

OxfordAQA

International A-level

Computer Science (9645)

Scheme of work

For teaching from September 2024 onwards

For International A-level exams in May/June 2026 onwards

This scheme of work suggests possible teaching and learning activities for each section of the specification that covers A2 level content. There are far more activities suggested than it would be possible to teach. It is intended that teachers should select activities appropriate to their students and the curriculum time available. The first two sections summarise the specification references, whilst the learning outcomes indicate what most students should be able to achieve after the work is completed. The resources section indicates resources commonly available to schools, and other references that may be helpful. The timings are only suggested, as are the possible teaching and learning activities, which include references to experimental work. Resources are only given in brief and risk assessments should be carried out.

Contents

You can use the title links to jump directly to the different sections of this scheme of work
(Use Ctrl and click to follow the link)

Section title	Page
Introduction	4
Unit 3: Advanced programming	5
3.9 Object-oriented and additional programming	7
3.10 Advanced data structures	14
3.11 Advanced algorithms	14
Unit 4: Advanced concepts and principles of computer science	24
3.12 Functional programming	24
3.13 Theory of computation	26
3.14 Networking and cyber security	35
3.15 Databases	50
3.16 Artificial intelligence	60

Introduction

This outline scheme of work is intended to help teachers plan and implement the teaching of the Oxford AQA International A-level Computer Science (9645) specification. The purpose of this scheme of work is to provide advice and guidance to teachers, not to prescribe and restrict their approach to the specification. This scheme has been produced by a senior Computer Science examiner and is intended to be helpful for teachers who are new to the specification. There are many other ways of organising the work, and there is absolutely no requirement to use this scheme. The scheme of work should be used alongside the specimen assessment material.

Assumed coverage

This scheme assumes that the A2 Computer Science content is a one-year course and is delivered after the AS Level Computer Science content has been taught. A separate scheme of work is available that covers the AS Level Computer Science content.

In this scheme of work, we have made the assumption that it will be taught over about 30 weeks with 4½ to 5 hours of contact time per week. Teachers will need to fine tune the timings to suite their own students and the time available. It could also be taught by one teacher or by more than one teacher with topics being taught concurrently.

International terminology

International terminology will be used, for more guidance please see subject specific vocabulary.

Unit 3: Advanced programming

Programming languages

A large section of this scheme of work covers practical programming activities. Centres must choose **one** of these programming languages:

- C#
- Python
- VB.Net

These links contain general resources that can be used to support each programming language. Teachers and students may find these links useful in their delivery/study. These resources are by no means exhaustive and centres are encouraged to use other resources.

Python:

Python official website:

docs.python.org/3/tutorial/index.html

Python official website:

docs.python.org/2/tutorial/index.html

Python tutor videos with schematic animations:

youtube.com/watch?v=joGNzHR4GhI

Dive into Python:

diveintopython3.net/

Dive into Python:

diveintopython.net/

Python tutor:

pythontutor.com/

Wikibooks:

en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial

C#:

Home and Learn:

homeandlearn.co.uk/csharp/csharp.html

MSDN:

learn.microsoft.com/en-gb/

Wikibooks:

en.wikibooks.org/wiki/C_Sharp_Programming

VB.Net:

Tutorialspoint:

[tutorialspoint.com/vb.net/](https://www.tutorialspoint.com/vb.net/)

Wikibooks:

[en.wikibooks.org/wiki/A-level Computing/AQA/Problem Solving, Programming, Data Representation and Practical Exercise/Fundamentals of Programming/A program](https://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Programming/A_program)

3.9 Object-oriented and additional programming

3.9.1 Classes, objects and instantiation

3.9.2 Encapsulation

Specification content

- Know why the object-oriented paradigm is used.
- Be familiar with the concepts of: class, property/attribute, method, object, instantiation, encapsulation, inheritance, overriding, association and be able to apply and identify these.
- Know what a class is and be able to create classes.
- Know what an object is and how to create objects using instantiation.
- Know that a constructor is called when an object is instantiated from a class and be able to create constructor functions. Know that a constructor initialises an object to a given state.
- Know what encapsulation is and how it is used in object-oriented programming. Be able to design classes to use encapsulation appropriately.
- Be able to use access modifiers to determine which classes can access properties and methods in a class.
- Use getter and setter methods to provide controlled access to data stored in a class.

Learning outcomes

- Know how an object-oriented paradigm is different to a procedural paradigm.
- Know different object-oriented key terminology.
- Know the meaning of a class and object and be able to create these within a programming language.
- Know the purpose of a constructor and create constructor methods to instantiate objects.
- Design classes to use encapsulation.
- Use getter and setter methods to access data stored in a class.

Suggested timing

10 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What is a programming paradigm?
 - Which programming paradigms do you know?
 - What is a procedural paradigm?
 - How is a procedural paradigm structured?
 - Why is a procedural paradigm used?
 - What do you think an object-oriented paradigm is?
 - How is this different to a procedural paradigm?

- **Research:** Ask students to research a definition of the key object-oriented terms: class, property/attribute, method, object, instantiation, encapsulation, inheritance, overriding and association.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Demonstration:** Ensure students fully understand the difference between a class, object and constructor method.
- Next demonstrate how to implement these concepts within the chosen programming language
- **Practical Exercises:** Give students different consolidation exercises that allow them to:
 - Create classes
 - Create constructor methods
 - Instantiate objects from a class
 - Use getter and setter methods
- **Teacher Demonstration:** Ensure students fully understand what is meant by the term encapsulation.
- Next demonstrate how to implement encapsulation within the chosen programming language.
- **Practical Exercises:** Give students different consolidation exercises that allow them to create classes that use encapsulation within the chosen programming language.

Resources

- Programming consolidation exercises to allow students to implement the different object-oriented features.
- Notes on object-oriented programming:
 - 101computing.net/object-oriented-programming-concepts/
 - adacomputerscience.org/topics/object_oriented_programming?examBoard=all&stage=all
 - youtube.com/watch?v=iE5Tga8Jhw
 - youtube.com/watch?v=37cXZnli3RE
 - youtube.com/watch?v=oCEaOJgorWo

3.9.3 Relationships between classes

Notes:

- Students using C# and VB.Net will need to declare a method as virtual so that it can be overridden in a subclass.
- The public access specifier will be indicated by the + symbol, the private access specifier by the - symbol and the protected access specifier by the # symbol.

Specification content

- Know that inheritance is a relationship between classes in which one class is a more specialised version of an existing class.
- Be able to create classes using inheritance.
- Know that when inheritance is used there is a base class and a subclass.
- Be able to use the protected access modifier to determine which methods and properties of a base class can be accessed in subclasses without being accessible to objects in other classes.
- Know what overriding is and be able to use overriding.
- Know that association is a relationship between two objects in which one object can make use of another.
- Be able to use association.
- Be familiar with the use of class diagrams to show inheritance and association relationships.

Learning outcomes

- Know what is meant by the term inheritance, base class, subclass and how to implement inheritance.
- Know what is meant by the term overriding and association and be able to use these techniques.
- Be able to draw class diagrams to show the design of classes including the use of inheritance and association.

Suggested timing

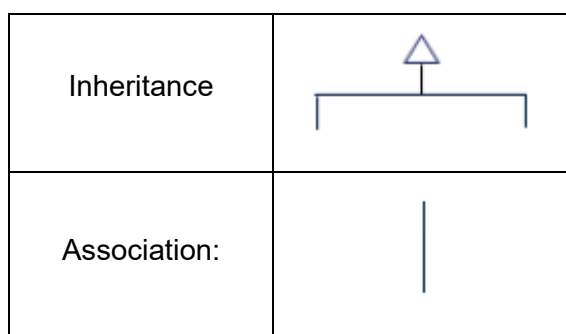
7 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What is inheritance?
 - What do you think the benefits of this are?
 - What is overriding?
 - What is association?
- **Teacher Input:** Remind students the meaning of classes and objects. Ensure students know the meaning of inheritance, a base class and a subclass.

