A LEVEL COMPUTER SCIENCE SUMMARY

OLDHAM SIXTH FORM COLLEGE

Table of Contents

Component 1 2

1. Data structures 2
2. Logical operations 3
3. Algorithms and programs 4
4. Principles of programming 8
5. Systems analysis 10
6. System Design 13
7. Software engineering 14
8. Program construction 15
9. Economic, moral, legal, ethical and cultural issues relating to computer science 16

Component 2 18

1. Hardware and communication 18
2. Data transmission 21
3. Data representation and data types 22
4. Organisation and structure of data 23
5. Databases and distributed systems 25
6. The operating system 27
7. The need for different types of software systems and their attributes 29
8. Data security and integrity processes 31

Diagrams and Exam Conventions 33

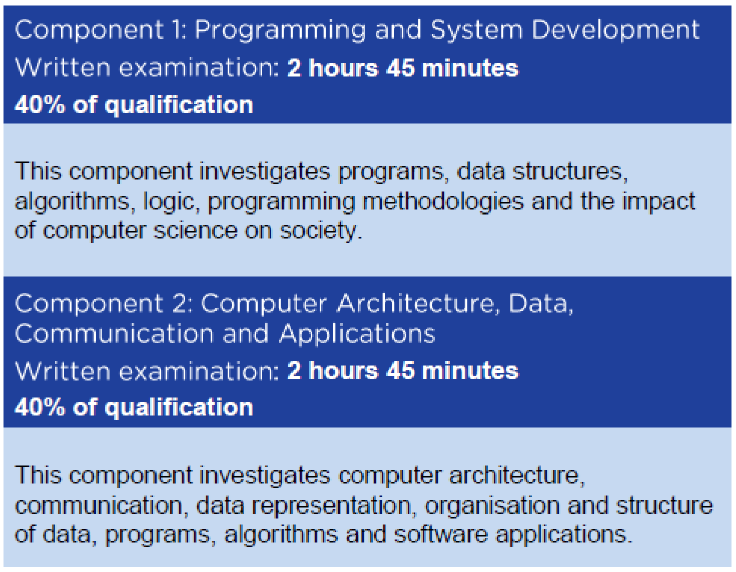
DFDs 33

ERDs 33

Von Neumann 33

Structured Query Language 34

Flowcharts 34



Component 1: Programming and System Development
Written examination: **2 hours 45 minutes**
40% of qualification

This component investigates programs, data structures, algorithms, logic, programming methodologies and the impact of computer science on society.

Component 2: Computer Architecture, Data, Communication and Applications
Written examination: **2 hours 45 minutes**
40% of qualification

This component investigates computer architecture, communication, data representation, organisation and structure of data, programs, algorithms and software applications.

Component 1

1. Data structures

Describe, interpret and manipulate data structures including arrays (up to three dimensions), records, stacks, queues, trees, linked lists and hash tables.

Describe the manipulation of records and arrays.

Represent the operation of stacks and queues using pointers and arrays.

Represent the operation of linked lists and trees using pointers and arrays.

Select, identify and justify appropriate data structures for given situations.

Summary

- Data structures are organised stores of data in a computer's memory. Each data structure has its own strengths and weaknesses that make it suitable for a given situation.
- Arrays hold data of a single type in rows and columns; they usually have one, two or three dimensions.
- The different sections in an array are called 'elements' and are accessed using an index (which always starts from 0).
- A record is a data structure that allows items of data of different types to be stored together in a single file.
- Lists can hold a range of data types but, as with arrays, their elements can be edited once they have been initiated.
- Linked lists represent each piece of data as a node which holds the data and a link to the next piece of data in the list.
- Data is removed from linked lists by deleting the node and pointing the item behind it to the item in front of it.
- Stacks are Last-In, First-Out data structures that hold data in the order in which it was entered.
- Data is removed or added at the top of the stack.
- Queues are First-In, First-Out data structures in which the last piece of data entered is the last piece of data retrieved.
- Tree data structures consist of nodes (leaves) and branches. Nodes contain data and the branches link nodes together.
- Hash tables use mapping functions to calculate where in the table a new piece of data should be stored.

2. Logical operations

Draw truth tables for Boolean expressions consisting of AND, OR, NOT, XOR, NAND and NOR logical operations.

Apply logical operations to combinations of conditions in programming, including clearing registers, masking, and encryption.

Simplify Boolean expressions using Boolean identities, rules and De Morgan's laws.

Summary

- Conjunction: both sides of the proposition must be true.
- Disjunction: either side of the proposition can be true.
- Negation: reverses the truth of a proposition.
- Implication: if something is true then we can infer that something else is also true.
- Commutation: the order does not matter when using conjunction or disjunction.
- Association: the order does not matter when we link multiple conjunctions or multiple disjunctions; this does not hold true if we start mixing conjunctions and disjunctions.
- Double negation: two negations cancel each other out.
- de Morgan's law: the negation of disjunctions is the conjunction of the negations.

commutation law for conjunction $P \,.\, Q \leftrightarrow Q \,.\, P$
commutation law for disjunction $P + Q \leftrightarrow Q + P$
association laws for conjunction $T \,.\, (P \,.\, Q) \leftrightarrow (T \,.\, Q) \,.\, P$
association laws for disjunction $T + (P + Q) \leftrightarrow (T + Q) + P$
distribution law 1 $T \,.\, (P + Q) \leftrightarrow (T \,.\, P) + (T \,.\, Q)$
distribution law 2 $T + (P \,.\, Q) \leftrightarrow (T + P) \,.\, (T + Q)$
de Morgan's law for disjunction $\overline{P + Q} \leftrightarrow \overline{P} \,.\, \overline{Q}$
de Morgan's law for conjunction $\overline{P \,.\, Q} \leftrightarrow \overline{P} + \overline{Q}$
double negative law $\overline{\overline{P}} \leftrightarrow P$

