Bishop Stopford's School

Curriculum Map Year 11 **Computer Science**

Curriculum Intent: To inspire every student to engage in lessons and want to explore the curriculum beyond the classroom

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Unit of work	**Paper 1 Computer Systems**	**Paper 2 Computational Thinking, Algorithms and Programming: Revision and Exam Practice**	**Programming**	**Paper 1 Computer Systems: Revision and Exam Practice**	**Paper 2 Computational thinking, algorithms and Programming: Revision and Exam Practice**	**Revision and Exam Practice – Paper 1 & 2. E-Revision – OCR J277**

Year 1

	Lesson 1	Lesson 2	Lesson 3
Autumn Term			
Week 1	2.4.1 Boolean logic	2.4.1 Boolean Logic	2.1 Boolean logic
Week 2	2.4.1 Boolean logic	2.2.3 Units	
Week 3	2.2.4 Data storage - Numbers	2.2.4 Data storage - Numbers	2.4 Data storage - Numbers
Week 4	2.2.4 Data storage - Numbers	2.2.4 Data storage - Numbers	
Week 5	2.2.4 Data storage - Characters	2.2.1.1 Computational thinking	2.2 Computational thinking
Week 6	2.2.4 Data storage - Characters	Practical Programming	
			Half Term
Week 7		2.1.2 Designing, creating and refining algorithms	
Week 8	2.2.4 Data storage - Images	2.1.2 Designing, creating and refining algorithms	2.2 Designing, creating and refining algorithms
Week 9	2.2.4 Data storage - Sound	2.1.2 Designing, creating and refining algorithms	
Week 10	2.2.4 Data storage - Sound	2.1.2 Designing, creating and refining algorithms	
Week 11	2.2.5 Data storage - Compression	2.1.2 Designing, creating and refining algorithms	2.2 Designing, creating and refining algorithms
Week 12	2.2.5 Data storage - Compression	Practical Programming	
			End of Term
Spring Term			
Week 1	2.1.1. Architecture of the CPU	2.2.1 Programming fundamentals	2.1 Programming fundamentals
Week 2	2.1.1. Architecture of the CPU	2.2.2 Data types	
Week 3	2.1.1. Architecture of the CPU	2.2.2 Data types	2.2 Data types
Week 4	2.1.1. Architecture of the CPU	2.2.2 Data types	
Week 5	2.1.1. Architecture of the CPU	2.2.1 Programming fundamentals	2.1 Programming fundamentals
Week 6	2.1.1 CPU Performance	2.1.1 CPU Performance	
			Half Term
Week 7	2.1.3 Embedded systems	2.2.1 Programming fundamentals	2.1 Programming fundamentals
Week 8	2.1.3 Embedded systems	2.2.1 Programming fundamentals	
Week 9	2.2.1 Primary storage (Memory)	2.2.3 Additional programming techniques	2.2 Additional programming techniques
Week 10	2.2.1 Primary storage (Memory)	2.2.3 Additional programming techniques	
Week 11	2.2.2 Secondary storage	2.2.3 Additional programming techniques	2.2 Additional programming techniques
Week 12	2.2.2 Secondary storage	2.2.3 Additional programming techniques	
			End of Term
Summer Term			
Week 1	2.2.2 Secondary storage	2.2.3 Additional programming techniques	2.2 Additional programming techniques
Week 2	2.2.2 Secondary storage	Practical Programming Skills	
Week 3	2.2.2 Secondary storage	Practical Programming Skills	Practical Programming Skills
Week 4	2.5.1 Networks and topologies	Practical Programming Skills	
Week 5	2.5.1 Networks and topologies	Practical Programming Skills	Practical Programming Skills
Week 6	2.5.1 Networks and topologies	Practical Programming Skills	
			Half Term
Week 7	2.5.1 Networks and topologies	2.5.1 Networks and topologies	2.1 Networks and topologies
Week 8	2.5.1 Networks and topologies	2.5.1 Networks and topologies	
Week 9	2.5.2 Wired and wireless networks, protocols and layers	2.5.2 Wired and wireless networks, protocols and layers	2.2 Wired and wireless networks, protocols and layers
Week 10	2.5.2 Wired and wireless networks, protocols and layers	2.5.2 Wired and wireless networks, protocols and layers	
Week 11	2.5.2 Wired and wireless networks, protocols and layers	2.5.2 Wired and wireless networks, protocols and layers	2.2 Wired and wireless networks, protocols and layers
Week 12	Exam Prep	Year 10 Exams	
			End of Term