- **Teacher Demonstration:** Demonstrate how to implement inheritance within the chosen programming language showing how a subclass can inherit attributes and methods from a base class.
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement inheritance within the chosen programming language.
- **Teacher Input:** Tell students the meaning of overriding and association.
- **Teacher Demonstration:** Demonstrate how to implement overriding and association within the chosen programming language.
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement overriding and association within the chosen programming language.
- **Teacher Demonstration:** Demonstrate how to draw class diagrams to show the design of a program including the use of inheritance and association relationships using these relationships:



- **Individual/Group/Paired Work:** Give students consolidation exercises containing questions asking students to draw and interpret class diagrams.

Resources

- Programming consolidation exercises to allow students to implement the different object-oriented features.
- Notes on inheritance, overriding, association and class diagrams:
 - 101computing.net/object-oriented-programming-concepts/
 - adacomputerscience.org/topics/object_oriented_programming?examBoard=all&stage=all
 - youtube.com/watch?v=oIU7GBse_yU
 - youtube.com/watch?v=lahWUiYY-qc

3.9.4.1 Files

Specification content

- Be able to read/write from/to a text file.

Learning outcomes

- Implement code to read/write data to/from text files.

Suggested timing

1.5 hours

Guidance

Possible teaching activities include:

- **Teacher Discussion:** Start by asking students key questions such as:
 - Why do you think programs need to use external files?
 - Why types of external files do you already know?
- **Teacher Input:** Focus on text files. Tell students the difference between read mode, write mode and append mode.
- **Teacher Demonstration:** Demonstrate how to write data to a text file within the chosen programming language and basic manipulation techniques such as:
 - Inserting new lines
 - Modifying specific lines
 - Deleting specific lines
 - Searching and replacing text
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement code to write data to a text file within the chosen programming language.
- **Teacher Demonstration:** Demonstrate how to read data to a text file within the chosen programming language including a single line and multiple lines.
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement code to read data from a text file within the chosen programming language.

Resources

- Programming consolidation exercises to allow students to implement code to read/write data to/from a text file.
- Notes on reading/writing data to/from a text file:
 - <https://www.youtube.com/watch?v=HQ--D7GTsbs>
 - <https://www.geeksforgeeks.org/reading-writing-text-files-python/>
 - <https://learnlearn.uk/alevelcs/writing-data-txt-files/>

3.9.4.2 Recursion

Note: Students should be able to read and write code that uses recursion and be able to trace recursive algorithms.

Specification content

- Know that a recursive subroutine is a subroutine that calls itself.
- Know that recursive subroutines have a base and a recursive case.
- Be able to solve simple problems using recursion.

Learning outcomes

- Know what is meant by the term recursion including its features.
- Know how to write programming code that implements recursion.

Suggested timing

4 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Give students a definition of recursion and tell students the difference between a base case and a recursive case and why these are needed.
- **Teacher Demonstration:** Demonstrate how to trace a recursive algorithm to show students how recursion works.
- Next demonstrate how to implement recursion within the chosen programming language.
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement recursion within the chosen programming language.
- **Class Discussion:** Ask students key questions such as:
 - How is recursion different to iteration?
 - Why do you think a programmer would choose to use recursion instead of iteration?
- **Research Work:** Ask students to research the benefits and drawbacks of using recursion instead of an iterative approach.
- **Present findings:** Students should present their findings from their research work.

Resources

- Programming consolidation exercises to allow students to implement recursion.
- Notes on recursion:
- learnlearn.uk/alevelcs/recursion/
- revisionworld.com/a2-level-level-revision/computing-0/algorithms/recursion
- adacomputerscience.org/concepts/recurs_basics?examBoard=all&stage=all

3.10 Advanced data structures and 3.11 Advanced algorithms

3.10.1 Graphs

3.11.1 Graph algorithms

Notes:

- Students should be familiar with the use of an adjacency matrix to represent both weighted and unweighted graphs.
- Students will not be expected to recall the steps in Dijkstra's shortest path algorithm or to write program code to implement it, unless the algorithm is given to them on a question paper.

Specification content

- Be aware of a graph as a data structure used to represent complex relationships.
- Be familiar with typical uses for graphs.
- Be able to explain the terms: graph, weighted graph, vertex/node, edge/arc, undirected graph and directed graph.
- Know how an adjacency matrix and an adjacency list may be used to represent a graph.
- Be able to compare the use of adjacency matrices and adjacency lists.
- Be able to trace breadth-first and depth-first search algorithms and describe typical applications of both.
- Be able to implement breadth-first and depth-first search algorithms in program code.
- Understand and be able to trace Dijkstra's shortest path algorithm.
- Be aware of applications of shortest path algorithm.

Learning outcomes

- Know what is meant by the term graph data structure, their typical uses and how they can be represented in an adjacency matrix and adjacency list.
- Know the key terminology associated with graph data structures.
- Know how to trace breadth-first and depth-first search algorithms and know example uses of these.
- Know how to implement breadth-first and depth-first search algorithms in program code.
- Be able to trace Dijkstra's shortest path algorithm and know example uses of this.

Suggested timing

7 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:

[oxfordaqa.com](https://www.oxfordaqa.com)

- What is a data structure?
 - What data structures do you already know?
- **Teacher Input:** Tell students what is meant by a graph data structure including the meaning of the key terminology associated with graphs: weighted graph, vertex/node, edge/arc, undirected graph and directed graph.
- **Research Work:** Ask students to research typical uses of graph data structures.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Demonstration:** Demonstrate how to represent graphs using an adjacency matrix and an adjacency list.
- Tell students what is meant by the term traversal and the difference between breadth-first and depth-first graph search algorithms.
- Next, demonstrate how to trace both breadth-first and depth-first graph search algorithms.
- **Individual/Group/Paired Work:** Give students different consolidation exercises to practice tracing different graph data structures using breadth-first and depth-first graph algorithms.
- **Research Work:** Ask students to research typical uses of breadth-first and depth-first graph algorithms.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Demonstration:** Demonstrate how to implement graphs including breadth-first and depth-first algorithms within the chosen programming language.
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement graph data structures within the chosen programming language.
- **Teacher Demonstration:** Tell students the purpose of the Dijkstra's shortest path algorithm.
- Next demonstrate how to trace the Dijkstra's shortest path algorithm on an example graph.
- **Practical Exercises:** Give students different consolidation exercises that allow them to trace the Dijkstra's shortest path algorithm on different graphs.
- **Research Work:** Ask students to research typical uses of the Dijkstra's shortest path algorithm.
- **Present findings:** Students should present their findings from their research work.

Resources

- Programming consolidation exercises to allow students to implement graph data structures including breadth-first and depth-first algorithms.
- Consolidation exercises to allow students to trace different breadth-first and depth-first algorithms and the Dijkstra's shortest path algorithm.
- Notes on graph data structures:
 - [geeksforgeeks.org/graph-data-structure-and-algorithms/](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/)
 - adacomputerscience.org/concepts/struct_graph?examBoard=all&stage=all
 - youtube.com/watch?v=PMPMDZqeipw&list=PLCiOXwirraUD0G290WrVKpVYd3IeGRRMW&index=7

3.11.2 Tree algorithms

Note: A class hierarchy in object-oriented programming is an example of a rooted tree, with Object as the root and all other classes descending from it.

Specification content

- Know that a tree is a connected, undirected graph with no cycles.
- Know that a rooted tree is a tree in which one vertex has been designated as the root. A rooted tree has parent-child relationships between nodes. The root is the only node with no parent and all other nodes are descendants of the root.
- Know that a binary tree is a rooted tree in which each node has at most two children.
- Understand how a binary tree can be used as a binary search tree.
- Be able to trace the tree-traversal algorithms: pre-order, post-order and in-order.
- Understand how to implement pre-order, post-order and in-order tree-traversal algorithms.
- Be able to describe uses of tree-traversal algorithms.

Learning outcomes

- Know what is meant by the term tree data structure and binary tree.
- Know how to trace and implement pre-order, post-order and in-order tree traversals within a programming language.
- Know different uses of the different tree-traversal algorithms.

Suggested timing

4.5 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students what is meant by a tree data structure, how to identify the root node and the meaning of a parent and child.
- Next tell students what is meant by the term binary search tree and how they are different to general tree structures.
- **Teacher Demonstration:** Demonstrate how to trace pre-order, post-order and in-order tree traversals.
- **Individual/Group/Paired Work:** Give students different consolidation exercises to practice tracing different tree data structures using pre-order, post-order and in-order tree traversals.
- **Teacher Demonstration:** Demonstrate how to implement trees including pre-order, post-order and in-order tree traversals within the chosen programming language.
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement tree data structures within the chosen programming language.