3. Algorithms and programs

	Explain the term algorithm and describe common methods of defining algorithms, including pseudo-code and flowcharts.
Variables and constants	Identify and explain the use of constants and variables in algorithms and programs.
Identifiers	Describe why the use of self-documenting identifiers, annotation and program layout are important in programs. Give examples of self-documenting identifiers, annotation and appropriate program layout.
Scope of variables	Describe the scope and lifetime of variables in algorithms and programs.
Parameters	Explain the purpose and effect of procedure calling, parameter passing and return, call by reference and call by value.
Recursion	Explain the use of recursion in algorithms and programs and consider the potential elegance of this approach.
Mathematical operations	Identify, explain and use mathematical operations in algorithms, including DIV and MOD.
Validation and verification	Identify, explain and apply appropriate techniques of validation and verification in algorithms and programs.
Sorting	Explain the need for a variety of sorting algorithms both recursive and non-recursive.
	Describe the characteristics of sorting algorithms: bubble sort, insertion sort, quicksort.
	Explain the effect of storage space required, number of comparisons of data items, number of exchanges needed and number of passes through the data on the efficiency of a sorting algorithm.
	Use Big O notation to determine the efficiency of different sorting algorithms in terms of their time and space requirements and to compare the efficiency of different sorting algorithms.
Searching	Explain and apply a linear search algorithm. Explain and apply the binary search algorithm. Explain and apply a shortest-path algorithm.
	Describe appropriate circumstances for the use of each searching technique. Use Big O notation to determine the efficiency of linear and binary searches in terms of execution time and space requirements and to compare the efficiency of different searching algorithms.

	Follow search and sort algorithms and programs and make alterations to such algorithms.
	Write search and sort algorithms and programs.
Programming constructs	Identify, explain and use sequence, selection and repetition in algorithms and programs.
	Identify, explain and use counts and rogue values in algorithms and programs.
	Follow and make alterations to algorithms and programs involving sequence, selection, and repetition.
	Write algorithms and programs involving sequence, selection, and repetition to solve non-standard problems.
Modular programming	Identify and explain the nature, use and possible benefits of standard functions, standard modules and user defined subprograms.
Logical operations in algorithms and programs	Identify, use and explain the logical operators AND, OR, NOT, XOR, NAND and NOR in algorithms and programs.
Traversal of data structures	Write and interpret algorithms used in the traversal of data structures.
Compression	Explain data compression and how data compression algorithms are used.
	Compare and explain the efficiency of data compression algorithms in terms of compression ratio, compression time, decompression time and saving percentage.
Testing	Select appropriate test data.
	Dry-run a program or algorithm in order to identify possible errors, including logical errors.
	Explain the purpose of a given algorithm by showing the effects of test data.
Comparing algorithms	Use Big O notation to determine the complexity and efficiency of given algorithms in terms of their execution time, their memory requirements and between algorithms that perform the same task.

Summary

- Algorithms are a set of mechanical steps which will take a given input and deterministically produce an output.
- Pseudocode is a method of representing algorithms which is not language specific.
- Flowcharts are another method of representing algorithms in a diagrammatic format and tend to be used for smaller algorithms.
- Variables are programming constructs which use identifiers to store information in RAM.
- The key difference between constants and variables is that once a constant has been assigned a value it cannot change while variables can be changed.
- Local scoped variables will only exist in the code block they were defined.
- Global scoped variables are available throughout the program.
- Self-documenting code is code which uses sensible identifiers, good layout and clear code to reduce the need for comments and other forms of documentation.
- Parameters provide input into procedures and functions; they become variables within the functions to allow the programmer access.
- Procedures do not return values while functions do.
- Passing parameters by value will create a copy of the value.
- Passing by reference means that if the parameter is changed then the original will also change.
- Recursions are functions which call themselves and is a method of iteration.
- Validation occurs while data is being inputted into a computer system. The common validation rules are:
- character check
- length check
- format check
- existence check
- check digit.
- Validation occurs after the data has been inputted. The two main methods are double entry and proof reading.
- Big O is a method of comparing algorithms together and measures the growth in terms of time or space.
- The order of growth in Big O, from smallest to largest, is shown in the table below–

Algorithm	Time complexity	Space complexity
Bubble sort	$O(n^2)$	$O(1)$
Insertion sort	$O(n^2)$	$O(1)$
Quicksort	$O(n \log_2 n)$	$O(\log_2 n)$
Linear search	$O(n)$	$O(1)$
Binary search	$O(\log_2 n)$	$O(\log_2 n)$
Dijkstra **graph** search	$O(\|V\|^2 + \|E\|)$	$O(\|E\| \log_2 \|V\|)$.

- There are three main sorting algorithms; bubble sort, insertion sort and quicksort.
- There are two main search algorithms; linear search and binary search.
- Dijkstra’s shortest path algorithm will find the shortest route in a given map if each edge has a weight.
- Sequence is the order in which programming statements are executed.
- Selection will choose one of two paths to follow based on a condition.
- Repetition or iteration will repeat a block of code until a condition is met. The three main types of iteration are for, while and repeat/until loops.
- Counters can be used to keep track of where in a loop we have reached and is commonly used as a method of stopping iteration.
- Rogue values can also be used to stop iteration by assigning an abnormal value to delimit a string of text or a list of data.
- Logical operators are used as part of conditions and will form part of a logical expression.
- Data compression will reduce the size of data. There are two types of compression; lossy, where data is lost so we get an approximation of the original data and lossless, where no data is lost.
- Run length encoding will replace strings of the same characters with a character count.
- Dictionary encoding will create a list of commonly occurring sequences and replace all occurrences with a shorter reference in the dictionary.
- Huffman encoding will use shorter binary values to represent letters which occur more frequently.

4. Principles of programming

	Explain the nature and relative advantages of different programming paradigms, and identify possible situations where they may be used.
	Describe the distinguishing features of different types of programming paradigms, including procedural, event-driven, visual and mark-up languages.
	Describe the role of an object-oriented approach to programming and the relationship between object, class and method.
	Describe the need for the standardisation of computer languages, and the potential difficulties involved in agreeing and implementing standards.
	Identify ambiguities in natural language and explain the need for computer languages to have an unambiguous syntax.
	Interpret and use formal methods of expressing language syntax: syntax diagrams and Backus-Naur form (extended Backus-Naur form is not to be used).
Levels of computer language	Describe the differences between high-level and low-level languages.
	Identify and describe situations that require the use of a high-level or a low-level language.
Types of computer language	Identify and justify which type of language would be best suited to develop a solution to a given problem.

