Subject :	Computer Science	Year Group:	10

	June - July	September to October	October - December	December - February	February - April	April - June
Scheme title	**Fundamentals of Algorithms**	**Fundamentals of Programming**	**Fundamentals of Programming**	**Fundamentals of Data Representation**	**Fundamentals of Computer Systems**	**Non Exam Assessment**
Purpose of Scheme	**A high-quality Computer Science education ensures all students: •are prepared for the future giving them opportunities to gain knowledge and develop skills for the ever changing digital world.**	**A high-quality Computer Science education ensures all students: •are prepared for the future giving them opportunities to gain knowledge and develop skills for the ever changing digital world.**	**A high-quality Computer Science education ensures all students: •are prepared for the future giving them opportunities to gain knowledge and develop skills for the ever changing digital world.**	**A high-quality Computer Science education ensures all students: •are prepared for the future giving them opportunities to gain knowledge and develop skills for the ever changing digital world.**	**A high-quality Computer Science education ensures all students: •are prepared for the future giving them opportunities to gain knowledge and develop skills for the ever changing digital world.**	**A high-quality Computer Science education ensures all students: •are prepared for the future giving them opportunities to gain knowledge and develop skills for the ever changing digital world.**
Knowledge in sequence	Fundamentals of Algorithms 3.1.1 Representing algorithms 3.1.2 Efficiency of algorithms 3.1.3 Searching algorithms 3.1.4 Sorting algorithms	Fundamentals of Programming 3.2.1 Data types 3.2.2 Programming concepts 3.2.3 Arithmetic operations in a programming language 3.2.4/5 Relational/Boolean operations in a programming language 3.2.6 Data structures 3.2.7 Input/output and file handling 3.2.8 String handling operations in a programming language 3.2.9 Random number generation in a programming language	Fundamentals of Programming 3.2.8 String handling operations in a programming language 3.2.9 Random number generation in a programming language 3.2.10 Subroutines (procedures and functions) 3.2.11 Structured programming 3.2.12 Robust and secure programming 3.2.13 Classification of pro-gramming languages	Fundamentals of Data Representation 3.3.1 Number bases 3.3.2 Converting between number bases 3.3.3 Units of information 3.3.4 Binary arithmetic 3.3.5 Character encoding 3.3.6 Representing images 3.3.7 Representing sound 3.3.8 Data compression	Fundamentals of Computer Systems 3.4.1 Hardware and Software 3.4.2 Boolean logic 3.4.3 Software classification 3.4.4 Systems architecture	Non Exam Assessment 3.9.1.1 Purpose of non-exam assessment Non-exam assessment (NEA) al-lows students to develop their practical skills in a problem-solving context by coding a solu-tion to a given problem. Non-exam assessment is as much a learning experience as it is a method of assessment: allowing students to work independently, over a period of time, extending their programming skills and increasing their understanding of practical, real world applica-tions of computer science.
Skills	Algorithmic Thinking is thinking like a computer in a sequence of instructions or a set of rules to get something done. Decomposition is the process of breaking down a task into smaller more manageable parts. Abstraction is about simplifying things identifying what's im-portant without worrying too much about detail	Algorithmic Thinking is thinking like a computer in a sequence of instructions or a set of rules to get something done. Decomposition is the process of breaking down a task into smaller more manageable parts. Abstraction is about simplifying things identifying what's im-portant without worrying too much about detail. Programming is the process of designing and writing a set of instructions for a computer in a language it can understand	Algorithmic Thinking is thinking like a computer in a sequence of instructions or a set of rules to get something done. Decomposition is the process of breaking down a task into smaller more manageable parts. Abstraction is about simplifying things identifying what's im-portant without worrying too much about detail. Programming is the process of designing and writing a set of instructions for a computer in a language it can understand	Decomposition is the process of breaking down a task into smaller more manageable parts. Abstraction is about simplifying things identifying what's im-portant without worrying too much about detail	Algorithmic Thinking is thinking like a computer in a sequence of instructions or a set of rules to get something done. Logical reasoning helps us ex-plain why something happens. Decomposition is the process of breaking down a task into smaller more manageable parts. Abstraction is about simplifying things identifying what's im-portant without worrying too much about detail.	Algorithmic Thinking is thinking like a computer in a sequence of instructions or a set of rules to get something done. Logical reasoning helps us ex-plain why something happens. Decomposition is the process of breaking down a task into smaller more manageable parts. Abstraction is about simplifying things identifying what's im-portant without worrying too much about detail. Evaluation is about making judgements, where possible is an objective and systemic way.
Key Words	Flowcharts, Pseudocode, Linear Search, Binary Search, Bubble Sort, Merge Sort, Comparison and effectiveness	KEYTERMS *Data Types *Basic Programming *Advanced Programming *Data Structures *File Handling *Subroutines	*Data Types *Basic Programming *Advanced Programming *Data Structures *File Handling *Subroutines *Classification of Languages	Number Bases, Conversion between bases, Unit of information, Binary adding, Binary shifting, Character sets, Images, Sound, Compression	Hardware, CPU architecture, RAM ROM, Virtual Memory, Secondary storage, Online storage, Software, Embedded systems, Boolean logic	Data Types, Basic Programming, Advanced Programming, Data Structures, File Handling, Subroutines
End Point	Students are able to write their own step by step algorithms for a given problem. They can write in pseudocode and create flowcharts. They will also be able to answer GCSE exam style questions.	Students are able to solve a variety of computational problems and can successfully debug their code. They will also be able to answer GCSE exam style questions.	Students are able to solve a variety of computational problems and can successfully debug their code. They will also be able to answer GCSE exam style questions.	Students are able to convert 8-bit binary numbers into denary and vice versa. Students understand data can be represented and manipulated digitally in the form of binary digits. They will also be able to answer GCSE exam style questions.	Students developed a good understanding of hardware components and are able to understand simple noolean logic for example AND OR and NOT. They will also be able to answer GCSE exam style questions.	NEA Completion
Assessment method	Final Written Assessment: *Algorithms exam 60 marks *Mid unit assessment Reflection grids x 2	Final Written Assessment: *Programming exam 60 marks *Mid unit assessment Reflection grids x 2	*Final Written Assessment: *Data Representation exam 60 marks *Mid unit assessment *Reflection grids x 2	Final Written Assessment: *Data Representation exam 60 marks *Mid unit assessment *Reflection grids x 2	Final Written Assessment: *Computer Systems exam 60 marks *Mid unit assessment *Reflection grids x 2	*20 hour NEA Sample sent to AQA in May every year