- **Research Work:** Ask students to research typical uses of pre-order, post-order and in-order tree traversals.
- **Present findings:** Students should present their findings from their research work.

Resources

- Programming consolidation exercises to allow students to implement tree data structures including pre-order, post-order and in-order tree traversals.
- Consolidation exercises to allow students to trace different pre-order, post-order and in-order tree traversals.
- Notes on tree data structures:
 - [geeksforgeeks.org/tree-data-structure/](https://www.geeksforgeeks.org/tree-data-structure/)
 - studysmarter.co.uk/explanations/computer-science/data-structures/tree-data-structure/
 - [youtube.com/watch?v=0lxX428YFfw](https://www.youtube.com/watch?v=0lxX428YFfw)
 - [youtube.com/watch?v=OyOn95AOXSM](https://www.youtube.com/watch?v=OyOn95AOXSM)
 - [youtube.com/watch?v=4Tg0Fg6qzJY](https://www.youtube.com/watch?v=4Tg0Fg6qzJY)

3.10.3 Hash tables

Specification content

- Be familiar with the concept of a hash table and its uses.
- Know what a hash function is.
- Be able to apply simple hash functions.
- Understand the properties of a good hash function.
- Know what is meant by a collision and how collisions are handled using rehashing.
- Understand the advantages and disadvantages of using a hash table as an alternative to an array.

Learning outcomes

- Know what is meant by the term hash table and its typical uses.
- Know what is meant by the term hash function, properties of a good hash function and how collisions can be handled.
- Know the advantages and drawbacks of a hash table.

Suggested timing

3 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students what is meant by the term hash table.
- **Research Work:** Ask students to research typical uses of a hash table.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Demonstration:** Tell students what is meant by the term hash function and demonstrate applying simple hash functions.
- **Research Work:** Ask students to research the properties of a good hash function and the advantages and drawbacks of hash tables as an alternative to an array.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Demonstration:** Tell students what is meant by the term collision and demonstrate how to handle a collision by storing the data in the next available position.
- **Individual/Group/Paired Work:** Give students different consolidation exercises to practice applying different hash functions and handling collisions.

Resources

- Consolidation exercises to allow students to practice applying different hash functions and handling collisions.
- Notes on hash tables and hash functions:
 - [youtube.com/watch?v=lyMIQvAiwc](https://www.youtube.com/watch?v=lyMIQvAiwc)
 - [youtube.com/watch?v=KI70jjzXI0s](https://www.youtube.com/watch?v=KI70jjzXI0s)
 - adacomputerscience.org/concepts/struct_hash_table?examBoard=all&stage=all
 - baeldung.com/cs/hash-tables

3.10.4 Priority queues

Specification content

- Know what a priority queue is.
- Be able apply the following operations to data stored in a priority queue: add an item (enqueue) and remove an item (dequeue).
- Be able to describe situations in which a priority queue is an appropriate data structure to use.
- Understand how to implement a priority queue using a one-dimensional array.
- For implementations in a one-dimensional array, understand how to test if the priority queue is empty or full.

Learning outcomes

- Know what is meant by the term priority queue and be able to add and remove items.
- Know when a priority queue would be used.
- Know how to implement a priority queue using a one-dimensional array including how to test if the priority queue is empty or full.

Suggested timing

3 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell student what is meant by the term priority queue.
- **Teacher demonstration:** Demonstrate how to add an item (enqueue) and remove an item (dequeue) items from a priority queue.
- **Research Work:** Ask students to research situations in which a priority queue is an appropriate data structure to use.
- **Present findings:** Students should present their findings from their research work.
- **Teacher demonstration:** Demonstrate how to implement a priority queue within the chosen programming language using one-dimensional arrays. Show students how to test if the priority queue is empty or full.
- **Practical Exercises:** Give students different consolidation exercises that allow them to implement a priority queue within the chosen programming language.

Resources

- Programming consolidation exercises to allow students to implement a priority queue.
- Notes on priority queues:
 - [geeksforgeeks.org/priority-queue-set-1-introduction/](https://www.geeksforgeeks.org/priority-queue-set-1-introduction/)
 - baeldung.com/cs/priority-queue
 - studysmarter.co.uk/explanations/computer-science/data-structures/priority-queue/

3.10.5 Dictionaries

Specification content

- Be familiar with the concept of a dictionary.
- Be familiar with simple applications of dictionaries.
- Have experience of using a programming language library that implements the dictionary data structure.

Learning outcomes

- Know what is meant by the term dictionary including example uses.

Suggested timing

3 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students what is meant by the term dictionary.
- **Research Work:** Ask students to research example uses of a dictionary.
- **Present findings:** Students should present their findings from their research work.
- **Practical Exercises:** Give students different consolidation exercises that allow them to use a programming language library to implement the dictionary data structure.

Resources

- Notes on dictionary data structures:
 - [youtube.com/watch?v=eiWhNptRztk](https://www.youtube.com/watch?v=eiWhNptRztk)
 - [geeksforgeeks.org/differences-between-array-and-dictionary-data-structure/](https://www.geeksforgeeks.org/differences-between-array-and-dictionary-data-structure/)

Unit 4: Advanced concepts and principles of computer science

3.12 Functional programming

3.12.2 Writing functional programs

3.12.3 Lists in functional programming

Note: The specification contains detailed guidance on this topic outlining what students are expected to know.

Specification content

- Please see the specification content outlined in the specification outlined in topic 3.12.1, 3.12.2 and 3.12.3.

Learning outcomes

- Know the functional programming paradigm.
- Know how to write functional programs.
- Know how lists are used in functional programs.

Suggested timing

8 hours

Guidance

The functional programming paradigm is almost certainly a new topic for students and may be new to teachers. Functional programs aim to avoid side-effects by not using local, global or static variables, this means that each block of code within a functional program can be executed without it affecting other blocks of code.

The behaviour of a functional program should not be determined by the history of execution of the rest of the program, only by the inputs to that particular function.

Functional programming makes use of recursive techniques, a topic that students often struggle with. Functional programs tend to avoid side effects, so the programs blocks can be independently tested and the execution of programs can be highly optimised, running on multiple processor cores at once.

Teachers could use a range of languages to introduce functional programming, including languages their students may already be familiar with such as C# or Python. Familiarity with this paradigm might take a little getting used to and it might be a good opportunity to introduce a new language such as Haskell or Lisp which some people find more suited to teaching the subject. Exam questions will use Haskell notation.

When dealing with list programming, students may benefit from having physical representations to work with through the use of numbered or alphabetised cards, where they can split the head from the tail. They can then work through coded examples of functional programming. List functions are a good place to introduce the idea of recursion, for example repeatedly removing the head of a list and checking the new head, until you find the item you are looking for. This is an example of a linear search.

Having physical dry wipe cards which students can write on may also help students understand the map, filter and reduce/fold instructions.

Resources

- Notes on functional programming:
 - physicsandmathstutor.com/computer-science-revision/a-level-aqa/functional-programming/
 - stem.org.uk/system/files/elibrary-resources/2018/01/Functional%20Programming%20in%20Haskell%20for%20A%20level%20teachers.pdf
 - youtube.com/watch?v=smOIsIkaFa0&t=15s
 - youtube.com/watch?v=pke7G2a_hfk
 - youtube.com/watch?v=kLh4cxvI290&list=PLCiOXwirraUCROzP0NztqBykYPO8p_ojLa&index=3
 - youtube.com/watch?v=fvt-cTMrjeg
 - youtube.com/watch?v=R2TYsEINXdQ&list=PLCiOXwirraUAHtUL7pwolStV8g-mLNvs4&index=2&t=10s
 - youtube.com/watch?v=Qs2IhjO1pkM&list=PLCiOXwirraUAHtUL7pwolStV8g-mLNvs4&index=3

3.13 Theory of computation

3.13.1 Finite state machines (FSMs)

3.13.2 Regular expressions and regular languages

Notes:

- For FSMs with output, only Mealy machines are required, not Moore machines.
- The specification contains metacharacters that students should be familiar with. Any other metacharacters used in an exam question will be explained as part of the question.