Summary

- There are many different programming paradigms, each with their own strengths and weaknesses.
- Procedural programming languages describe which steps to take to solve a problem and how to complete each step.
- Assembly language instructions consist of two distinct sections: operators, such as ADD or SUB, and operands, which are usually memory addresses.
- Visual languages allow code to be created through drag and drop. They are commonly used for learning programming by students.
- Event-driven languages will run functions depending on what events are happening in the system, normally triggered by the user.
- Software standards define how code should be written with regards to the structure, naming of identifiers and how modules are created. They may also define the specifics of how a language works, but leave the implementation up to other vendors.
- Examples of standards include HTML and JavaScript.
- Object-oriented languages describe the solution to a problem by creating objects that represent real-world entities.
- Classes are templates for real-world objects which have attributes and methods.
- Objects are instances of classes and represent a real-world object.
- Methods are actions that classes or objects can perform, for example SpinWheel().
- Attributes are properties of a class or object such as radius or colour.
- Classes can inherit attributes and methods from other classes. For example, a Car class might inherit the attributes and methods from a Vehicle class. In this example, Car would be a derived class and Vehicle a superclass.
- Backus Naur form (BNF) is a way of representing syntax as a context-free grammar. BNF contains a number of key elements:
    - The rule written as <rulename> :: -
    - Concatenation: elements of the BNF are written next to each other
    - <IF> ::- IF <expression> THEN <statement>
    - Disjunction: elements are separated by the | symbol and effectively offer a choice of options
    - <digit> ::- 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
    - Recursion: allows rules to be iterative
        - <digits> ::- <digit> | <digit> <digits>

5. Systems analysis

Approaches	Describe different appropriate approaches to analysis and design, including Waterfall and Agile.
Feasibility	Describe the purpose of a feasibility study and describe the processes that an analyst would carry out during a feasibility study.
	Explain that proposed solutions must be cost effective, developed to an agreed time scale and within an agreed budget.
Investigation	Describe the different methods of investigation.
Analysis	Analyse a problem using appropriate techniques of abstraction and decomposition.
	Represent and interpret systems in an appropriate diagrammatic form showing the flow of data and the information processing requirements.
	Describe the selection of suitable software and hardware to address the requirements of a problem.
Changeover	Describe the various methods of changeover: direct, pilot, phased and parallel, identify the most suitable method for a given situation and its relative merits.
Program testing	Describe the use of alpha, beta and acceptance testing.
Maintenance	Describe the nature and use made of perfective, adaptive and corrective maintenance.
Backup and recovery	Describe different procedures for backing up data.
	Explain how data might be recovered if lost.
Documentation	Explain at which stage of the development each piece of documentation would be produced.
	Describe the contents and use made of user documentation and maintenance documentation.
	Describe the components of maintenance documentation, including annotated listings, variable lists, algorithms and data dictionaries.

Summary

- A stakeholder is anyone who has a vested interest in the development of a system. Common stakeholders include:
    - user
    - programmer
    - project manager
    - system analyst.

- Analysis of a computing system will include:
- Problem definition: the current problems with the system will be investigated and initially agreed upon.
- Feasibility: an investigation to ensure that the project is feasible by checking the following factors:
- Economic: Is there enough money to pay for the system?
- Time: Can the project be done in the timescales allowed?
- Technical: Is the technology and expertise needed to produce the system available?
- Legal: Will the project comply with the laws of the countries it will be released in?
- Fact finding: gather information needed to accurately define the new system to be built, using the following methods:
    - observation
    - questionnaire
    - interview
    - collecting documentation.

- Requirements specification will be produced at the end of analysis.
- Design is based on the requirements specification and is broken down into input, output, processing and storage design.
- Testing is where the system is put through its paces to ensure that it has met the requirements and most of the bugs have been fixed.
- Part of the deliverables of a project include documentation:
- User documentation: a manual on using the system provided to the end user
- Technical documentation: information to allow the system to be maintained.

- When a project is implemented, one of the following methods is used:
    - direct
    - phased
    - pilot
    - parallel.

- Once the system has been deployed, it must be maintained through one of the following methods:
    - corrective
    - adaptive
    - perfective.

- When developing systems, different methodologies will be used to implement the systems lifecycle:
- Program development techniques cover the various approaches to the design, writing and testing of programs. Some of the more common ones are:
- Incremental development is where the entire system is broken down into modules which are designed, implemented and tested before being added to the final product.
- Iterative development will provide a full system upfront but then build upon the functionality as time goes on.
- Prototyping is where only a small number of features are implemented combining both incremental and iterative approaches.
- Systems lifecycle methodologies are the different ways in which stages of the systems lifecycle can be combined and implemented:
- The Waterfall model is a linear approach to the systems lifecycle where each phase is completed fully before moving on to the next.
- Agile development is focused on producing software rather than documentation and places value on communication and collaboration. Team meetings are favoured over documentation.

6. System Design

Human-computer interaction	Discuss contemporary approaches to the problem of communication with computers. Describe the potential for a natural language interface. Describe the problems of ambiguity that can be associated with input that is spoken.
Design validation	Explain the need for a design review to: check the correspondence between a design and its specification; confirm that the most appropriate techniques have been used; confirm that the user interface is appropriate.
Design evaluation	Describe criteria for the evaluation of computer based solutions.

Summary

- Common interface devices, including keyboards and mice, are still in widespread use but are starting to be superseded by other human-computer interaction methods.
- Voice input makes use of spoken language to allow interaction with a computer. It has a number of draw backs:
- Commands have to be phrased in a specific way.
- Current software is not always able to judge the context or have the knowledge to answer the user.
- Accents or back ground noise can impact understanding.
- Slang will not always be interpreted.
- Touch screens are a feature of most new devices and can take advantage of gestures.
- Force feedback offers haptic feedback to the user which is adding a new dimension to human-computer interactions
- Virtual reality uses headsets to immerse the end user in an alternate world where they can interact by moving their bodies. Sensors track a user's movement and translate it into changes in the world the user is seeing.
- Augmented reality overlays details onto a live camera feed or projects onto glass. It will make use of sensors and other data to give the user a more immersive experience.
- New system design must be validated before being used to ensure that its implementation will match the requirements.
- Prototypes can be used to test out an interface design.
- Feedback in initial stages of a new design can be used to improve it before implementation.
- Evaluation allows the development company and the customer to decide whether a project has been successful. The key issues that need to be part of any evaluation are:
- Ensuring the requirements are met.
- Ensuring the system responds in an acceptable timeframe.
- Ensuring the product is robust.
- Review if the product was developed on cost and on time.
- Assess how useable the system is by the end user.

