

What are the aims and intentions of this curriculum?

The aims of the year 13 curriculum map are to enable learners to develop:

- An understanding and ability to apply the fundamental principles and concepts of computer science, including: abstraction, decomposition, logic, algorithms and data representation
- The ability to analyse problems in computational terms through practical experience of solving such problems, including writing programs to do so.
- The capacity to think creatively, innovatively, analytically, logically and critically
- The capacity to see relationships between different aspects of computer science
- Mathematical skills.

Term	Topics	Knowledge and key terms	Skills developed	Assessment
Autumn 1	<p>2.2 Problem solving and programming</p> <p>2.2.2 Computational methods</p> <p>1.2 Software and software development</p> <p>1.2.4 Types of Programming Language</p>	<p>(a) Features that make a problem solvable by computational methods.</p> <p>(b) Problem recognition. (c) Problem decomposition.</p> <p>(d) Use of divide and conquer.</p> <p>(e) Use of abstraction.</p> <p>(f) Learners should apply their knowledge of:</p> <ul style="list-style-type: none"> ✓ Backtracking ✓ data mining ✓ heuristics ✓ performance modelling ✓ pipelining ✓ visualisation to solve problems. <p>(a) Need for and characteristics of a variety of programming paradigms.</p> <p>(b) Procedural languages.</p> <p>(c) Assembly language (including following and writing simple programs with the Little Man Computer instruction set). See appendix 5d.</p> <p>(d) Modes of addressing memory (immediate, direct, indirect and indexed).</p>	<ul style="list-style-type: none"> • Know how quicksort use the so-called divide and conquer strategy • Define backtracking, data mining, and heuristics. • Identify a list of real-life (non-computing) applications of backtracking. • Know which computational problems can assist through visualization. • Plotting responses for visual correlation. <p>undertake and use a range of assessments to confirm medium and longer term goals and update career and learning targets.</p> <ul style="list-style-type: none"> • Give a comparison of various programming languages after actual use. • Use, understand and know how the following statement types can be combined in programs: <ul style="list-style-type: none"> • variable declaration • constant declaration • assignment 	<ul style="list-style-type: none"> • Individual presentations • Group Presentations • Case Studies • Reviews • End of topic quiz • End of term test • Microsoft Teams collaborative activities. • Home work • Class Discussions • Topic Worksheets • Past Paper question sheets • Research • Quizlet

2.2 Problem solving and programming cont'd

2.2.1 Programming techniques

1.1 The characteristics of contemporary processors, input, output and storage devices

1.1.1 Structure and function of the processor

(e) Object-oriented languages (see appendix 5d for pseudocode style) with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism.

How computers can be used to solve problems and programs can be written to solve them (Learners will benefit from being able to program in a procedure/imperative language and object oriented language.)

- (a)** Programming constructs: sequence, iteration, branching.
- (b)** Recursion, how it can be used and compares to an iterative approach.
- (c)** Global and local variables.
- (d)** Modularity, functions and procedures, parameter passing by value and reference. **(e)** Use of an IDE to develop/debug a program.
- (f)** Use of object oriented techniques.

- (a)** The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: how this relates to assembly language programs.
- (b)** The fetch-decode-execute cycle, including its effect on registers.
- (c)** The factors affecting the performance of the CPU; clock speed, number of cores, cache.

- string handling
- file handling
- subroutine (procedure/function)
- Identify and use mnemonics from LMC
- Write simple programs using Little Man Computing

seek redress if their consent has not been respected; how to recognise and seek help in the case of sexual exploitation, assault or rape.

- Be able to express the solution to a simple problem as an algorithm using pseudo-code, with the standard constructs:
 - ✓ sequence
 - ✓ branching
 - ✓ iteration
- Create flowcharts to identify and represent the various elements of a system.
- Know that branching effectively replaces iteration.
- Develop iterative solutions.
- Develop recursive solutions.

recognise mental health issues in others; be able to offer or find support for those experiencing difficulties with their mental health.

- Know how parts of the CPU interact and function to process instructions and 'compute'.
- Evaluate the development of computer technology and the effects it has had.
- Understand and explain the Fetch-Execute cycle.

- Individual presentations
- Group Presentations
- Case Studies
- Reviews
- End of topic quiz
- End of term test
- Microsoft Teams collaborative activities.
- Home work
- Class Discussions
- Topic Worksheets
- Past Paper question sheets
- Demonstrations

1.1.2 Types of processor

(d) The use of pipelining in a processor to improve efficiency.