Specification content

- Be able to draw and interpret simple state transition diagrams and state transition tables for FSMs with output and without output.
- Be able to form and use simple regular expressions for string matching.
- Be able to describe the relationship between regular expressions and FSMs.
- Be able to write a regular expression to recognise the same language as a given FSM and design an FSM to recognise the same language as a given regular expression.
- Know that a language is called regular if it can be represented by a regular expression.
- Know that any regular language can be recognised by an FSM.
- Know that a regular expression can be written to describe the language recognised by any FSM.

Learning outcomes

- Know how to draw and interpret simple state transition diagrams and state transition tables.
- Know how to form and use simple regular expressions for string matching.
- Know the relationship between regular expressions and FSMs.
- Know how to write a regular expression to recognise the same language as a given FSM.
- Know when a language is called regular and that any regular language can be recognised by an FSM.
- Know that a regular expression can be written to describe the language recognised by any FSM.

Suggested timing

3.5 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students what is meant by the term Finite State Machine (FSM), state transition diagram and state transition table.

- **Teacher Demonstration:** Demonstrate how to draw simple state transition diagrams and state transition tables for FSMs.
- **Practical Exercises:** Give students different consolidation exercises that allow them to draw simple state transition diagrams and state transition tables for different FSMs.
- **Teacher Demonstration:** Demonstrate how to form and use simple regular expressions for string matching using the metacharacters as outlined in the specification:
 - * (0 or more repetitions)
 - + (1 or more repetitions)
 - ? (0 or 1 repetitions, ie optional)
 - | (alternation, ie or)
 - () to group regular expressions.
- **Teacher Input:** Tell students the relationship between regular expressions and FSMs.
- **Practical Exercises:** Give students different consolidation exercises that allow them to:
 - Write a regular expression to recognise the same language as a given FSM
 - Design an FSM to recognise the same language as a given regular expression
- **Research Work:** Ask students to research:
 - When a language is called regular.
 - How a regular language can be recognised by an FSM.
 - How regular expressions can be written to describe the language recognised by any FSM.
- **Present findings:** Students should present their findings from their research work.

Resources

- Consolidation exercises to allow students to practice drawing state transition diagrams and state transition tables and write regular expressions.
- Notes on FSMs:
 - adacomputerscience.org/concepts/machines_fsm?examBoard=all&stage=all
 - youtube.com/watch?v=9YnjgXmv6fU&t=335s
 - studyrrocket.co.uk/revision/a-level-computer-science-aqa/theory-of-computation/finite-state-machines-fsms-without-output
- Notes on regular expressions and regular languages:
 - youtube.com/watch?v=XfKOIWi420s
 - youtube.com/watch?v=VFIMYkzrE10
 - adacomputerscience.org/concepts/maths_regex?examBoard=all&stage=all
 - adacomputerscience.org/concepts/machines_regex?examBoard=all&stage=all

3.13.3 Turing machines

Note: Exam questions will only be asked about Turing machines that have one tape that is infinite in one direction. No distinction will be made between the input alphabet and the tape alphabet.

Specification content

- Know that a Turing machine can be viewed as a computer with a single fixed program.
- Be able to represent transition rules using a transition function, represent transition rules using a state transition diagram and hand-trace simple Turing machines.
- Understand the equivalence between a transition function and a state transition diagram.
- Be able to explain the importance of Turing machines and the Universal Turing machine (UTM) to the subject of computation.

Learning outcomes

- Know how a Turing machine can be viewed as a computer with a single fixed program.
- Know how to represent rules using a transition function, diagram and hand-trace simple Turing machines.
- Know the importance of Turing machines and the Universal Turing machine (UTM).

Suggested timing

2.5 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students the purpose of a Turing machine and how it can be viewed as a computer with a single fixed program, expressed using:
 - A finite set of states in a state transition diagram
 - A finite alphabet of symbols
 - An infinite tape with marked-off squares
 - A sensing read-write head that can travel along the tape, one square at a time.
- **Teacher Demonstration:** Demonstrate how to represent transition rules using a transition function and a state transition diagram.
- **Practical Exercises:** Give students different consolidation exercises that allow them to represent transition rules using a transition function and a state transition diagram.
- **Teacher Demonstration:** Demonstrate how to hand-trace simple Turing machines.
- **Practical Exercises:** Give students different consolidation exercises that allow them to hand-trace Turing machines.

- **Research Work:** Ask students to research the importance of Turing machines and the Universal Turing machine (UTM) to the subject of computation.
- **Present findings:** Students should present their findings from their research work.

Resources

- Consolidation exercises to allow students to practice representing transition rules using transition functions and state transition diagrams and to allow students to practice hand-tracing Turing machines.
- Notes on Turing machines:
 - adacomputerscience.org/concepts/machines_turing?examBoard=all&stage=all
 - youtube.com/watch?v=Ty57TncUSno
 - youtube.com/watch?v=zGWsOfw_F2g
 - geeksforgeeks.org/turing-machine-in-toc/

3.13.4 Backus-Naur Form (BNF) and syntax diagrams

Specification content

- Be able to check language syntax by referring to BNF or syntax diagrams and formulate simple production rules.
- Know that BNF can represent some languages that regular expressions cannot represent.

Learning outcomes

- Know how to check the language syntax using BNF rules and syntax diagrams and how to formulate simple production rules.
- Know how BNF can represent languages that regular expression cannot represent.

Suggested timing

2 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students what is meant by the term Backus-Naur Form (BNF) and cover basic rules around non-terminals, terminals, production rules etc.
- **Teacher Demonstration:** Demonstrate how to use BNF rules to define the syntax of a language and create simple production rules.
- **Practical Exercises:** Give students different consolidation exercises that allow them to use BNF to define the syntax of a language and create simple production rules.
- **Teacher Demonstration:** Demonstrate how to use syntax diagrams to check the syntax of a language.
- **Practical Exercises:** Give students different consolidation exercises that allow them to draw syntax diagrams to check the syntax of a language.
- **Research Work:** Ask students to research how BNF can represent some languages that regular expressions cannot represent.
- **Present findings:** Students should present their findings from their research work.

Resources

- Consolidation exercises to allow students to practice using BNF to check the syntax of a language, create simple production rules and draw syntax diagrams.
- Notes on BNF and syntax diagrams:
 - adacomputerscience.org/concepts/trans_bnf?examBoard=all&stage=all
 - geeksforgeeks.org/bnf-notation-in-compiler-design/
 - youtube.com/watch?v=x1qGlnKNCrW&t=30s

3.13.5.1 Comparing algorithms

3.13.5.2 Order of complexity

Specification content

- Understand that algorithms can be compared by expressing their complexity.
- Understand that some algorithms are more efficient than others time-wise and space-wise.
- Be familiar with Big O notation to express time complexity and be able to apply it to cases where the running time requirements of the algorithm grows.
- Be able to derive the time complexity of a simple algorithm.
- Know that algorithms may have a best and a worst-case time complexity and that these may be different.
- Know the best and worst-case time complexity of the following algorithms: linear search, binary search, bubble sort, merge sort, Dijkstra's shortest path algorithm and searching a binary search tree.

Learning outcomes

- Know that algorithms can be compared by expressing their complexity time-wise and space-wise.
- Know how to use Big O notation to derive the time complexity of algorithms.
- Know that algorithms may have a best and a worst-case time complexity and that these may be different.
- Know the best and worst-case time complexity of different standard algorithms.

Suggested timing

2.5 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - If two algorithms complete the same task, how do you decide which is better?
 - Why is it important to be able to compare algorithms?
- **Teacher Input:** Tell students that algorithms can be compared by expressing their complexity and tell students the difference between time complexity and space complexity.
- Next, tell students that Big O can be used to express the complexity of an algorithm. Tell students the meaning of:
 - Constant time – $O(1)$
 - Logarithmic time – $O(\log n)$
 - Linear time – $O(n)$
 - Polynomial time – $O(n^a)$
 - Exponential time – $O(a^n)$

- **Teacher Demonstration:** Demonstrate how to use the Big O complexities to determine the time complexity of a simple algorithm including how algorithms may have a best and a worst-case time complexity and how these may be different.
- **Practical Exercises:** Give students different consolidation exercises that allow them to use Big O notation to determine the time complexity of simple algorithms.
- **Research Work:** Ask students to research the best and worst-case time complexity of these standard algorithms:
 - Linear search
 - Binary search
 - Bubble sort
 - Merge sort
 - Dijkstra's shortest path algorithm
 - Searching a binary search tree
- **Present findings:** Students should present their findings from their research work.