Year 2

	Lesson 1	Lesson 2	Lesson 3
	1.4.1 Threats to computer systems and networks	2.5.1 Defensive design	1.4.1 Threats to computer systems and networks
	1.4.1 Threats to computer systems and networks	2.5.1 Defensive design	
	1.4.2 Identifying and preventing vulnerabilities	2.5.1 Defensive design	1.4.2 Identifying and preventing vulnerabilities
	1.5.1 Operating systems	2.5.1 Defensive design	
	1.5.1 Operating systems	2.5.1 Defensive design	1.5.1 Operating systems
	2.5.2 Testing	2.5.2 Testing	
	1.5.2 Utility software	2.5.2 Testing	1.5.2 Utility software
	1.5.2 Utility software	2.5.2 Testing	
	1.6.1 Ethical, legal, cultural and environmental impact	2.5.1 Languages	1.6.1 Ethical, legal, cultural and environmental impact
	1.6.1 Ethical, legal, cultural and environmental impact	2.5.1 Languages	
	1.6.1 Ethical, legal, cultural and environmental impact	2.5.2 The Integrated Development Environment (IDE)	1.6.1 Ethical, legal, cultural and environmental impact
	1.6.1 Ethical, legal, cultural and environmental impact	2.5.2 The Integrated Development Environment (IDE)	
	Programming Revision	Programming Revision	Programming Revision
	Programming Revision	2.1.3 Searching and sorting algorithms	
	2.1.3 Searching and sorting algorithms	2.1.3 Searching and sorting algorithms	2.1.3 Searching and sorting algorithms
	Searching and Sorting Practical Programming skills	Searching and Sorting Practical Programming skills	
	Searching and Sorting Practical Programming skills	Searching and Sorting Practical Programming skills	Searching and Sorting Practical Programming skills
	Searching and Sorting Practical Programming skills	Searching and Sorting Practical Programming skills	
	Mocks	Mocks	Mocks
	Theory Revision	Practical Programming Skills Revision	
	Theory Revision	Practical Programming Skills Revision	Theory Revision
	Theory Revision	Practical Programming Skills Revision	
	Theory Revision	Practical Programming Skills Revision	Theory Revision
	Theory Revision	Practical Programming Skills Revision	
	Theory Revision	Practical Programming Skills Revision	Theory Revision
	Theory Revision	Practical Programming Skills Revision	
	Theory Revision	Practical Programming Skills Revision	Theory Revision
	Theory Revision	Practical Programming Skills Revision	
	Theory Revision	Theory Revision	Theory Revision
	Theory Revision		

Core Skills	Computational and Algorithmic Thinking, Logical reasoning and Analytical skills	Problem Solving, Planning and Designing skills,	Evaluative and testing skills	Ability to Implement planned solutions and Synthesize information	Logicl Reasoning and evaluative skills	Revision , Self regulation and metacoitive skills. AO1 - Demonstrate knowledge and understanding of the key concepts and principles of computer science AO2 - Apply knowledge and understanding of key concepts and principles of computer science AO3 - Analyse problems in computational terms: • To make reasoned judgments and to design, program, evaluate and refine solutions curriculum is developed so that students are taught the principles of problem-solving and computation. Develop knowledge, skills and understanding through critical computational concepts and experience.
Core Knowledge	Systems Architecture; Memory, Storage; Wired and wireless networks; Network topologies; Protocols and Layers; System Security; System Software; Ethical, Legal, Cultural and Environmental Concerns	Algorithms; Programming Techniques; Producing Robust Programs; Computational Logic; Translators and Facilities of Languages; Data Representation;	Programming fundamentals, techniques and Data Types.	Systems Architecture, Memory, storage, Networks, Topologies, Protocols and layers and the Impact of IT	Algorithms, Programming Techniques, Producing Robust Programs, Computational Logic Translators & Facilities of Languages and Data Representation.	Computer Theory, Computational Thinking and Problem Solvng.
Assement & Feedback	Class structured assessments, End of unit and topic assessments, Diagnostic Assessments with detailed constructive feedback given to improve and progress. Half Term assessment of skills by the student based on units' delivered. Termly assessment of skills by the teacher (In addition to formal school reporting) in class mini plenaries (every lesson) Homework tasks. Trial exams					
Link to prior learning	The Big ideas: The big ideas of the curriculum content are Digital literacy, Programming, Algorithms and Computational thinking. In Year 10, students studied Algorithms and Data Representation. You were also introduced to programming in SQL, and techniques to develop their skills in programming. In Year 11, the curriculum students will engage with will be hinged on programming techniques and producing robust programs. Students will develop greater insight into analysing, designing, developing, testing and evaluating programs. Students will consider Boolean logic as you work with programming constructs and data structures. They will also learn about systems security and focus hugely on how to use theoretical knowledge and programming skills to answer exam styled questions. Beyond Year 11, students will develop their knowledge of Computer Science to an advanced level and gain mastery in programming concepts and problem solving, developing and using models to create solutions to problems. Important Skills students will acquire and develop which will be crucial to your study are: mathematical, investigative, analytical, reasoning, problem solving skills and critical thinking.					
Outside learning/Trips	Visits to top London Computer Science Universities (Forensics, programming and Artificial Intelligence). University of Oxford, University of Cambridge, Imperial College London, University College London (UCL). King's College London, London School of Economics and Political Science (LSE)					