(d) Von Neumann, Harvard and contemporary processor architecture.

(a) The differences between and uses of CISC and RISC processors.

(b) GPUs and their uses (including those not related to graphics).

(c) Multicore and Parallel systems.

1.1.3 Input, output and storage

a) How different input, output and storage devices can be applied to the solution of different problems.

(b) The uses of magnetic, flash and optical storage devices.

(c) RAM and ROM.

(d) Virtual storage.

1.2 Software and software development cont'd

1.2.1 Systems Software

Types of software and the different methodologies used to develop software

(a) The need for, function and purpose of operating systems.

(b) Memory Management (paging, segmentation and virtual memory).

(c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle.

(d) Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time.

- Compare several PROCESSOR specifications and professional reviews of the performance of the CPU.
- Discuss the use of pipelining in a processor.
- Compare the Von Neumann and Harvard CPU architecture.

- Compare and contrast a range of CPU benchmarks or look for a specific model.
- Analyse the suitability of the GPU for a range of tasks other than playing video games.
- Tell the difference between serial and parallel processing of instructions.

- Revise key terms and learn definitions of input and output devices.
- Discuss optical and magnetic storage properties
- Justify where each type of storage is used and its suitability.
- Discuss the concept of: a slow computer requires more RAM, so why not just download some more? Learners can discuss the issues or reality of this is RAM downloadable? Would it work?
- Compare the advantages and disadvantages of virtual storage

obtain careers information relevant to needs and process it effectively.

- Understanding the need for and function of operating systems.
- Know that the OS handles interrupts, scheduling, resource management, managing hardware to allocate processors, memories and I/O devices among competing processes.

- Individual presentations
- Group Presentations
- Case Studies
- End of topic quiz
- End of term test
- Microsoft Teams collaborative activities.
- Home work
- Class Discussions
- Topic Worksheets
- Past Paper question sheets

- (e) Distributed, embedded, multi-tasking, multi-user and real time operating systems.
- (f) BIOS.
- (g) Device drivers.
- (h) Virtual machines, any instance where software is used to take on the function of a machine, including executing intermediate code or running an operating system within another.

1.3 Exchanging data

How data is exchanged between different systems

1.3.1 Compression, Encryption and Hashing

- (a) Lossy vs Lossless compression.
- (b) Run length encoding and dictionary coding for lossless compression.
- (c) Symmetric and asymmetric encryption.
- (d) Different uses of hashing.

1.4.1 Data Types

- (a) Primitive data types, integer, real/floating point, character, string and Boolean.
- (b) Represent positive integers in binary.
- (c) Use of sign and magnitude and two's complement to represent negative numbers in binary.
- (d) Addition and subtraction of binary integers.
- (e) Represent positive integers in hexadecimal.
- (f) Convert positive integers between binary hexadecimal and denary.
- (g) Representation and normalisation of floating point numbers in binary.
- (h) Floating point arithmetic, positive and negative numbers, addition and subtraction.
- (i) Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR.
- (j) How character sets (ASCII and UNICODE) are used to represent text.

1.4.2 Data Structures

- a) Arrays (of up to 3 dimensions), records, lists, tuples.

- Understand the term 'embedded system' and explain how an embedded system differs from a Distributed system.
- Know the instance where software is used to take on the function of a machine including executing intermediate code or running an operating system within another.
- Discuss why certain operating systems are used.
- Justify reasons for using virtual machines.
- Demonstrate the difference between lossy and lossless audio compression.
- Investigate the difference between the HD and SD quality on YouTube.
- Use hashing principles in python.
- Understand the concept of a data type.
- Create their own examples of the listed data types.
- Know how to:
 - represent negative and positive integers in two's complement
 - perform subtraction using two's complement
- Be able to convert between unsigned binary and decimal and vice versa.
- Be able to add and subtract binary as well as to convert between decimal, binary and hexadecimal number bases.
- Manipulate binary by using bitwise and shifting.
- Describe ASCII and Unicode coding systems for coding character data and explain why Unicode was introduced.
- Use arrays in the design of solutions to simple problems.
- Use stacks, queues, tree and hash table to structure data.

1.3.2 Networks

(a) Characteristics of networks and the importance of protocols and standards.

(b) Internet structure:

- ✓ The TCP/IP stack.
- ✓ DNS
- ✓ Protocol layering.
- ✓ LANs and WANs.
- ✓ Packet and circuit switching.

(c) Network security and threats, use of firewalls, proxies and encryption.