Resources

- Consolidation exercises to allow students to practice using Big O notation to determine the time complexity of simple algorithms.
- Notes on Big O notation:
 - adacomputerscience.org/concepts/complex_big_o?examBoard=all&stage=all
 - 101computing.net/big-o-notation/
 - youtube.com/watch?v=Tv2fY9Smjjw
 - youtube.com/watch?v=G77RVWK_1wo&list=PLCiOXwirraUDwnyalzSD1aCvtUnw_UHkd
 - youtube.com/watch?v=sp9d7LIBJEM&list=PLCiOXwirraUDwnyalzSD1aCvtUnw_UHkd&index=2
 - youtube.com/watch?v=VTkqNgW_rm0&list=PLCiOXwirraUDwnyalzSD1aCvtUnw_UHkd&index=4

3.13.5.3 Classification of algorithmic problems

3.13.5.4 Computable and non-computable problems

Notes:

- Students will be expected to explain what a heuristic method is but do not need to be familiar with any specific heuristic techniques.
- Students will not be required to prove that the Halting problem is unsolvable.

Specification content

- Know that algorithms may be classified as being either tractable or intractable.
- Know that heuristic methods are often used when tackling intractable problems.
- Understand that a heuristic is a set of rules or knowledge about the problem domain.
- Be aware that some problems cannot be solved algorithmically.
- Describe the Halting problem.
- Understand the significance of the Halting problem for computation.

Learning outcomes

- Know the difference between tractable and intractable algorithms.
- Know how heuristic methods are used when tackling intractable problems.
- Know that some problems cannot be solved algorithmically.
- Know what is meant by the term Halting problem and the significance of this for computation.

Suggested timing

1.5 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students the difference between tractable and intractable algorithms.
 - Tractable – problems that have a polynomial (or less) time solution
 - Intractable – problems that have no polynomial (or less) time solution
- **Research Work:** Ask students to research:
 - What is meant by the term heuristics.
 - How heuristic methods are used when tackling intractable problems.
- **Present findings:** Students should present their findings from their research work.
- **Research Work:** Ask students to research:
 - Examples of problems that cannot be solved algorithmically.
 - What is meant by the term Halting problem.
 - The significance of the Halting problem for computation.

- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on tractable algorithms, intractable algorithms, heuristics and the Halting problem:
 - adacomputerscience.org/concepts/complex_limits?examBoard=all&stage=all
 - youtube.com/watch?v=LJvtQRtOUMA
 - youtube.com/watch?v=iSShQcn6Bh0

3.14 Networking and cyber security

3.14.1.1 Communication methods

3.14.1.2 Communication basics

Specification content

- Define serial and parallel transmission methods and discuss the advantages of serial over parallel transmission.
- Define and compare synchronous and asynchronous data transmission.
- Describe the purpose of start and stop bits in asynchronous data transmission.
- Define baud rate, bit rate, bandwidth, latency and protocol.
- Understand the difference between baud rate and bit rate.
- Understand the relationship between bit rate and bandwidth.

Learning outcomes

- Know what is meant by the terms serial and parallel transmission and the advantages of these.
- Know what is meant by the terms synchronous and asynchronous data transmission.
- Know the purpose of a start bit and stop bit in asynchronous data transmission.
- Know the meaning of baud rate, bit rate, bandwidth, latency and protocol including the key difference between baud rate and bit rate and the relationship between bit rate and bandwidth.

Suggested timing

2 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students the meaning of serial transmission, parallel transmission, synchronous transmission and asynchronous transmission.
- **Research Work:** Ask students to research:
 - The advantages of serial transmission over parallel transmission.
 - The purpose of the start and stop bits in asynchronous data transmission.
- **Present findings:** Students should present their findings from their research work.
- **Research Work:** Ask students to research:
 - The meaning of these key terms:
 - Baud rate
 - Bit rate
 - Bandwidth
 - Latency
 - Protocol
 - The difference between baud rate and bit rate.

- The relationship between bit rate and bandwidth.
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on the different communication key terms:
 - adacomputerscience.org/concepts/comms_types?examBoard=all&stage=all
 - geeksforgeeks.org/difference-between-synchronous-and-asynchronous-transmission/
 - youtube.com/watch?v=6KsEiwCZi4g&t=4s
 - youtube.com/watch?v=I4dSIZM7Qdk&t=52s

3.14.2.1 Types of networking between hosts

3.14.2.2 Thin-client versus thick-client computing

Specification content

- Explain the following and describe situations where they might be used: peer-to-peer networking and client-server networking.
- Compare and contrast thin-client computing with thick-client computing.
- Understand the different hardware and networking requirements of thin-client and thick-client systems.

Learning outcomes

- Know the meaning peer-to-peer networking and client-server networking and when they would be used.
- Know the meaning thin-client computing with thick-client computing and the different hardware and requirements for each.

Suggested timing

1.5 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What is a computer network?
 - What types of networks do you know?
 - What are the benefits and drawbacks of networking computers?
- **Teacher Input:** Tell students the meaning of:
 - Peer-to-peer networking - each computer has equal status.
 - Client-server networking - most computers are nominated as clients and one or more as servers. The clients request services from the servers, which provide these services, for example file server, email server.
- **Research Work:** Ask students to research:
 - The benefits and drawbacks of peer-to-peer networks and client-server networks.
 - Situations where of these would be used.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Tell students the meaning of:
 - Thin-client computing
 - Thick-client computing
- **Research Work:** Ask students to research the hardware and networking requirements of thin-client and thick-client systems.

- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on peer-to-peer and client-server networking:
 - [youtube.com/watch?v=Z67WhfeBlgg](https://www.youtube.com/watch?v=Z67WhfeBlgg)
 - [geeksforgeeks.org/difference-between-client-server-and-peer-to-peer-network/](https://www.geeksforgeeks.org/difference-between-client-server-and-peer-to-peer-network/)
 - [studyrocket.co.uk/revision/gcse-computer-science-ocr/networks/client-server-and-peer-to-peer-networks](https://www.studyrocket.co.uk/revision/gcse-computer-science-ocr/networks/client-server-and-peer-to-peer-networks)
- Notes on thin-client computing with thick-client computing:
 - [geeksforgeeks.org/difference-between-thin-clients-and-thick-clients/](https://www.geeksforgeeks.org/difference-between-thin-clients-and-thick-clients/)
 - en.wikibooks.org/wiki/A-level_Computing/AQA/Paper_2/Fundamentals_of_communication_and_networking/Thin_versus_thick_client_computing
 - [youtube.com/watch?v=03n1riV0UO8&t=2s](https://www.youtube.com/watch?v=03n1riV0UO8&t=2s)

3.14.2.3 Wired networking

3.14.2.4 Wireless networking

3.14.2.5 Compare wired and wireless networking

Specification content

- Compare copper and fibre-optic cables.
- Be able to describe how the protocol Carrier Sense Multiple Access with Collision Detection (CSMA/CD) works.
- Explain the purpose of Wi-Fi.
- Be familiar with the purpose of a Service Set Identifier (SSID).
- Be familiar with the components required for wireless networking.
- Be familiar with how wireless networks are secured.
- Be able to describe how the wireless protocol Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with and without Request to Send/Clear to Send (RTS/CTS) works.
- Compare the advantages and disadvantages of wireless networking compared to wired networking.

Learning outcomes

- Know the difference between copper and fibre-optic cables.
- Know how CSMA/CD and RTS/CTS works.
- Know the purpose of Wi-Fi and SSID.
- Know the components required for wireless networking.
- Know how to secure wireless networks.
- Know the advantages and disadvantages of wireless networking compared to wired networking.

