim to enable students to develop a love for the subject and an understanding that there are no limits to their own development in programming and IT. To enthuse students to have an understanding far deeper than the interface that they currently operate. This is done by offering challenging opportunities and personal development.

Our vision is to provide quality computing education to equip students to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems.

Irresistible and enriching learning by a wide range of educational experiences to engage, cultivate and extend lifelong effect of learning. All students take part in challenging opportunities by completing The Bebras challenge and CyberFirst CyberStart competitions. The CyberFirst Competition provides a fun and challenging environment to inspire the next generation of young people to consider a career in cyber security.

Students are given the opportunity to enter a range of National Competitions such as and CyberFirst and Cyber Centurion events.

Due to the forever changing world of technology the curriculum and skills need is taken into account. Staff are involved with the local primary schools and the whole community including Computing at School and exam boards to ensure that the curriculum is achievable and forward thinking, to ensure that students are equipped for their future pathways.

Assessment

Please see website for the formal internal assessment record.
Paper 1 On-screen exam, subject coverage:

- Fundamentals of programming
- Fundamentals of data structures
- Fundamentals of algorithms
- Theory of computation
- Written exam - students will be issued a preliminary material, a skeleton program (available in each of the programming languages) and, where appropriate, test data, for use in the exam. 2 hours 30 minutes
- 40% of A level exam

Paper 2 Written exam, subject coverage:

- Fundamentals of data representation
- Fundamentals of computer systems
- Fundamentals of computer organisation and architecture
- Consequences of uses of computing
- Fundamentals of communication and networking
- Fundamentals of databases
- Big Data
- Fundamentals of functional programming
- legal and environmental impacts of digital technology on wider society, including issues of privacy
- Written exam: 2 hours 30 minutes
- 40% of A level exam
- Compulsory short-answer and extended-answer questions
- NEA Programming project
- 20% of A level

Homework

Repl.it set homework for coding and theory practice
Clubs and/or intervention
Lunchtime drop-in sessions available. Extra revision available after school as needed.

Parental/Carer support

Parents/Carers can find the subject content and specification at: https://filestore.aqa.org.uk/resources/computing/specifications/AQA-7516-7517-SP-2015.PDF

Helpful sources of information

https://www.aqa.org.uk/subjects/computer-science-and-it/as-and-a-level/computer-science-7516-7517 - exam board subject homepage
https://repl.it – programming practice challenges
https://w3schools.com – tutorials, references for programming languages

www.codeacademy.com – learn technical skills in an interactive environment

Connections to future pathways

Careers: Software Developer, Information Security Analysts, Computer systems analyst, Computer and information systems manager, Computer and information research scientists, Computer network architect, Network and computer systems administrators, Database administrator, Web developer, Computer support specialist.

Future learning: Higher Apprenticeship, Degree

Year 12 Overview

Term	Knowledge	Assessment	Connections to learning
Autumn 1	**Computational thinking** Rationale: Understanding how computational thinking is the basis for problem solving. Understand the need to have different data types in theory will help when using text-based programming languages.		
	➢ Different data types and how they are used ➢ Basic arithmetic operations in a typical programming language ➢ Basic string handling operations, variables and constants ➢ Pseudocode solutions to simple problems ➢ Relational operators and use of Boolean operations AND, OR, NOT ➢ Nested selection statements ➢ Use of three different types of iterative statement: WHILE, REPEAT and FOR ➢ Random number generation ➢ Structured approach to program design and construction ➢ Construct and use hierarchy charts when designing programs ➢ Data structure, 1- and 2-dimensional arrays in the design of solutions to	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class) ➢ Knowledge Quizzes. ➢ Application of knowledge understanding and skills using pseudocode and programming	**4.1 Fundamentals of programming** 4.1.1 Programming 4.1.1.1 Data types 4.1.1.2 Programming concepts 4.1.1.3 Arithmetic operations 4.1.1.6 Constants and Variables 4.1.1.7 String handling 4.1.2 Programming paradigms 4.1.1.4 Relational operators 4.1.1.5 Boolean operators 4.1.1.8 Random number generation 4.1.2.3 Object-orientated programming (in brief) **4.2 Fundamentals of data structures** 4.2.1 Data structures and abstract types 4.2.1.1 Data structures 4.2.1.2 Single and multi-dimensional arrays 4.2.1.3 Fields, records and files ➢