(d) Network hardware.

(c) Client-server and peer to peer.

1.3.4 Web Technologies

(a) HTML, CSS and JavaScript. See appendix 5d.

(b) Search engine indexing.

(c) PageRank algorithm.

(d) Server and client side processing.

1.4.3 Boolean Algebra

(a) Define problems using Boolean logic. See appendix 5d.

(b) Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions.

(c) Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation.

(d) Use logic gate diagrams and truth tables. See appendix 5d.

(e) The logic associated with D type flip flops, half and full adders.

2.1 Elements of computational thinking

2.1.1 Thinking abstractly

Understand what is meant by computational thinking

(a) The nature of abstraction.

- Appreciate the importance of protocols and standards.
- Describe the 4 layer TCP/IP model:
 - ✓ application layer
 - ✓ transport layer
 - ✓ internet layer
 - ✓ link layer.
- Evaluate different network security measures.
- Identify different network hardware and explain their purpose.
- Explain the following and describe situations where they might be used:
 - ✓ peer-to-peer networking
 - ✓ client-server networking.

- Be able to build webpages with the implementation of CSS and JavaScript.
- Demonstrate how search engines work.
- Know how pages are ranked.

- Write a Boolean expression for a given logic gate circuit.
- Use Karnaugh maps appropriately.
- Complete a truth table for a given logic gate circuit.
- Construct truth tables for the following logic gates:
 - NOT
 - AND
 - OR
- Know the logic associated with D type flip flops, half and full adders.

- Be aware that before a problem can be solved, it must be defined, the requirements of the system that solves the problem must be established

- Individual presentations
- Group Presentations
- Case Studies
- Reviews
- End of topic quiz
- End of term test
- Microsoft Teams collaborative activities.
- Home work
- Class Discussions
- Topic Worksheets
- Past Paper question sheets
- Dramatization

	<p>2.1.2 Thinking ahead</p> <p>2.1.3 Thinking procedurally</p> <p>2.1.4 Thinking logically</p> <p>2.1.5 Thinking concurrently</p>	<p>(b) The need for abstraction. (c) The differences between an abstraction and reality. (d) Devise an abstract model for a variety of situations.</p> <p>(a) Identify the inputs and outputs for a given situation. (b) Determine the preconditions for devising a solution to a problem. (c) The nature, benefits and drawbacks of caching. (d) The need for reusable program components.</p> <p>(a) Identify the components of a problem. (b) Identify the components of a solution to a problem. (c) Determine the order of the steps needed to solve a problem. (d) Identify sub-procedures necessary to solve a problem.</p> <p>(a) Identify the points in a solution where a decision has to be taken. (b) Determine the logical conditions that affect the outcome of a decision. (c) Determine how decisions affect flow through a program.</p> <p>(a) Determine the parts of a problem that can be tackled at the same time. (b) Outline the benefits and trade offs that might result from concurrent processing in a particular situation.</p>	<ul style="list-style-type: none"> • The capacity to think creatively, innovatively, analytically, logically and critically • Know the impact of caching in relation to a programming solution or IT system. • Practical skills in the context of solving a realistic problem <p>H19. manage personal safety off-line, including when socialising (including meeting someone in person for the first time whom they met online, drink spiking, looking out for friends) and travelling (especially cycle safety, young driver safety, and passenger safety, including the risks of being a passenger with an intoxicated driver, and using only licenced taxis).</p> <ul style="list-style-type: none"> • Know how to get from a problem to a solution for computational problems. • Outline the benefits and trade offs that might result from concurrent processing in a particular situation. <p>Justify and evaluate the range of opportunities they are considering including HE, training, employment or starting their own business.</p>	
Spring 1	<p>1.5 Legal, moral, ethical and cultural issues</p> <p>1.5.1 Computing related legislation</p>	<p>The individual moral, social, ethical and cultural opportunities and risks of digital technology. Legislation surrounding the use of computers and ethical issues that can or may in the future arise from the use of computers.</p> <p>(a) The Data Protection Act 1998. (b) The Computer Misuse Act 1990. (c) The Copyright Design and Patents Act 1988. (d) The Regulation of Investigatory Powers Act 2000.</p>	<p>R10. appreciate the ways different cultures and faiths view relationships, respecting others' right to hold their own views.</p> <p>R21. recognise forced marriage and 'honour' based violence; get help for themselves or others they believe to be at immediate or future risk.</p> <ul style="list-style-type: none"> • An understanding of the consequences of using computers unlawfully. 	<ul style="list-style-type: none"> • Group Presentations • Case Studies • End of topic quiz • End of term test • Microsoft Teams collaborative activities. • Home work • Class Discussions

1.5.2 Ethical, moral and cultural issues

(a) The individual moral, social, ethical and cultural opportunities and risks of digital technology:

- ✓ 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.