Suggested timing

2.5 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students what the difference is between wired and wireless networking.
- **Teacher Input:** Once students understand the difference between these two methods, introduce students to copper cables and fibre-optic cables.
- **Research Work:** Ask students to research:
 - The differences between copper cables and fibre-optic cables in terms of:
 - Cost
 - Speed
 - Capacity

- The meaning of CSMA/CD and how this works when transmitting data on a wired network.
 - The benefits and drawbacks of wired networks.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Tell students what is meant by the term Wi-Fi including how they are based on international standards.
- **Research Work:** Ask students to research:
 - The purpose of a Service Set Identifier (SSID).
 - The components required for wireless networking including:
 - The wireless network adapter.
 - The wireless access point.
 - How wireless networks can be secured including:
 - Strong encryption of transmitted data using WPA2.
 - SSID (Service Set Identifier) broadcast disabled.
 - MAC (Media Access Control) address allow list.
 - The use of CSMA/CA and RTS/CTS.
 - The benefits and drawbacks of wireless networks.
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on wired and wireless networks:
 - [geeksforgeeks.org/difference-between-fiber-optic-cable-and-copper-wire/](https://www.geeksforgeeks.org/difference-between-fiber-optic-cable-and-copper-wire/)
 - [youtube.com/watch?v=03cdhk8WruE](https://www.youtube.com/watch?v=03cdhk8WruE)
 - [youtube.com/watch?v=WpSZBu9i108](https://www.youtube.com/watch?v=WpSZBu9i108)
 - [studyrocket.co.uk/revision/a-level-computer-science-aqa/fundamentals-of-communication-and-networking/wireless-networking](https://www.studyrocket.co.uk/revision/a-level-computer-science-aqa/fundamentals-of-communication-and-networking/wireless-networking)

3.14.3 The Internet

Specification content

- Understand the structure of the Internet.
- Understand the role of packet switching and routers.
- Know the main components of a packet.
- Explain how routing is achieved across the Internet.
- Describe the term 'uniform resource locator' (URL) in the context of internetworking.
- Explain the terms 'fully qualified domain name' (FQDN), 'domain name' and 'IP address'.
- Describe how domain names are organised.
- Understand the purpose and function of the domain service and its reliance on the Domain Name System (DNS).
- Be able to outline how the Domain Name System works using Domain Name Servers.
- Explain the service provided by Internet registries and why they are needed.

Learning outcomes

- Know what is meant by the term Internet and the role of packet switching and routers.
- Know the components of a data packet and how these are routed across the internet.
- Know what is meant by the terms URL, FQDN, domain name, IP address and DNS.
- Know how the DNS works and the services provided by Internet registries.

Suggested timing

2.5 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What is the Internet?
 - How is the Internet different to the World Wide Web (WWW)?
- **Teacher Input:** Tell students:
 - A definition of the Internet.
 - What is meant by the term data packet.
 - The role of packet switching and routers when transmitting data packets.
- **Research Work:** Ask students to research the main components of a packet including:
 - Source address
 - Destination address
 - Packet sequence number
 - Time to live
 - Payload
 - Error detection/correction information
- **Present findings:** Students should present their findings from their research work.

- **Research Work:** Next, ask students to extend their research by finding:
 - The meaning of these key terms:
 - URL
 - FQDN
 - Domain name
 - IP address
 - DNS
 - How domain names are organised.
 - How the DNS works using Domain Name Servers.
 - The service provided by Internet registries.
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on the structure of the internet:
 - en.wikibooks.org/wiki/A-level_Computing/AQA/Computer_Components,_The_Stored_Program_Concept_and_the_Internet/Structure_of_the_Internet/Packet_switching
 - youtube.com/watch?v=YLiY-nUSkuE
 - youtube.com/watch?v=80XDMwDthuE
 - youtube.com/watch?v=5XPB4HqIFWU

3.14.4.1 The TCP/IP model

3.14.4.2 Standard application layer protocols

Specification content

- Describe the role of the four layers of the TCP/IP stack (application, transport, internet, link).
- Understand why networking protocols like TCP/IP use layers.
- Describe the role of sockets in the TCP/IP stack.
- Be familiar with the role of MAC (Media Access Control) addresses.
- Be familiar with the following protocols: FTP, HTTP, HTTPS, IMAP, POP3, SMTP and SSH.
- Be familiar with FTP client software and an FTP server, when transferring files using anonymous or authenticated access.
- Be aware that FTP is being replaced by alternative protocols such as SFTP that encrypt files for secure transmission.
- Be familiar with how SSH is used for remote management.
- Be familiar with using SSH to log in securely to a remote computer and execute commands.
- Describe the role of an email server when sending or receiving emails.
- Describe how SMTP and POP3 or IMAP protocols are used when an email is sent.
- Describe the role of a web server in serving up web pages in text form.
- Understand the role of a web browser in retrieving web pages and web page resources and rendering these accordingly.

Learning outcomes

- Know the role of the four layers in the TCP/IP stack, why protocols use layers and the role of sockets.
- Know the role of a MAC address.
- Know the purpose of a range of different networking protocols.
- Know the role of a web server in serving up web pages in text form.
- Know the role of a web browser in retrieving web pages and web page resources.

Suggested timing

2.5 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Introduce students to the TCP/IP stack including why it is used.
- **Research Work:** Ask students to research:
 - The role of the four layers of the TCP/IP stack (application, transport, internet, link).
 - Why networking protocols like TCP/IP use layers.
 - The role of sockets in the TCP/IP stack.
 - The role of MAC (Media Access Control) addresses.

- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Introduce students to the HTTP and HTTPS protocols and why they are used.
- Next, introduce students to the FTP protocol.
- **Research Work:** Ask students to research:
 - The purpose of the FTP protocol.
 - The use of FTP client software and an FTP server when transferring files using anonymous or authenticated access.
 - How FTP is being replaced by alternative protocols such as SFTP that encrypt files for secure transmission.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Introduce students to the SSH protocol.
- **Research Work:** Ask students to research:
 - The purpose of the SSH protocol.
 - How SSH is used for remote management.
 - How to use SSH to log in securely to a remote computer and execute commands.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Introduce students to the IMAP, POP3, SMTP protocols.
- **Research Work:** Ask students to research:
 - The purpose of the IMAP, POP3, SMTP protocols.
 - The role of an email server when sending or receiving emails.
 - How SMTP and POP3 or IMAP protocols are used when an email is sent.
- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Introduce learners to the concept of a webserver.
- **Research Work:** Ask students to research:
 - 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.
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on the TCP/IP Stack:
 - adacomputerscience.org/concepts/internet_tcp_ip?examBoard=all&stage=all
 - 101computing.net/tcp-ip-stack-network-layers-and-protocols/
 - youtube.com/watch?v=R5aJSWucbe8
- Notes on the standard application protocols:
 - geeksforgeeks.org/types-of-internet-protocols/
 - learnlearn.uk/alevelcs/networking-protocols/
 - youtube.com/watch?v=n8anpB3dvsQ
 - youtube.com/watch?v=ndqlayKWNp0
 - youtube.com/watch?v=vFq3C3jiZLw
 - youtube.com/watch?v=H17xGNGdZIM
 - youtube.com/watch?v=63Uk7qiaq_4

3.14.4.3 IP addresses

3.14.4.4 Dynamic Host Configuration Protocol (DHCP)

Specification content

- Know that an IP address is split into a network identifier part and a host identifier part.
- Know that networks can be divided into subnets and know how a subnet mask is used to identify the network identifier part of the IP address.
- Know that there are currently two standards of IP address, v4 and v6.
- Know why v6 was introduced.
- Distinguish between routable and non-routable IP addresses.
- Understand the purpose of the DHCP system.
- Be able to describe how the DHCP system works.
- Understand the advantages of the DHCP system over manual configuration.

Learning outcomes

- Know what is meant by the term IP address and how these are structured.
- Know how networks can be divided into subnets and how subnet masks are used.
- Know what is meant by IPv4 and IPv6 and why IPv6 was introduced.
- Know the difference between routable and non-routable IP addresses.
- Know the purpose of the DHCP system, how it works and the advantages of the DHCP.

Suggested timing

2 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What does IP stand for?
 - What is an IP address used for?
 - How is this different to a MAC address?
- **Research Work:** Ask students to research:
 - The structure of an IP address.
 - How networks can be divided into subnets.
 - How a subnet mask is used to identify the network identifier part of the IP address.
 - The differences between IPv4 and IPv6.
 - Why IPv6 was introduced.
 - The difference between routable and non-routable IP addresses.
- **Present findings:** Students should present their findings from their research work.