	simple problems and the advantages of the structured approach		
	➢ "Computational thinking" and the skills involved ➢ Strategies for problem-solving, simple logic problems and checking solutions ➢ Concept of abstraction and examples ➢ The purpose of testing, a test plan using test data covering normal (typical), boundary and erroneous data, hand-trace algorithms ➢ Finite state machine, uses, draw and interpret simple state transition diagrams ➢ Draw a state transition table for a finite state machine with no output	➢ Teacher/pupil questioning ➢ Application of knowledge understanding and skills using pseudocode and programming ➢ Exam style question practice (homework's and in class)	**4.4 Theory of computation** 4.4.1 Abstraction and automation 4.4.1.2 Following and writing algorithms 4.4.1.1 Problem Solving 4.4.2 Regular languages 4.4.2.1 Finite state machines (FSMs) with and without output **4.13 Systematic approach to problem solving** 4.13.1 Aspects of software development ➢ 4.13.1.4 Testing
Autumn 2	**Programming Concepts** Rationale: Students need a theoretical understanding of all the topics in this section for the exams even if the programming language(s) they have been taught do not support all topics. Understand Understanding the fundamentals of programming concepts.		
	➢ Subroutines, their uses and advantages ➢ Subroutines that return values to the calling routine ➢ Arguments/parameters to pass data within programs ➢ Contrast the use of local and global variables ➢ Define the terms field, record, file ➢ Read from and write to a text file and read from and write to a binary file ➢ Use of exception handling in a program ➢ Categorise numbers as natural, integer, rational, irrational, real or ordinal.	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class) ➢ Application of knowledge understanding and skills using pseudocode and programming	**4.1 Fundamentals of programming** 4.1.1 Programming 4.1.1.10 Subroutines 4.1.1.11 Parameters of subroutines 4.1.1.12 Returning a value from a subroutine 4.1.1.13 Local variables in a subroutine 4.1.1.14 Global variables in a programming language 4.1.1.9 Exception handling **4.2 Fundamentals of data structures** 4.2.1 Data structures and abstract types 4.2.1.3 Fields, records and files **4.5 Fundamentals of data representation**

	➢ Understand how the base of a number affects the format of the value it represents. ➢ Convert between decimal, binary and hexadecimal number systems. ➢ Data is stored and processed in a computer system ➢ Bits and bytes, and use of names, symbols and corresponding powers of 2 for binary prefixes (Ki, Mi,Ti etc.) ➢ Differentiate between the character code of a decimal digit and its pure binary representation ➢ ASCII and Unicode coding systems and why Unicode was introduced ➢ Methods used for error-checking and correction ➢ Add and multiply together two unsigned binary numbers ➢ Convert between signed binary and decimal and vice versa ➢ Represent positive and negative numbers in two's complement and specify the range of n bits ➢ Subtraction using two's complement ➢ Numbers with a fractional part can be represented in binary ➢ Fixed point binary to represent a real number in a given number of bits ➢ Fractional part can be represented in floating point form ➢ Normalise un-normalised floating point numbers with positive or negative mantissas ➢ Calculate the absolute and relative errors of numerical data stored and processed in computer systems ➢ Compare fixed- and floating-point form		4.5.1 Number systems 4.5.2 Number bases 4.5.3 Units of information 4.5.4 Binary number system 4.5.6 Representing images, sound and other data. ➢

