Subject: *Computer Science: Paper 2 – Algorithms and programming* **Year group:** *13*

Time period	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer
Content *Declarative Knowledge* – *'Know What'*		*1.2.4 Types of programming languages* (a) programming paradigms. (b) Procedural languages. (e) Object-oriented languages with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism. *2.1.5 Thinking concurrently* (a) What parts of a problem could be solved concurrently. (b) The benefits and trade-offs that might result from concurrent processing in a particular situation.	*2.3.1 Algorithms* Recap of (a) Analysis and design of algorithms .(b) The suitability of different algorithms (c) Big O notation (d) Comparison of the complexity of algorithms. (e) Algorithms for the main data structures, (f) Standard algorithms – recap of searching and sorting. - Introduction to shortest path algorithms – Djikstra and A* algorithm *1.3.2 Databases* (a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing.	*1.3.2 Databases continued* (b) capturing, selecting, managing and exchanging data. (c) Normalisation to 3NF. (d) SQL (e) Referential integrity. (f) Transaction processing, ACID record locking and redundancy. *1.5.2 Moral and ethical issues* The individual moral, social, ethical and cultural opportunities and risks of digital technology.	*Revision - See week by week "run in" schedule*
	Component 3 *Non Exam Assessment – programming project* - Analysis -Design -Iteration(s) Testing Evaluation				

Skills *Procedural Knowledge* *–* *'Know How'*		*1.2.4 Types of programming languages* (a) Understand the need for and characteristics of a variety of programming paradigms? Procedural, Logical, Functional, Object oriented and event driven. (b) Understand the drawbacks of using procedural languages to write complex and larger solutions to problems. (e) understand the benefits of using OOP to solve problems as opposed to using modular procedural code. Understand the structure of the class and how to create a class using a programming language. Learn about creating objects based on the class and the use of the constructor. Know and apply the concept of encapsulation to OOP code. Know and apply the concepts of inheritance and polymorphism to OOP code. 2.1.5 Concurrent processing (a) Determine the parts of a problem that can be tackled at the same time. (b) Be able to outline, discuss and evaluate the benefits and trade off's that may result from concurrent processing in a given situation.	*2.3.1 - Algorithms* a) (b) (c) (d) (e) Recap on how to compare the suitability and complexity of algorithms in terms of worst case big O notation time and space complexity. Students will recap on the need to understand the Big O notation for the searching and sorting algorithms for larger and smaller data sets. (f) investigate and analyse how Dijkstra's shortest path algorithm, A* algorithm is carried out and how they best determine the shortest path algorithm solution. *1.3.2 Databases* (a) Understand the need for and creation of a Relational database. Understand, apply and create primary keys, foreign keys, secondary keys, Use entity relationship modelling to create efficient relational DBMS or RDBMS using normalisation and indexing.	b) Methods of capturing, selecting, managing and exchanging data. (c) Using Normalisation to create efficient relational DBMS up to and including 3NF. (d) SQL – Interpret and modify SQL commands to manipulate the data in a RDBMS. (e) Understand the need for Referential integrity in data in a database and how ensuring this avoids data redundancy and duplication. (f) Understand transaction processing, ACID, record locking and redundancy in a RDBMS (Relational Database Management System). *1.5.2 Moral and ethical issues* Discuss and evaluate: • 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.	
	Component 3 *Non Exam Assessment – programming project* Students will be expected to (3.1) analyse, (3.2) design, (3.3) develop, (3.4) test, (3.5) They will document a program written in a suitable programming language. The underlying approach to the project is to apply the principles of computational thinking to a practical coding problem. Students are expected to apply appropriate principles from an agile development approach to the project development.				

Key Questions	What is computational thinking? How do we apply decomposition and abstraction to a given problem? What are tractable and intractable problems? What is data mining? What is big data? Where do I apply these techniques? What are heuristics? How do I apply these to computational problems? How is heuristics helpful in problem solving? How does performance modelling and visualisation improve the quality of the data representation? Where do I ply the rules of backtracking? How is pipelining useful in solving problems computationally?	What is a programming paradigm? Where might we use a particular paradigm? What are the drawbacks of using procedural over OOP? What is the OOP paradigm? What is a class? How do we create objects from the class? Why does the IDE need a constructor? How do I create a class in program code? What is encapsulation? What are accessors and mutators and why do I need to use them to encapsulate the data? What are getters and setters' accessors and mutators? What is inheritance? What is a parent and child class? What is a base and derived class. How do I inherit from a parent class? What is polymorphism? What are the benefits in solving problems of being able to create methods using polymorphism. Why is programming using Object orientation so efficient? What is concurrent programming? Where could I apply concurrent thinking to a given problem? What are the benefits and trades off's of using concurrent programming?	What are the different searching, sorting and shortest path algorithms? What is divide and conquer? When is it suitable to use each algorithm for a given problem? What is the time complexity of the different searching and sorting algorithms? Why is time complexity important in relation to the data set given for each of the algorithms? What are the different data structures used to hold data and programs in memory? How is each of the data structures designed and implements as algorithms? What is the time and space complexity of the data structure. What is a shortest path algorithm? Where is this type of algorithm used in real world applications? What is Djikstra's algorithm? How do I apply this to find the shortest path? What are heuristics? Why are heuristics used in the A* algorithm? Does using heuristics allow for a more efficient shortest path result?	What is a flat file database? What is a Relation database management system (RDBMS). Why normalise? What is the benefit? How do I know I have completed 1NF? What is 2NF? How do I know my database is in 2NF? Why use 3NF? Is my database know a RDBMS? It is efficient? What is SQL? How do I use SQL to interpret and modify the database? What is referential integrity? Why do I need to ensure referential integrity? How does it make my database less prone to redundancy or duplication of data? What are tables? What are records? What are fields? What is an ER diagram? How do I use an ER diagram to model the structure of the entities in my database? What is transactional processing? What is ACID (Atomicity, Consistency, Isolation, Durability)? How do I apply this to the data in the database? What are the ethical and moral issues with using technology applied to the given scenarios?	
Component 2	What is a problem definition? What or who is a stakeholder? What do we mean by the complexity of our project? How do I describe the essential features of a computational? solution explaining these choices. Explain the limitations of the proposed solution. Justify the solution requirements? What is success criteria? What is the iterative development process? What is iterative and final testing? What is algorithmic design? How does my success criteria inform my final product?				
#Assessment	End of unit tests, Programming activities using the OOP paradigm Past exam questions to consolidate learning Exam style HBL questions Trail exams Programming project				
Literacy/Numeracy/ SMSC/Character	Programming language literacy Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills	Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills Mathematical computation Data handling Linear Algebra Discrete mathematics	Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills Mathematical computation Data handling Linear Algebra Discrete mathematics Graph theory	Programming language literacy Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills Mathematical computation Data handling Linear Algebra Discrete mathematics	