- **Teacher Input:** Introduce students to the purpose of DHCP.
- **Research Work:** Ask students to research:
 - How the DHCP system works.
 - The advantages of the DHCP system over manual configuration.
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on IP addresses:
 - adacomputerscience.org/concepts/internet_ip_addresses?examBoard=all&stage=all
 - learnlearn.uk/alevelcs/subnet-masks/
 - youtube.com/watch?v=xa5MXKQbfWQ&t=3s
 - youtube.com/watch?v=47FXI_sYrpY
 - youtube.com/watch?v=EOhBzhs0ACU
- Notes on DHCP:
 - baeldung.com/cs/dhcp-intro
 - youtube.com/watch?v=4pMII6ZwZ0k

3.14.5 Cyber security

Specification content

- Understand how a firewall works (packet filtering, proxy server, stateful inspection).
- Understand symmetric and asymmetric (private/ public key) encryption and key exchange.
- Understand how digital certificates and digital signatures are obtained and used.
- Discuss worms, trojans and viruses, and the vulnerabilities that they exploit.

Learning outcomes

- Know how a firewall works.
- Know how symmetric and asymmetric encryption works.
- Know the role of digital certificates and digital signatures.
- Know different threats that face computer systems and networks.

Suggested timing

2 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What cyber security threats are you aware of?
 - What risks do these threats impose?
- **Research Work:** Ask students to research common threats that face computer systems and networks and the vulnerabilities that they exploit including:
 - Worms
 - Trojans
 - Viruses
- **Present findings:** Students should present their findings from their research work.
- **Research Work:** Ask students to research cyber security threat prevention methods including:
 - Firewalls:
 - Packet filtering
 - Proxy server
 - Stateful inspection
 - Encryption:
 - Aymmetric
 - Asymmetric (private/ public key)
 - Key exchange
 - Digital certificates and digital signatures:
 - How these are obtained and used
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on cyber security threats and prevention methods:
 - adacomputerscience.org/topics/malicious_code?examBoard=all&stage=all
 - adacomputerscience.org/topics/security?examBoard=all&stage=all
 - adacomputerscience.org/topics/encryption?examBoard=all&stage=all
 - youtube.com/watch?v=GN6xLwEjgR0
 - youtube.com/watch?v=thESC86I2Ps
 - youtube.com/watch?v=nWF1MGy8tOA
 - youtube.com/watch?v=2T-Fdx_LWSw
 - youtube.com/watch?v=-ul83xZkJgw

3.15 Databases

3.15.1.1 Conceptual data models and entity relationship modelling

Specification content

- Produce a data model from given data requirements for a simple scenario involving multiple entities.
- Produce Entity Relationship Diagrams (ERD) representing a data model and entity descriptions in the form: Entity1(Attribute1, Attribute2, ...).

Learning outcomes

- Know how to produce a data model from given data requirements.
- Know the purpose of an entity relationship diagram and their structure.





Suggested timing

2 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Ask students key questions such as:
 - What is data?
 - What is a database?
 - Why do you think it's important to try and keep data organised?
- **Teacher Input:** Introduce students to the idea of an ERD and cover the different relationships that are used in ERDs.

One-to-many	
Many-to-one	
One-to-one	
Many-to-many	

- **Teacher Demonstration:** Demonstrate how to apply the relationships to different scenarios starting with two entities and then build this up to four or five entities. Show students how to do this using the format: Entity1(Attribute1, Attribute2, ...)
- **Practical Exercises:** Give students different consolidation exercises that allow them to draw ERDs for a range of different scenarios.

Resources

- Consolidation exercises to allow students to practice drawing ERDs for a range of different scenarios.
- Notes on ERDs:
 - adacomputerscience.org/concepts/dbs_entity?examBoard=all&stage=all
 - 101computing.net/tag/erd/
 - youtube.com/watch?v=fvkzrlMVt1s

3.15.1.2 Key concepts

Specification content

- Explain the concept of a relational database.
- Be able to define the terms: attribute/field, entity identifier/primary key, composite entity identifier/composite primary key, foreign key and relation/table.

Learning outcomes

- Know what is meant by the term relational database.
- Know the meaning of key database terminology.

Suggested timing

2 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students what is meant by the term relational database (i.e. a database containing several tables that are linked together using relationships).
- **Teacher Demonstration:** Show students a database containing two or three different tables. As a group draw the ERD for the database. Then, tell students the meaning of these key terms and add these onto the ERD:
 - Attribute/field
 - Entity identifier/primary key
 - Composite entity identifier/composite primary key
 - Foreign key and relation/table
- If suitable software is available within the centre, then demonstrate how to set up this database.
- **Practical Exercises:** Give students different consolidation exercises that allow them to identify the different keys (primary, composite and foreign) within a range of database scenarios.
- If suitable software is available within the centre, then allow students to have practical experiences of setting up a relational database.

Resources

- Consolidation exercises to allow students to identify the different keys (primary, composite and foreign) within a range of database scenarios.
- Notes on database concepts:
 - [youtube.com/watch?v=fvkzrlMVt1s&list=PLCiOXwirraUBzDcJq7WKHmnSAaTTLmCt1&index=1](https://www.youtube.com/watch?v=fvkzrlMVt1s&list=PLCiOXwirraUBzDcJq7WKHmnSAaTTLmCt1&index=1)

3.15.1.3 Database design and normalisation techniques

Specification content

- Be able to normalise relations to third normal form.
- Understand why databases are normalised.

Learning outcomes

- Know how to normalise relations to third normal form.
- Know why databases are normalised.

Suggested timing

2 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** If possible, show students data in first normal form with lots of redundant data. Ask students key questions such as:
 - What problem can you see with this data?
 - What additional problems does this create?
 - How do you think we can reduce this?
- **Teacher Input:** Introduce students to normalisation and cover the requirements of data in first, second and third normal form.
- **Teacher Demonstration:** Demonstrate how to apply normalisation to data by modelling how to put data into first, second and then third normal form.
- **Practical Exercises:** Give students different consolidation exercises that allow them to normalise different datasets into first, second and third normal form.
- After each normalisation task, students can then consolidate their knowledge of previous database topics by drawing an ERD diagram for the normalised database and identify the different keys (primary, composite and foreign).
- **Research Work:** Ask students to research why normalisation is used.
- **Present findings:** Students should present their findings from their research work.

Resources

- Consolidation exercises to allow students to normalise different datasets into first, second and third normal form.
- Notes on normalisation:
 - adacomputerscience.org/concepts/dbs_normalisation?examBoard=all&stage=all
 - freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/
 - youtube.com/watch?v=1YMdx97o1U8&list=PLCiOXwirraUBzDcJq7WKhmnSAaTTLMct1&index=2

3.15.1.4 Structured Query Language (SQL)

Note: The specification specifies the SQL commands and functions that students are expected to be familiar with.

Specification content

- Be able to use SQL to retrieve, update, insert and delete data from multiple tables of a relational database.
- Be able to use aggregate SQL functions.
- Be able to use SQL to define a database table.

Learning outcomes

- Know how to write SQL code to retrieve, update, insert and delete data from multiple tables in a database.
- Know how to use aggregate SQL functions.
- Know how to use SQL to define a database table

Suggested timing

4 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What does SQL stand for?
 - What is SQL used for?
 - What SQL commands / functions do you already know?
- **Teacher Input:** Tell students the purpose of these commands:
 - SELECT, FROM, WHERE, ORDER BY, GROUP BY
 - UPDATE
 - INSERT
 - DELETE
- **Teacher Demonstration:** Demonstrate how to write SQL code for these commands. This can be done using database management system tools, text editors or an IDE.
- **Practical Exercises:** Give students different consolidation exercises that allow them to write SQL code for these commands for a range of different databases.
- **Teacher Input:** Tell students the purpose of these functions:
 - COUNT
 - SUM
 - AVG
 - MIN
 - MAX

- **Teacher Demonstration:** Demonstrate how to write SQL code for these functions.
- **Practical Exercises:** Give students different consolidation exercises that allow them to write SQL code for these functions for a range of different databases.
- **Teacher Demonstration:** Demonstrate how to write SQL code to define a database table including:
 - Using data types for integer and real numbers, strings, Boolean values, dates and times.
 - Specifying the primary key.
 - Specifying foreign key constraints.
- **Practical Exercises:** Give students different consolidation exercises that allow them to write SQL code to define different database tables.