7. Software engineering

Software tools	Describe the types of software tool that have been designed to assist the software engineering process. Explain the role of appropriate software packages in systems analysis, systems specification, systems design and testing. Explain the role of Integrated Development Environment (IDE) tools in developing and debugging programs. Explain program version management.

Summary

- Software development can be supported by using CASE tools; Computer-aided software engineering.
- Analysis and planning tools enable the developer to create diagrams quickly, such as UML, validate requirements and collaborate on every part of analysis.
- Design tools allow mock ups of software to be produced for verification purposes.
- Test plans and bug repositories have CASE tools, which help large-scale projects keep track of which tests were run and what bugs were produced.
- Integrated development environments (IDEs) offer additional tools to the developer such as debugging, coding frameworks and the ability to manage larger code projects.
- Version management will monitor changes to code and will manage how code files are merged back into the master branch to avoid conflicts.

8. Program construction

Compilers, interpreters and assemblers	Describe the function of translation programs in making source programs executable by the computer. Describe the purpose and give examples of the use of compilers, interpreters and assemblers, and distinguish between them. Describe the principal stages involved in the compilation process: lexical analysis, symbol table construction, syntax analysis, semantic analysis, code generation and optimisation. Distinguish between and give examples of translation and execution errors.

Summary

- Translators take source code and turn it into machine code. There are three main types: compilers, assemblers and interpreters.
- Interpreters will read, translate and execute the code line by line.
- Intermediate code is produced by a compiler but requires an interpreter to run. It allows code to be both compiled and interpreted to provide the benefits of both.
- Machine code instructions are made up of an opcode and operand.
- Assembly uses mnemonics where each one represents a single machine code instruction.
- Assemblers will take a two-phase approach:
- phase one – turn symbols into memory addresses and store this in a symbol table
- phase two – convert assembly mnemonics into machine code.
- Compilers perform the following phases:
- lexical analysis – removes comments and white space before producing a token stream
- syntax analysis – checks the order of expressions using the token stream
- machine code generation – matches parts of the syntax to machine code; it will optimise the generated machine code
- linking – links the generated machine code to library software; it will add loaders to the code to enable dynamic library linking.

9. Economic, moral, legal, ethical and cultural issues relating to computer science

	Describe social and economic changes occurring as a result of developments in computing and computer use, and their moral, ethical, legal, cultural and other consequences.
Professional behaviour	Describe the role of codes of conduct in promoting professional behaviour.
Effect on employment	Describe the possible effects of computers on the nature of employment in the computing industry and wider society.
Legislation	Explain how current legislation impacts on security, privacy, data protection and freedom of information.

Summary

- The Data Protection Act sets out how those who collect personal and private data may use it. It also covers where this information can be sent and how long it can be kept for.
- The Computer Misuse Act makes it illegal to access a computer without consent as well as establishing the penalties for doing so.
- The Copyright Design and Patents Act sets out who has the right to make money from a particular creative work and how long those rights last.
- The Regulation of Investigatory Powers Act sets out the rules that govern how the UK government can collect information on its citizens.
- Computers have had an enormous impact on almost every work place imaginable. Some see this impact as a negative one costing jobs and reducing wages. Others see it as a very positive one that has created whole new industries.
- Advances in artificial intelligence have given rise to the prospect of automated decision making. Many are happy that impartial machines will make decisions formerly made by fallible humans; others are dismayed by the idea that a computer could make life-or-death decisions.
- Computers have made it easier to publicise the environmental impact of the actions of industries and individuals. But computers themselves require huge amounts of electricity and rare minerals to create and maintain.
- The internet has provided a wonderful forum for debate and free exchange of ideas. However, some governments view this freedom of expression as a threat and seek to censor it.
- Computers allow us not only to hold vast amounts of personal information, but also to analyse it to spot trends and patterns. For some this is useful, for others it is a serious invasion of privacy.
- Piracy and offensive communications began long before the invention of the computer but the advent of the internet means that it is quicker and easier than ever before to distribute offensive and illegal materials.
- The internet has made the world a much smaller place. This presents a unique set of challenges to those who are trying to communicate with everyone in this vast global marketplace.

Component 2

1. Hardware and communication

	Identify and describe the hardware and communication elements of contemporary computer systems and how they are connected.
Architecture	Identify and describe the main components of computer architecture, including Von Neumann and contemporary architectures.
	Describe different types of memory and caching.
	Describe and explain parallel processing and the limiting factors to parallelisation.
	Calculate the runtime of given tasks as a result of parallelisation and evaluate the effect of parallelisation.
Fetch-execute cycle	Describe the fetch-execute cycle, including how data can be read from RAM into registers.
Assembly language programming	Write simple programs in assembly language and demonstrate how these programs could be executed.
Input / output	Describe the use of contemporary methods and their associated devices for input and output.
	Explain the use of these methods and devices in contemporary computer systems and their suitability in different situations.
	Describe and differentiate between voice input for command and control systems to operate a computer system, vocabulary dictation systems for general input and voice print recognition for security. Discuss the suitability of each system in different situations.
Secondary storage	Compare the functional characteristics of contemporary secondary storage devices.
Data storage on disc	Explain fragmentation and its consequences and describe the need for defragmentation.
Networking	Describe networks and how they communicate.
	Explain the importance of networking standards.
	Describe the importance and the use of a range of contemporary protocols including HTTP, FTP, SMTP, TCP/IP, IMAP, DHCP, UDP and wireless communication protocols.
	Explain the role of handshaking.
	Identify and describe applications where connecting a portable device to a network is required.
	Describe the hardware required to make a wireless connection and explain how this might be achieved using contemporary wireless technologies.