	➢ Bitmapped images are represented in terms of size in pixels, resolution and colour depth. Storage requirements. ➢ Images contain metadata and be able to describe typical metadata ➢ Audio signals can be stored and transmitted in digital form ➢ Sampling rate and resolution and the quality and size of an audio signal ➢ Error detection methods when transmitting signals ➢ MIDI an alternative way of transmitting audio		
Spring 1	**Computer systems** Rationale: Understand the role of hardware and software and how the structure of computer architecture and organisation is affected.		
	➢ Writing and interpreting algorithms using pseudocode ➢ Internal components of a computer system ➢ The role of the processor, main memory, buses and I/O controllers and how these components are connected and how communication is controlled between them ➢ The most appropriate computer architecture for a given application ➢ Concept of addressable memory, the stored program concept ➢ The role and operation of a processor and its major components ➢ The stages of the Fetch-Execute cycle and determine the roles of the various processor registers in facilitating this ➢ Factors that affect the performance of a processor ➢ The role of an instruction set within processors	➢ Application of knowledge understanding and skills using pseudocode and programming ➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class)	**4.1 Fundamentals of programming** **4.2 Fundamentals of data structures** **4.3 Fundamentals of algorithms** 4.3.4 Searching algorithms 4.3.5 Sorting algorithms **4.4 Theory of computation** 4.4.1 Abstraction and automation 4.4.1.2 Following and writing algorithms **4.5 Fundamentals of data representation** 4.5.6 Representing images, sound and other data **4.6 Fundamentals of computer systems** 4.6.1 Hardware and software 4.6.1.4 Role of an OS 4.6.2 Classification of programming languages **4.7 Fundamentals of computer organisation and architecture** 4.7.3 Structure and role of the processor and its components ➢

	➢ The format of processor instructions and the role various components perform ➢ Direct and immediate addressing within processor instructions ➢ How and why image, sound and text data are compressed ➢ Lossy and lossless methods of compression in terms of size and accuracy of data ➢ Caesar and Vernam encryption techniques in their suitability for encrypting messages ➢ Hardware and software and understand the relationship between them ➢ System software and application software ➢ The need for, and attributes of, different types of software ➢ Function of operating systems, utility programs, libraries and translators ➢ The role of an operating system is to create a virtual machine to hide the complexities of operation from the user ➢ The importance of resource management and processor scheduling ➢ Classification of programming languages into low- and high-level languages ➢ Low-level languages: machine-code and assembly language ➢ 'Imperative high-level language' and its relationship to low-level programming		
Spring 2	**Fundamentals of computer organisation and architecture** Rationale: How the structure and role of the processor and its components are within the architecture of the computer.		

	➢ Basic machine code operations expressed in mnemonic-form assembly language ➢ Apply immediate and direct addressing modes ➢ Characteristics and principles of: ➢ Barcode readers ➢ Digital cameras ➢ Laser printers ➢ RFID ➢ The need for secondary storage within a computer system ➢ The main characteristics and principles of operations of: ➢ Hard disk ➢ Optical disk ➢ Solid-State Disk (SSD) ➢ The role of an assembler, compiler and interpreter ➢ Bytecode is produced as the final output by some compilers and how it is subsequently used ➢ Source and object (executable) code ➢ Drawing and interpreting of logic gate circuit diagrams involving multiple gates, half-adder and a full-adder ➢ Edge-triggered D-type flip-flop as a memory unit ➢ Use of Boolean identities and De Morgan's laws to manipulate and simplify Boolean expressions ➢ Boolean expressions for a given logic gate circuit, and vice versa	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class) ➢ Knowledge Quizzes ➢ Application of knowledge understanding ➢ Summative end of year 12 exam paper in the style of the A Level paper	**4.6 Fundamentals of computer systems** 4.6.2 Classification of programming languages 4.6.3 Types of program translator 4.6.4 Logic gates 4.6.5 Boolean algebra **4.7 Fundamentals of computer organisation and architecture** 4.7.3 Structure and role of the processor and its components 4.7.4 External hardware devices 4.7.4.1 Input and output devices ➢
Summer 1	**Networks** Rationale: Understand the fundamentals of communication and networking. Know what network architecture is and why it is required.		
	➢ Serial and parallel transmission methods	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class)	**4.9 Fundamentals of communication and networking** 4.9.1 Communication