Resources

- Consolidation exercises to allow students to write SQL code for the different commands and functions stated within the specification.
- Notes on SQL:
 - [w3schools.com/sql/](https://www.w3schools.com/sql/)
 - sqlteaching.com/#!/select
 - adacomputerscience.org/topics/sql?examBoard=all&stage=all
 - filestore.aqa.org.uk/resources/computing/AQA-8525-TG-SQL.PDF
 - youtube.com/watch?v=lljuqq5h15M&list=PLCiOXwirraUBzDcJq7WKhmnSAaTTLmCt1&index=3

3.15.1.5 Client server databases

Specification content

- Know that a client server database system provides simultaneous access to the database for multiple clients (known as concurrent access).
- Understand that concurrent updates of a database can result in the lost update problem.
- Know how concurrent access can be controlled using record locks to preserve the integrity of a database.

Learning outcomes

- Know the purpose of a client server database.
- Know how concurrent updates of a database can result in the lost update problem.
- Know how concurrent access can be controlled using record locks to preserve the integrity of a database.

Suggested timing

1.5 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students what is meant by the term client server database.
- **Research Work:** Ask students to research:
 - The benefits of concurrent access.
 - The problems that may arise by several clients accessing a database at the same time.
 - What is meant by the lost update problem.
 - How concurrent access can be controlled with record locks.
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on concurrency problems:
 - [geeksforgeeks.org/concurrency-problems-in-dbms-transactions/](https://www.geeksforgeeks.org/concurrency-problems-in-dbms-transactions/)
 - medium.com/@bindubc/distributed-system-concurrency-problem-in-relational-database-59866069ca7c
 - [geeksforgeeks.org/types-of-locks-in-concurrency-control/](https://www.geeksforgeeks.org/types-of-locks-in-concurrency-control/)

3.15.2 Big data

Note: The specification contains additional information outlining what students are expected to know for this topic.

Specification content

- Know that 'Big Data' is a catch-all term for data that cannot be stored or processed using traditional methods and the meaning of volume, velocity and variety.
- Know that when data sizes are so big as not to fit on to a single server: the processing must be distributed across more than one machine and functional programming is a solution, because it makes it easier to write correct and efficient distributed code.
- Be familiar with the fact-based model for representing data, graph schema for capturing the structure of the dataset and nodes, edges and properties in graph schema.

Learning outcomes

- Know what is meant by Big Data in terms of volume, velocity and variety.
- Know why distributed processing and functional programming is used when handling Big Data.
- Know the fact-based model for representing data, graph schema for capturing the structure of the dataset and the meaning of nodes, edges and properties in graph schema.

Suggested timing

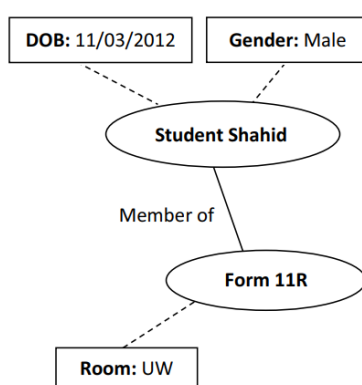
4 hours

Guidance

Possible teaching activities include:

- **Teacher Input:** Tell students that 'Big Data' is a catch-all term for data that cannot be stored or processed using traditional methods.
- **Research Work:** Ask students to research:
 - The meaning of these terms in relation to Big Data:
 - Volume
 - Velocity
 - Variety
 - The challenges that Big Data poses. Students should focus their research on:
 - How analysing the data is made significantly more difficult.
 - Why relational databases are not appropriate.
 - The use of distributed processing when handling Big Data.
 - The use of functional processing when handling Big Data. Students should focus their research on:
 - Immutable data structures
 - Statelessness
 - Higher-order functions that can combine the results of processing on different servers (map-reduce).

- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Tell students about:
 - Fact-based models for representing data and how each fact within a fact-based model captures a single piece of information.
 - Graph schema for capturing the structure of the dataset covering:
 - An individual entity about which data are stored is represented as an oval node.
 - The properties of an entity are drawn in rectangles, which are attached to the oval for the entity by a dashed line.
 - Relationships between the entities are represented by solid line edges drawn between the them, labelled with text to describe the nature of the relationship.
 - Nodes, edges and properties in graph schema. For example:



Resources

- Notes on Big Data:
 - adacomputerscience.org/topics/big_data?examBoard=all&stage=all
 - physicsandmathstutor.com/computer-science-revision/a-level-aqa/big-data/
 - pmt.physicsandmathstutor.com/download/Computer-Science/A-level/Notes/AQA/11-Big-Data/Advanced/11.%20Big%20Data%20-%20Advanced.pdf
 - youtube.com/watch?v=qGNiQCKNWU
 - youtube.com/watch?v=sDUVYO7-CpM

3.16 Artificial intelligence

3.16.1 Applications of artificial intelligence

3.16.2 Creating artificially intelligent systems

3.16.3 Benefits and risks of artificial intelligence

Note: The specification contains additional information outlining what students are expected to know for this topic.

Specification content

- Know the characteristics of artificial intelligence.
- Be aware that current artificial intelligence systems are not generally intelligent, but work in narrow fields.
- Be aware of the following common application areas for artificial intelligence: generative AI, search and recommendation systems, playing strategic games and medical diagnosis.
- Know what a neural network is and understand how neural networks can be used.
- Be able to describe the structure of a neural network, how backpropagation is commonly used to train a neural network and that deep learning systems use neural networks with several hidden layers of nodes to produce output and that using more layers of nodes allows for more complex problems to be solved.
- Know that machine learning is a type of artificial intelligence in which the performance of the system is improved based on experience.
- Know that artificially intelligent systems are often trained using data and that care must be taken when selecting the training data to ensure that a system does not develop bias.
- Understand the benefits associated with the use of artificial intelligence.
- Understand the risks associated with the use of artificial intelligence.

Learning outcomes

- Know the characteristics of artificial intelligence and fields that they work within.
- Know common application areas of artificial intelligence.
- Know the purpose of a neural network, its structure and why backpropagation is used.
- Know what is meant by the term machine learning and why care must be taken when selecting the training data.
- Know the benefits and risks associated with the use of artificial intelligence.

Suggested timing

4 hours

Guidance

Possible teaching activities include:

- **Class Discussion:** Start by asking students key questions such as:
 - What is artificial intelligence?
 - What do you use artificial intelligence for?
 - What have been your experiences (positive and negative) of using artificial intelligence?
- **Teacher Input:** Tell students that there is no one accepted definition of artificial intelligence, however, systems are often described as being artificially intelligent if they solve a complex problem and arrive at a solution:
 - Using a similar method to that which a human might follow, or
 - That is at least as good as a human might arrive at.
- **Research Work:** Ask students to research these application areas of artificial intelligence:
 - Generative AI
 - Search and recommendation systems
 - Playing strategic games
 - Medical diagnosis
- **Present findings:** Students should present their findings from their research work.
- **Teacher Input:** Tell students what is meant by the term neural network and how:
 - The nodes in a neural network are built up in layers.
 - The outputs of nodes in one layer are weighted to form the inputs to nodes in the next layer.
 - A simple neural network has three layers: an input layer, a hidden processing layer and an output layer
- **Research Work:** Ask students to research:
 - How backpropagation is commonly used to train a neural network.
 - Why more layers of nodes are used.
 - What is meant by machine learning.
 - Why care must be taken when selecting the training data to ensure that a system does not develop bias.
- **Present findings:** Students should present their findings from their research work.
- **Research Work:** Ask students to research the benefits associated with the use of artificial intelligence and the risks associated with the use of artificial intelligence. The specification lists example benefits and risks for student to research.
- **Present findings:** Students should present their findings from their research work.

Resources

- Notes on artificial intelligence:
 - teachcomputing.org/artificial-intelligence
 - en.wikipedia.org/wiki/Artificial_intelligence
 - cloud.google.com/discover/ai-applications
 - [en.wikipedia.org/wiki/Neural_network_\(machine_learning\)](https://en.wikipedia.org/wiki/Neural_network_(machine_learning))
 - tableau.com/data-insights/ai/advantages-disadvantages
 - briteris.com/blog/what-exactly-is-artificial-intelligence-understanding-the-risks-and-benefits-for-your-business/