Summary

- A CPU is made up of the following components:
    - ALU – arithmetic logic unit
    - MU – memory unit
    - CU – control unit.
- Machine code instructions are written in binary and are made up of an opcode and data.
- Little Man Computer a is simplified architecture used to help learners understand the basics of low-level programming.
- The fetch–decode–execute (FDE) cycle will fetch instructions from memory to be run on the CPU. During the cycle the following registers are used:
    - MAR – memory address register, which stores the address of the instruction to fetch
    - MDR – memory data register, which stores the data or instruction once fetched
    - PC – program counter, which stores the address of the next instruction to fetch
    - CIR – current instruction register, from where the instruction will be decoded and executed.
- Cache is fast memory that sits between memory and the CPU. It is used to help remove the impact of the von Neumann bottleneck.
- Processors, to improve the throughput of instructions, perform parts of the FDE cycle concurrently, which is known as pipelining.
- A multi-core system is a common architecture where processors have more than one core allowing multiple processes to be run concurrently.
- Parallel systems are made up of multiple processors in order to perform multiple operations concurrently. Code must be written in such a way as to take advantage of any parallelism.
- Random-access memory (RAM) stores the currently running programs, operating systems and user files.
- Read-only memory (ROM) stores the boot-up software required to initialise the hardware, load settings and initiate the operating system (OS).
- Input devices provide data for processing and come in the form of scanners, optical character recognition, optical mark recognition, barcodes, magnetic ink recognition, touch screens and sensors.
- Storage devices store data and programs when the computer is powered down. The main types of storage device are:
    - magnetic – back-up tape and hard drives
    - optical – CD, DVD and Blu-ray
    - solid state – solid state hard drives, memory sticks and secure digital (SD) cards.
- Redundant array of independent disks (RAID) arrays are used in servers and for more data-critical systems. They allow multiple drives to be connected together to form a single drive.
- RAID arrays can provide fault tolerance in the form of mirroring and parity bits.
- Storage area networks (SAN) can connect numerous different types of storage device as a single device.
- Virtual storage will abstract the type of storage device in a SAN, allowing devices to be added and removed without having to administer the individual devices.
- Networks rely on the use of protocols and standards to ensure that communication between devices can occur even if those devices were produced by different companies.

- Client–server-based architecture is where a powerful computer (known as a server) provides functionality to the rest of the network (a service). This can include providing files, websites or databases.
- Peer-to-peer networks have no server; rather each computer has the same status as the other. These networks are commonly used for file sharing.
- Error detection ensures that packets arrive at their destination without corruption. Common error detection methods include:
    - parity bits: adding additional bits to make the number of ones odd or even, depending on the type of parity chosen
    - echo: packets are returned from the destination and then compared by the sender
    - checksum: calculations are performed on the packet generating a checksum that is sent with the packet; this calculation is repeated and the checksums compared.

- Protocols are a set of rules that govern communication.
- Handshaking is a process that devices undertake when a connection is set up. This is where the set of protocols to be used is agreed.
- Protocol stacks allow the separation of logic for each protocol, enabling different parts of the stack to be swapped.
- Transmission control protocol/internet protocol (TCP/IP) stack is the most common protocol stack used in networking.
- Hypertext transfer protocol (HTTP) is used for the transmission of webpages and webpage components. HTTP uses port 80.
- HTTPS (secured) is a secure version of HTTP and communicates over port 443. Certificates and the secure sockets layer (SSL) protocol are used to secure communication.
- File transfer protocol (FTP) uses port 21 and allows files to be transmitted.
- The transport layer of TCP/IP uses either TCP or UDP:
    - TCP provides a streamlike communication with error detection, ordering of packets and confirmation that packets have been received.
    - UDP (user datagram protocol) is a fire-and-forget approach with minimal protections.

- IP is a network layer protocol that provides routing. It uses the end-to-end principle where each node is considered to be unreliable and therefore each node is asked to perform routing.
- Domain name service (DNS) will translate uniform resource locators (URL) into IP addresses.
- Domain name resolution is done through a hierarchical set of servers. Requests start at the bottom of the hierarchy and work their way up until a server knows about the domain being looked up.
- Common networking hardware includes:
    - network interface card: allows a computer to connect to the network
    - hub: connects multiple computers and packets are broadcast to all connected devices
    - switch: allows multiple devices to be connected and provides routing by media access control (MAC) address
    - router: connects different networks together and uses IP addresses for routing; commonly used to connect to the internet.

2. Data transmission

	Describe serial and parallel transmission, their advantages and disadvantages.
	Describe simplex, half duplex and full duplex transmission methods.
	Explain the need for multiplexing and switching.
Communication networks	Describe, using appropriate network protocols, such as TCP/IP the typical contents of a packet.
	Explain network collision, network collision detection and how these collisions are dealt with.
	Describe methods of routing traffic on a network.
	Calculate data transfer rates on a network.
	Calculate lowest cost routes on a network.
	Describe the internet in terms of a world-wide communications infrastructure.

Summary

- When data is transmitted over a network it is broken down into packets. Each packet will have a header and footer attached which is generated by the protocols employed to transmit that packet.
- Duplex mode signifies which direction packets can travel at any given time:
  - duplex (or full duplex): both ways at the same time
  - half duplex: both ways but not at the same time
  - simplex: one direction only.
- Bit rate is the measure of how much data can be transmitted in a second. Measured as Mbps, or Megabits per second.
- Packet switching is where each packet will take an independent route through a network.
- Circuit switching is where all packets will follow the same route.

3. Data representation and data types

Representation of data as bit patterns	Explain the terms bit, byte and word.
	Describe and use the binary number system and the hexadecimal notation as shorthand for binary number patterns.
Storage of Characters	Describe how characters and numbers are stored in binary form
	Describe standardised character sets.
Representation of numbers as bit patterns	Apply binary arithmetic techniques.
	Explain the representation of positive and negative integers in a fixed-length store using both two's complement, and sign and magnitude representation.
	Describe the nature and uses of floating point form.
	State the advantages and disadvantages of representing numbers in integer and floating point forms.
	Convert between real number and floating point form.
	Describe truncation and rounding, and explain their effect upon accuracy.
	Explain and use shift functions: logical and arithmetic shifts.Interpret and apply shifts in algorithms and programs.
	Describe the causes of overflow and underflow.