- Understand the professional, ethical, legal, security and social issues and responsibilities
- Understand that:
 - ✓ developments in computer science and the digital technologies have dramatically altered the shape of communications and information flows in societies, enabling massive transformations in the capacity to:
 - monitor behaviour
 - amass and analyse personal information
 - distribute, publish, communicate and disseminate personal information.

- Topic Worksheets
- Past Paper question sheets

2.3 Algorithms

2.3.1 Algorithms

The use of algorithms to describe problems and standard algorithms

(a) Analysis and design of algorithms for a given situation.

(b) The suitability of different algorithms for a given task and data set, **in terms of execution time and space.**

(c) **Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity).**

(d) **Comparison of the complexity of algorithms.**

(e) **Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).**

(f) **Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search).**

- Be able to convert an algorithm from pseudo-code into high level language program code.
- Be able to develop solutions to simple logic problems.
- Know when and how to use different algorithm sorting and searching methods.
- Comparing suitability of different algorithms for a given task and data set.
- Know and use different measures and methods to determine the efficiency of different algorithms
- Explain and use Dijkstra's shortest path algorithm.

- Group Presentations
- Case Studies
- End of topic quiz
- End of term test
- Microsoft Teams collaborative activities.
- Home work
- Class Discussions
- Topic Worksheets
- Past Paper question sheets
- Programming project

**Non exam assessment
PROGRAMMING
PROJECT
(begins Autumn 1)**

R2. accept and use positive encouragement and constructive feedback.

3.1. Analysis of the problem (10 marks)

3.1.1 Problem identification

(a) Describe and justify the features that make the problem solvable by computational methods.
(b) Explain why the problem is amenable to a computational approach.

3.1.2 Stakeholders

(a) Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs (this may be named individuals, groups or persona that describes the target end user).

3.1.3 Research the problem

(a) Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution. **(b)** Describe the essential features of a computational solution explaining these choices. **(c)** Explain the limitations of the proposed solution.

3.1.4 Specify the proposed solution

(a) Specify and justify the solution requirements including hardware and software configuration (if appropriate). **(b)** Identify and justify measurable success criteria for the proposed solution.

3.2 Design of the solution (15 marks)

3.2.1 Decompose the problem

(a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made.

3.2.2 Describe the solution

(a) Explain and justify the structure of the solution. **(b)** Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem. **(c)** Describe usability features to be included in the solution. **(d)** Identify key variables / data structures / classes justifying choices and any necessary validation.

3.2.3 Describe the approach to testing

(a) Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.

- Choose a problem from the list given by OCR and analyse this problem to determine; the stakeholders, the need for a solution and the necessary requirements including hardware software.

consider the possible implications of changes in learning and work for their own career goals and plans including financial options.

- Use appropriate software to design the user interface for the chosen project.
- Students should be able to design and apply test data, normal, boundary and erroneous to the testing of programs so that they are familiar with these test data types and the purpose of testing.

- Seminars
- Programming project
- Past programming project analysis and comparison
- Individual presentation
- Case Studies
- Programming Project
- Class Discussions
- Research
- Past project sample review

Spring 2	<p>3.4 Evaluation (20 marks)</p> <p>3.4.1 Testing to inform evaluation</p> <p>3.4.2 Success of the solution</p> <p>3.4.3 Describe the final product</p> <p>3.4.4 Maintenance and development</p>	<p>(a) Provide annotated evidence of testing the solution of robustness at the end of the development process. (b) Provide annotated evidence of usability testing (user feedback).</p> <p>(a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis.</p> <p>(a) Provide annotated evidence of the usability features from the design, commenting on their effectiveness.</p> <p>(a) Discuss the maintainability of the solution. (b) Discuss potential further development of the solution.</p>	<ul style="list-style-type: none"> • Evaluate their project’s solution to determine its success. • Critically discuss the maintainability of the solution and further development. 	<ul style="list-style-type: none"> • Individual presentation • Case Studies • Project • Class Discussions • Research • Past project samples
Summer 1	Revision External Exams			<ul style="list-style-type: none"> • Past paper questions • Discussions • One-to-one tutoring