	➢ Synchronous and asynchronous data transmission ➢ Baud rate, bit rate, bandwidth, latency, protocol ➢ Basic concepts of OOP ➢ Class ➢ Object ➢ Instantiation ➢ Encapsulation ➢ Inheritance, polymorphism and overriding ➢ Concepts of aggregation ➢ Composition ➢ Association ➢ Object-oriented design principles: ➢ Encapsulate what varies ➢ Favour composition over inheritance ➢ Program to interfaces, not implementation ➢ Draw and interpret class diagrams ➢ Aspects of software development ➢ Prototyping/agile approach that may be used in the analysis, design and implementation of a system ➢ Criteria for evaluating a computer system ➢ Star and bus topologies for a local area network, difference between physical and logical network topologies ➢ Operation of physical star and bus network topologies, advantages and disadvantages of each ➢ Peer-to-peer and client-server networking, situations might be used ➢ Advantages and drawbacks of cloud computing	➢ Application of knowledge understanding	4.9.2 Networking 4.9.2.1 Network topology 4.9.2.2 Types of networking between hosts **4.1 Fundamentals of programming** 4.1.2 Programming Paradigms 4.1.2.2 Procedural-oriented programming 4.1.2.3 Object-oriented programming **4.13 Systematic approach to problem solving** 4.13.1 Aspects of software development ➢
Summer 2	**Databases** Rationale: Understand the fundamentals of databases. The concept of databases and how SQL is used within programming to access information.		

	➢ Purpose of Wi-Fi and the components required for wireless networking. How wireless networks are secured. Wireless protocols CSMA/CA and RTS/CTS ➢ The purpose of SSIDs ➢ Developments in digital technologies enable organisations to monitor behaviour, amass and analyse personal information ➢ The potential for individual computer scientists and software engineers, the challenges facing legislators in the digital age, as well as the responsibilities ➢ The current capacity to distribute, publish, communicate and disseminate personal information. ➢ Software and their algorithms embed moral and cultural values ➢ Entity descriptions representing a data model in the form: Entity1 (Attribute1, Attribute2…). Representing a data model including: attribute, primary key, composite primary key, foreign key ➢ Concept of a relational database, normalise relations to third normal form ➢ SQL to retrieve data from multiple tables of a relational database	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class) ➢ Application of knowledge understanding and skills using SQL	**4.9 Fundamentals of communication and networking** 4.9.1 Communication 4.9.2 Networking 4.9.2.3 Wireless networking **4.8 Consequences of uses of computing** 4.8.1 Individual (moral), social (ethical), legal and cultural issues and opportunities **4.9 Fundamentals of communication and networking** **4.10 Fundamentals of Databases** 4.10.1 Conceptual data models and entity relationship modelling 4.10.2 Relational databases 4.10.3 Database design and normalisation techniques 4.10.4 Structured Query Language (SQL) 4.10.5 Client server databases ➢