Summary

- File sizes are measured using magnitudes of bytes, from smallest to largest; byte, kilobyte, megabyte, gigabyte, terabyte and petabyte.
- Binary is base 2 and can be used to represent numbers; every column can be determined by the formula 2^n, where n is the column number starting at 0.
- Hexadecimal is base 16 and uses letters to represent values greater than 9.
- Characters are encoded by giving each of them a number. That number can then be converted into binary.
- ASCII uses 8 bits to store characters while Unicode uses 16 bits. Unicode can be used to represent symbols from other languages such as Japanese.
- The key data types are character, string, Boolean, integer and real.
- Two's complement and sign-magnitude are methods of storing negative numbers in binary. Both use the most significant bit to represent negative numbers. 1 always means negative.
- Binary addition uses *carries* if the result of the addition is greater than 1.
- Binary subtraction is easier if you convert the number to be subtracted into a negative two's complement number.
- Floating point numbers are made up of a mantissa and an exponent.
- Normalised floating point numbers will always start with either 01 or 10.
- Overflow is when a number is too large for the number of bits being used to represent it.
- Underflow is when a number is too small for the number of bits being used to represent it.

4. Organisation and structure of data

	Explain the purpose of files in data processing.
File design	Define a file in terms of records and fields.
	Describe how files may be created, organised, updated and processed by programs.
	Explain fixed and variable length fields and records and give examples of the appropriate use of each type.
	Design files and records appropriate for a particular application.
File organisation	Distinguish between master and transaction files.
	Describe sequential, indexed sequential and direct (random) file access.
	Distinguish between the use of serial and sequential file access methods in computer applications. Describe and design algorithms and programs for sequential file access and update.
	Explain the purpose of, and be able to use, a hashing algorithm.
	Compare different hashing algorithms.
	Explain the use of multi-level indexes.
	Explain the techniques used to manage overflow and the need for file re-organisation.
	Explain the need for file security, including file backup, generations of files and transaction logs.
	Describe the need for archiving files.

Summary

- Fixed length records are made up of a number of fields, each one having a specific data type and size.
- Common data types include integer, float, string, date and Boolean.
- Estimating file size requires the size of single records to be calculated, multiplied by the number of records and 10% added on for metadata.
- Files are opened in one of three modes: append, read and write.
- Files are read by iterating over records using end of file, EOF, to end the loop.
- Serial files store records in chronological order.
- Sequential files order records by a primary key field.
- Indexed sequential files will have an index which will point to groups of related records.
- Inserting and deleting from sequential and index sequential files require a new file to be created and the file to be re-organised.
- Multi-level indexes can speed up access to groups of records in larger files.
- Random access files are broken into blocks of records.
- Records are added to blocks by using a hash function.
- Hash functions take a key and calculate a block number. Their key design goal is to ensure a wide spread of blocks for any given set of input.
- By looking at the number of collisions, you can compare hash functions.
- Master files store all data while transaction files store data currently being worked on.
- Transaction files are used to speed up access for day-to-day processing of data.
- Backing up files will be based on a policy set up by a company. It will consider how often the back-up will occur, when it should be taken, how long it will be kept and where it will be stored.
- Archiving is the process of moving old files to a separate system in order to speed up the main one.

5. Databases and distributed systems

	Explain what is meant by data consistency, data redundancy and data independence.
	Describe and discuss the benefits and drawbacks of relational database systems and other contemporary database systems.
	Explain what is meant by relational database organisation and data normalisation (first, second and third normal forms).
	Restructure data into third normal form.
	Explain and apply entity relationship modelling and use it to analyse simple problems.
	Describe the use of primary keys, foreign keys, and indexes.
	Describe the advantages of different users having different views of the data in a database.
	Explain how the data can be manipulated to provide the user with useful information.
Data validation and verification	Explain and apply appropriate techniques for data validation and verification of data in databases.
Searching data	Explain the purpose of query languages.
	Construct and run queries using Structured Query Language (SQL).
Database management systems	Explain the purpose of a database management system (DBMS) and data dictionaries.
Big Data	Explain what is meant by Big Data, predictive analytics, data warehousing and data mining.
Distributed systems	Explain that distribution can apply to both data and processing.
	Describe distributed databases and the advantages of such distribution.

Summary

- Flat file databases store information in a single large table.
- Relational databases store data in many smaller tables, linked together using keys.
- Primary keys are unique attributes that can be used to identify a record in a table.
- Foreign keys are the primary keys from another table and are used to link tables together.
- Entity relationship models visually describe the relationships between the different entities in a database.
- Normalisation is the process of converting a flat file database into a relational one.
- An index is a data structure used to shorten the length of time it takes to search a database.
- There are different stages of normalisation. The most common is third normal form (3NF). A table is in third normal form if it contains no transitive dependencies.
- Structured query language (SQL) is used to create, edit and delete data in databases.
- Referential integrity is needed to ensure that the foreign keys in one table refer to a primary key in another.
- Data mining and predictive analytics allow analysis of big data to uncover patterns and predict future outcomes.
- Database views allow a restricted view of the entire database for performance and security reasons.
- Database management systems allow users and databases to be managed. They also implement the data dictionary which stores all of the metadata of the database.
- Distributed databases split an organisation’s data need between multiple database servers.