Year 13 Overview

Term	Knowledge	Assessment	Connections to learning

Autumn 1	**Programming Project** Rationale: The programming project allows students to develop their practical skills in a problem solving context by coding a solution to a given problem and producing a report documenting the development of the solution.		
	➢ Application of programming skills to given programming project ➢ concept of an abstract data type ➢ concept and uses of a queue ➢ creation and maintenance of data within a queue (linear, circular, priority) ➢ Using a linear, circular and priority queue ➢ Add an item ➢ Remove an item ➢ Test for an empty queue ➢ Test for a full queue ➢ A list may be implemented as a static or dynamic data structure ➢ Items may be added to or deleted from a list ➢ Concept and uses of a stack ➢ Creation and maintenance of data within a stack ➢ Push, pop, peek (or top), test for empty stack, test for full stack ➢ A stack frame is used with subroutine calls to store return addresses, parameters and local variables ➢ A hash table and its uses ➢ Simple hashing algorithms ➢ Collision and how collisions are handled using rehashing ➢ Concept of a dictionary ➢ Simple applications of a dictionary ➢ A graph as a data structure used to represent complex relationships and typical uses ➢ Graph, weighted graph, vertex/node, edge/arc, undirected graph, directed graph	➢ Application of knowledge understanding and skills using pseudocode and programming ➢ This will challenge their knowledge and application of the project	**4.2 Fundamentals of Data Structures** 4.2.1.4 Abstract data types/data structures 4.2.2 Queues 4.2.2.1 Queues 4.2.1.4 Abstract data types/data structures 4.2.1.2 Single- and multi-dimensional arrays (or equivalent) **4.1 Fundamentals of programming** 4.1.1.15 Role of stack frames in subroutine calls 4.2.3.1 Stacks 4.2.6.1 Hash tables 4.2.7.1 Dictionaries ➢ 4.2.4.1 Graphs

	➢ An adjacency matrix and an adjacency list may be used to represent a graph ➢ Compare the use of adjacency matrices and adjacency lists		
	➢ Draw and interpret simple state transition diagrams for FSMs with no output and with output ➢ Draw and interpret simple state transition tables for FSMs with no output and with output ➢ Concept of a set and the notations used for specifying a set and set comprehension ➢ Compact representation of a set ➢ Concept of finite and infinite sets, countably infinite sets, cardinality of a finite set, Cartesian product of sets ➢ The meaning of the terms subset, proper subset, countable set ➢ Set operations: membership, union, intersection, difference ➢ Regular expression is a way of describing a set ➢ Regular expressions allow particular types of languages to be described in a convenient shorthand notation ➢ Form and use simple regular expressions for string manipulation and matching ➢ The relationship between regular expressions and finite state machines ➢ Write a regular expression to recognise the same language as a given FSM and vice versa ➢ The structure and use of Turing machines that perform simple computations	➢ Application of knowledge understanding and skills	**4.4 Theory of Computation** 4.4.2 Regular languages 4.4.2.1 Finite state machines (FSMs) with and without output 4.4.2.2 Maths for regular expressions 4.4.2.3 Regular expressions 4.4.2.4 Regular language 4.4 Theory of Computation 4.4.5 A model of computation 4.4.5.1 Turing machine 4.4.3 Context-free languages 4.4.3.1 Backus-Naur Form (BNF)/syntax diagrams **4.3 Fundamentals of Algorithms** 4.3.2 Tree-traversal 4.3.2.1 Simple tree-traversal algorithms 4.3.3 Reverse Polish ➢ 4.3.3.1 Reverse Polish – infix transformations

	➢ A Turing machine can be viewed as a computer with a single fixed program ➢ Transition rules using a transition function or state transition diagram ➢ Hand-trace a simple Turing machine ➢ The importance of Turing machines and the Universal Turing machine to the subject of computation ➢ Backus-Naur Form (BNF) can be used to represent language syntax and formulate simple production rules ➢ BNF can represent some languages that cannot be represented using Regular Expressions ➢ A syntax diagram to represent an equivalent BNF expression ➢ Convert simple expressions in infix form to Reverse Polish Notation (RPN) and vice versa ➢ Be aware of why and where RPN is used		
Autumn 2	**Structure of the Internet** Rationale: Understand the structure of the Internet and how standards and protocols are put in place to ensure that information is accessible and secure.		
	➢ Application of programming skills to given programming project	➢ Application of knowledge understanding and skills using pseudocode and programming	➢
	➢ A tree is a connected, undirected graph with no cycles ➢ A binary tree is a rooted tree in which each node has at most two children ➢ Typical uses for rooted trees ➢ Concept of a vector and notations for specifying a vector as a list of numbers, as a function or as a geometric point in space ➢ A vector using a list, dictionary or array data structure	➢ Teacher/pupil questioning ➢ Application of knowledge understanding and skills using pseudocode and programming ➢ Exam style question practice (homework's and in class)	**4.2 Fundamentals of Data Structures** 4.2.1.4 Abstract data types/data structures 4.2.5.1 Trees (including binary trees) 4.2.8.1 Vectors **4.1 Fundamentals of programming** 4.1.1.16 Recursive techniques **4.4 Theory of Computation** 4.4.4 Classification of algorithms 4.4.4.1 Comparing algorithms **4.3 Fundamentals of Algorithms**

	➢ Perform operations on vectors: addition, scalar vector multiplication, convex combination, dot or scalar product ➢ The dot product to find the angle between two vectors ➢ The use of recursive techniques in programming languages ➢ Solve simple problems using recursion ➢ Trace recursive tree-traversal algorithms: pre-order, post-order, in-order ➢ The concept of a function as a mapping from one set of values to another ➢ The concept of constant, linear, polynomial, exponential and logarithmic functions ➢ The notion of permutation of a set of objects or values ➢ The Big-O notation to express time complexity ➢ Derive the time complexity of an algorithm ➢ Trace and analyse the time complexity of the linear search and binary search algorithms ➢ Trace and analyse the time complexity of the binary tree search algorithm ➢ Trace and analyse the time complexity of the bubble sort algorithm ➢ Trace and analyse the time complexity of the merge sort algorithm		4.3.2 Tree-traversal 4.3.2.1 Simple tree-traversal algorithms 4.4.4.2 Maths for understanding Big-0 notation 4.4.4.3 Order of complexity 4.4.4.4 Limits of computation 4.4.4.5 Classification of algorithmic problems 4.4.4.6 Computable and non-computable problems 4.3.4 Searching algorithms 4.3.5 Sorting algorithms ➢
	➢ The structure of the Internet ➢ ‘Uniform Resource Locator’ (URL) in the context of networking ➢ ‘Domain name’ and ‘IP address’ ➢ How domain names are organised	➢ Teacher/pupil questioning ➢ Exam style question practice (homework’s and in class) ➢ Application of knowledge understanding and skills	**4.9 Fundamentals of communication and networking** 4.9.3 The Internet 4.9.3.1 The Internet and how it works 4.9.3.2 Internet security

	➢ The purpose and function of the Domain Name Server (DNS) system ➢ The service provided by Internet registries and why they are needed ➢ The role of packet switching and routers ➢ The main components of a packet ➢ Where and why routers and gateways are used ➢ How routing is achieved across the Internet ➢ How a firewall works ➢ Symmetric and asymmetric encryption and key exchange ➢ How digital signatures and certificates are obtained and used ➢ Worms, Trojans and viruses and the vulnerabilities that they exploit ➢ Improved code quality, monitoring and protection can be used against such threats ➢ The roles of the four layers in the TCP/IP protocol stack and sockets ➢ MAC addresses ➢ The common protocols and the well-known ports they use ➢ Transferring files using FTP as an anonymous and non-anonymous user ➢ Secure Shell (SSH) is used for remote management including the use of application level protocols for sending and retrieving email ➢ The role of an email server in sending and retrieving email ➢ The role of a web server in serving up web pages in text form ➢ The role of a web browser in retrieving web pages and web page resources and rendering these accordingly		4.9.4 The Transmission Control Protocol/Internet Protocol (TCP/IP) protocol 4.9.4.1 TCP/IP 4.9.4.2 Standard application layer protocols 4.9.4.3 IP address structure 4.9.4.4 Subnet masking 4.9.4.5 IP standards 4.9.4.6 Public and private IP addresses 4.9.4.7 Dynamic Host Configuration Protocol (DHCP) 4.9.4.8 Network Address Translation (NAT) ➢ 4.9.4.9 Port forwarding