6. The operating system

Managing resources	Describe the need for and the role of the operating system kernel in managing resources, including peripherals, processes, memory protection and backing store.
Providing an interface	Describe the need for and the role of the operating system in providing an interface between the user and the hardware.
Managing backing store	Explain the hierarchical structure of a directory and describe file attributes.
Utility software	Explain the need for and use of a range of utility software.
Modes of operation	Describe the main features of batch processing, real time control and real time transaction systems.
	Identify and describe applications that would be suitable to these modes of operation.
Types of operating system	Explain the following types of system: batch, single-user (standalone), multi-user (multi-access), multi-tasking and multi-programming.
Consideration of human-computer interaction	Explain the need to design systems that are appropriate to the variety of different users at all levels and in different environments.
Interrupts	Describe a range of conditions or events which could generate interrupts.
	Describe interrupt handling and the use of priorities.
	Describe the factors involved in allocating differing priorities.
Memory management and buffering	Explain the reasons for, and possible consequences of, partitioning of main memory.
	Describe methods of data transfer including the use of buffers to allow for differences in speed of devices.
	Describe buffering and explain why double buffering is used.
Scheduling	Describe the principles of high level scheduling: processor allocation, allocation of devices and the significance of job priorities.
	Explain the three basic states of a process: running, ready and blocked.
	Explain the role of time-slicing, polling and threading.

Summary

- System software manages the hardware and software of the computer.
- The operating system is responsible for memory management, process management and managing devices.
- Memory management is done through two mechanisms: paging and segmentation.
- Pages are blocks of memory of fixed sized that are referenced through a paging table. The paging table contains the page number and offset to physical memory.
- Each process has a virtual address space which is translated by the memory unit to physical addresses by the formula
- address = page number * page size + offset
- Segmentation allows blocks of memory to be used, which are variable in size.
- Each segment is stored in a segmentation table and will record the ID, start address and length for each segment.
- Segments are commonly used to store library software or other blocks of data that are shared across multiple processes.
- Virtual memory is where the hard drive is used as additional memory in order to increase the number of processes that can be active at once.
- When a process is saved into virtual memory it is suspended. When loaded back into RAM for processing it is restored.
- Page tables will record which processes are located in RAM or in virtual memory.
- An interrupt is an electronic signal (hardware interrupt) or a process-generated signal (software interrupt) that is sent to the CPU to inform it that a device or process requires attention.
- Common interrupts include storage devices, timers, peripherals, power, and software and hardware failure.
- Buffers are used on devices to allow the CPU to perform other tasks while data is being saved.
- Interrupt service routines (ISR) are bits of code that are run when an interrupt occurs; these are normally within a device driver.
- A process can be in one of the following states:
  - running – currently has CPU time
  - blocked – waiting for an operation to complete
  - ready to run – waiting for the CPU.
- Processes are swapped by storing both special and general registers into a process control block. This can then be used to restore the process.
- The main design goal of scheduling is to maximise the throughput of processes. To do this it must minimise starvation and eliminate deadlock.
- Threading is the method of splitting a process into multiple smaller processes to take advantage of multiple cores.
- There are a number of different types of operating system:
  - single-user
  - multi-user
  - real time.

7. The need for different types of software systems and their attributes

Types of software	Explain the use of a range of types of software, including open source software, bespoke and off-the-shelf.
Safety related systems	Explain that some computer applications are safety related and require a high level of dependability, and hence that the development of safety critical systems is a highly specialised field.
Industrial, technical and scientific	Describe the role of the computer in weather forecasting, computer aided design, robotics and the use of computer generated graphics and animation.
Control systems	State the nature and scope of computer control and automation.
	Describe the benefits and implications of automation.
Expert systems	Explain the purpose, use and significance of expert systems.
	Discuss the possible effects of expert systems on professional groups and the wider community.
Internet and Intranet	Describe the use of search engines on the internet.
	Describe common contemporary applications.
	Discuss the possible effects of the internet upon professional groups and the wider community.

Summary

- Open source software tends to be free and will always give access to the source code for people to read, edit and share.
- Closed source software will not give access to the source code, but may be free to use.
- Proprietary software is also referred to as closed source.
- Off-the-shelf software can be bought immediately but may not be able to offer the same level of functionality that custom built software can.
- Safety critical systems are classified as any system which could lead to death, injury or loss to company assets.
- Safety critical systems must have fail-safes and fault tolerance and are much harder to produce than regular software.
- Knowledge based systems, also known as expert systems, are broken into four parts; knowledge base, rule base, inference engine and GUI.
- Utilities are software which help the user accomplish tasks and tend to come shipped as part of the operating system.
- The modern trend in software is moving towards a higher number of smaller specialised apps rather than larger monolithic applications.
- Applications such as 3D ray tracers can produce detailed computer-generated images and animation.
- Weather forecasting is based on mathematical models and is backed by vast amounts of data.
- Computer automation makes use of sensors to record the physical environmental and actuators to control it.
- Computer-aided design allows designers to try ideas out on the computer, including modelling how the product will react in different scenarios and simulations, without having to build expensive prototypes.
- Computer-aided manufacture will take designs and produce exactly the same product using an assembly line.
- Search engines give greater access to a wider range of information.
- The PageRank algorithm will determine where a website appears in search results.
- A company’s reputation must be carefully managed as negative information can have a negative impact on sales.

8. Data security and integrity processes

Protecting data integrity	Explain the special security and integrity problems which can arise during online updating of files.
Privacy and security	Describe the dangers that can arise from the use of computers to manage files of personal data.
	Describe contemporary processes that protect the security and integrity of data including standard clerical procedures, levels of permitted access, passwords for access and write-protect mechanisms.
Cryptography	Describe the need for and the purpose of cryptography.
	Describe techniques of cryptography and their role in protecting data.
	Follow algorithms and programs used in cryptography.
	Compare cryptographic methods and their relative strength.
Biometrics	Describe the purpose and use of contemporary biometric technologies.
	Describe the benefits and drawbacks of biometric technologies.
	Describe the complexities of capturing, storing and processing biometric data.
Disaster planning	Describe the various potential threats to computer systems.
	Describe contingency planning to recover from disasters.
Malicious and accidental damage	Describe malicious and accidental damage to data and identify situations where either could occur.
Malicious software and mechanisms of attack and defence	Describe types and mechanisms of malicious software and their vectors.
	Describe black hat hacking, white hat hacking and penetration testing

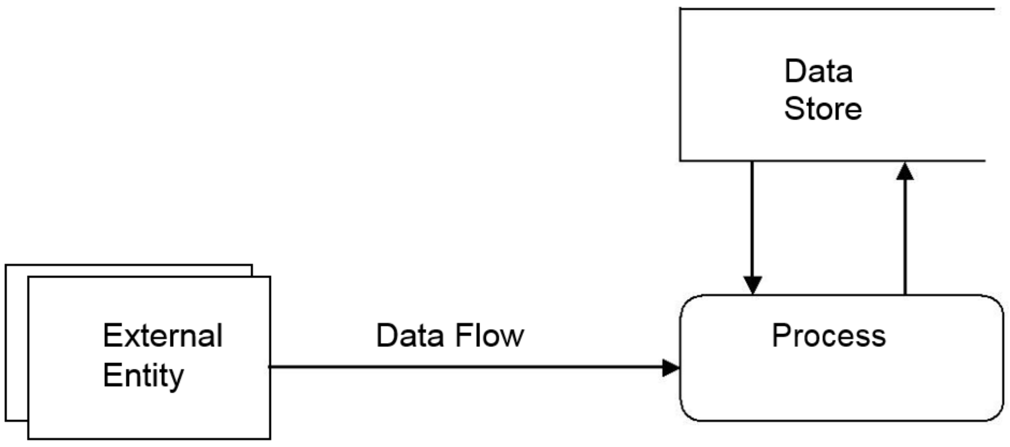
Summary

- Security risks to modern systems include outside access of files, corruption of data, unauthorised reading or duplication and loss or deliberate deleting.
- Policies ensure that staff use data responsibly and minimise risk of data loss.
- Passwords need to be strong in order to protect data and files.
- Access levels restrict which users can access sensitive files and data.
- In symmetric encryption the process of decryption is the opposite of the process used to encrypt. The Caesar cipher is an example of symmetric encryption.
- In asymmetric encryption algorithms the encryption and decryption keys are different.
- Cryptographic methods can be compared by looking at how well they diffuse information and how hard they are to brute-force.
- Biometric data uses human metrics to identify individuals.
- Common biometric methods include facial recognition, finger prints, iris scans, DNA and palm prints.
- Biometric data is unique to individuals which makes it very difficult to impersonate.
- Biometric data can be inaccurate if recorded in sub-optimal conditions.
- Privacy concerns arise as a result of using biometric data.
- Black hat hackers will break into systems for their own ends while white hat hackers use their skills to identify and fix security flaws.
- Malicious software comes in many forms including viruses, Trojans, spyware, ransom-ware and botnets.
- Contingency planning allows companies to develop disaster recovery procedures. The plans must be tested and constantly updated to ensure the business has limited exposure to down time.

Diagrams and Exam Conventions

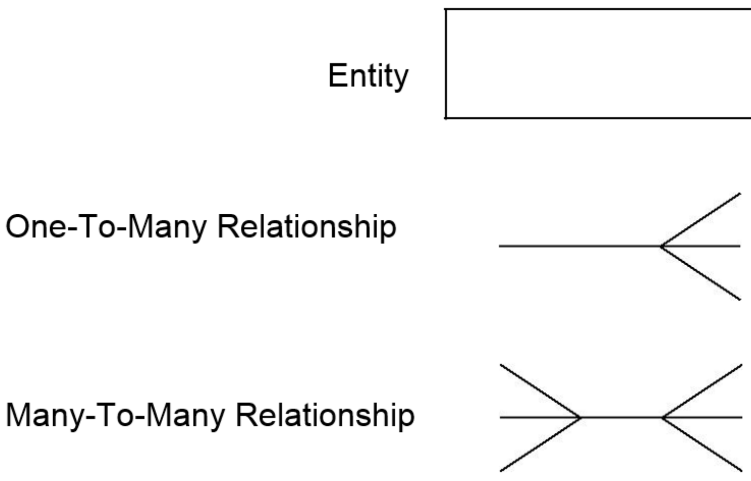
DFDs

In the case of data flow diagrams, where no generally accepted symbols currently exist, candidates should be familiar with the following symbols, used in a number of current GCE textbooks:



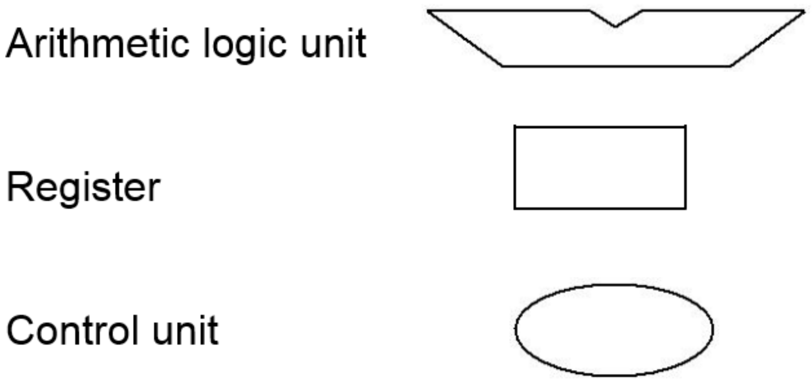
ERDs

The following symbols are used for entities and relationships.



Von Neumann

Where Von Neumann architecture is represented diagrammatically, the following symbols are used.



Structured Query Language.

Candidates should be familiar with the following commands and operators:

- CREATE TABLE …

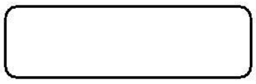
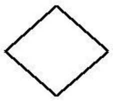
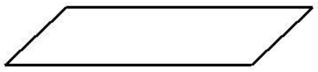
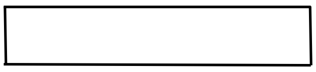
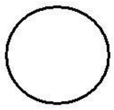
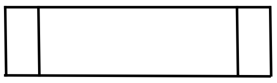
- PRIMARY KEY
- NOT NULL
- Int
- Char(n)
- Numeric(m,n)
- DateTime
- INSERT INTO … VALUES

- SELECT … FROM … WHERE …
- SELECT * FROM … WHERE …
- IN
- AND
- OR
- ORDER BY
- GROUP BY
- UPDATE … SET …
- =
- >
- >=
- <
- <=
- <>

Candidates should also be familiar with the use of sub queries and parentheses.

Flowcharts

The following symbols are used in flowcharts:

Start / Stop procedure	
Decision box	
Input / Output	
Operation	
Connector	
Store / Subroutine call	
Flow of control