	➢ An IP address is split into a network identifier and a host identifier part ➢ A subnet mask is used to identify the network identifier part of the IP address ➢ There are currently two standards of IP address, (v4 and v6) and why v6 was introduced ➢ Routable and non-routable IP addresses ➢ The purpose and function of the Dynamic Host Configuration Protocol (DHCP) system ➢ Basic concepts of Network Address Translation (NAT) and port forwarding and why they are used		
Spring 1	**Fundamentals of communication and networking** Rationale: The necessity of fundamentals of communication and networking. Whether thick or thin client is best.		
	➢ tTace depth-first and breadth-first algorithms and typical applications of each ➢ Trace Dijkstra's shortest path algorithm ➢ Applications of the shortest path algorithm ➢ Algorithmic complexity and hardware impose limits on what can be computed ➢ Algorithms may be classified as being either tractable or intractable ➢ Some problems cannot be solved algorithmically ➢ The Halting problem, and its significance for computation	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class) ➢ Application of knowledge understanding and skills using pseudocode and programming	**4.3 Fundamentals of algorithms** 4.3.1 Graph-traversal 4.3.1.1 Simple graph-traversal algorithms 4.3.6 Optimisation algorithms 4.3.6.1 Dijkstra's shortest path algorithm **4.4 Theory of Computation** 4.4.4.4 Limits of computation 4.4.4.5 Classification of algorithmic problems 4.4.4.6 Computable and non-computable problems ➢ 4.4.4.7 Halting problem
	➢ The client server model ➢ The WebSocket protocol and know why and where it is used	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class)	**4.9 Fundamentals of communication and networking** 4.9.4 The Transmission Control Protocol/Internet Protocol (TCP/IP)

	➢ The principles of web CRUD applications and Representational State Transfer (REST) ➢ Compare JSON (JavaScript Object Notation) with XML ➢ Compare and contrast thin-client computing with thick-client computing	➢ Application of knowledge understanding and skills using pseudocode and programming	protocol 4.9.4.10 Client server model 4.9.4.11 Thin- versus thick-client computing **4.9 Fundamentals of communication and networking** 4.9.3 The internet 4.9.4 The Transmission Control Protocol/Internet Protocol (TCP/IP) ➢ protocol
Spring 2	**Preparing for the exams** Rationale: You will be using the lessons to look at exam techniques, go through past papers, revising different topics and reinforcing your learning in preparation for your two exams papers. Practice writing algorithms and using to answer questions.		
	➢ Exam skills and misconception ➢ What is meant by a programming paradigm ➢ Function type, domain and co-domain ➢ What is meant by a first-class object and how such an object may be used ➢ Evaluate simple functions ➢ Functional composition to combine two functions ➢ Partial function application ➢ A function takes only one argument which may itself be a function ➢ Higher-order functions, including map, filter and fold ➢ A list is a concatenation of a head and a tail, where the head is an element of a list and the tail is a list ➢ An empty list ➢ Apply list operations: ➢ Return head/tail of list ➢ Test for empty list ➢ Return length of list ➢ Construct an empty list ➢ Prepend / append an item to a list ➢ That Big Data is a term used to describe data whose volume is too	➢ Teacher/pupil questioning ➢ Exam style question practice (homework's and in class) ➢ Application of knowledge understanding and skills using pseudocode and programming ➢ Exampro	**4.12 Fundamentals of functional programming** 4.12.1 Functional programming paradigm 4.12.2 Writing functional programs 4.12.3 Lists in functional programming **4.11 Big Data** ➢

	large to fit on a single server and is generally unstructured ➢ Features of functional programming which make it suitable for analysing Big Data ➢ Fact-based model for representing data ➢ Graph schema for capturing the structure of the dataset		
Summer 1	**Getting Ready for the exam** Rationale: You will be using the lessons to look at exam techniques, go through past papers, revising different topics and reinforcing your learning in preparation for your two exams papers. Practice writing algorithms and using to answer questions.		
	➢ Exam practise & skills	➢ Practise exam papers and questions ➢ Timed responses ➢ Marking activities ➢ Examiner's report ➢ Exampro	
Summer2	**External Exams** External exams begin		