| No. | Spec | | Section | | Term | | Definition |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1.1.1a** | Structure and function of the processor | | ALU | | **Arithmetic Logic Unit:** “The part of the CPU where data is processed and manipulated. This processing and manipulation normally consists of arithmetic operations or logical comparisons allowing a program to make decisions.” | |
|  | **1.1.1a** | Structure and function of the processor | | Control Unit | | “The part of the CPU that manages the execution of instructions. The control unit fetches each instruction in sequence, and decodes and synchronises it before executing it by sending control signals to other parts of the computer.” | |
|  | **1.1.1a** | Structure and function of the processor | | Register | | “Tiny areas of extremely fast memory located in the CPU normally designed for a specific purpose, where data or control information is stored temporarily.” | |
|  | **1.1.1a** | Structure and function of the processor | | PC | | **Program Counter**: “A register in the control unit which holds the address of the next instruction to be executed.” | |
|  | **1.1.1a** | Structure and function of the processor | | ACC | | **Accumulator:** “A special register within the ALU. It is used to hold the data currently being processed by the central processor. Any data to be processed is stored temporarily in the accumulator, the results ending up back in the accumulator being stored in the memory unit.” | |
|  | **1.1.1a** | Structure and function of the processor | | MAR | | **Memory Address Register:** “A register in the CPU that stores the address of the memory location currently in use. In the fetch phase, this would be the address of the instruction being loaded; in the execute phase, it would be the address of the data being used.” | |
|  | **1.1.1a** | Structure and function of the processor | | MDR | | **Memory Data Register:** “A register in the CPU that stores data being transferred to and from the immediate-access store. It acts as a buffer, allowing the central processor and memory unit to act independently without being affected by minor differences in operation. A data item will be copied to the MDR ready for use at the next clock pulse, when it can either be used by the central processor or be stored in main memory.” | |
|  | **1.1.1a** | Structure and function of the processor | | CIR | | **Current Instruction Register:** “A register in the control unit that stores the address of the next instruction currently being executed and decoded.” | |
|  | **1.1.1a** | Structure and function of the processor | | Busses | | “A common physical pathway shared by signals to and from several components of a computer.” | |
|  | **1.1.1a** | Structure and function of the processor | | Data Bus | | “The part of the bus which carries the actual information.” | |
|  | **1.1.1a** | Structure and function of the processor | | Address Bus | | “The part of the bus which carries identification about where the data is being sent.” | |
|  | **1.1.1a** | Structure and function of the processor | | Control Bus | | “This bus carries command and control signals to and from every other component of a computer.” | |
|  | **1.1.1b** | Structure and function of the processor | | Fetch-Decode-Execute | | “The complete process of retrieving an instruction from store, decoding it and carrying it out. Also known as the instruction cycle.” | |
|  | **1.1.1c** | Structure and function of the processor | | CPU | | **Central Processing Unit:** “The main part of the computer, consisting of the registers, ALU and control unit.” | |
|  | **1.1.1c** | Structure and function of the processor | | Clock Speed | | “Measured in Hertz, the clock speed is the frequency at which the internal clock generates pulses. The higher the clock rate, the faster the computer may work. The “clock” is the electronic unit that synchronises related components by generating pulses at a constant rate.” | |
|  | **1.1.1c** | Structure and function of the processor | | Cores | | “A part of a multi-core processor. A multi-core processor is a single component with two or more independent actual CPUs, which are the units responsibly for the fetch-decode-execute cycle.” | |
|  | **1.1.1c** | Structure and function of the processor | | Cache | | “A part of the main store between the central processor and the rest of the memory. It has extremely fast access, so sections of a program and its associated data are copied there to take advantage of its short fetch cycle.” | |
|  | **1.1.1d** | Structure and function of the processor | | Pipelining | | “The concurrent decoding of two or more machine instructions. While part of one instruction (for example, an address field) is being decoded, another part of a second instructions (for example, an operation code) may also be decoded.” | |
|  | **1.1.1e** | Structure and function of the processor | | Von Neumann Architecture | | “Traditional computer architecture that forms the basis of most digital computer systems. A single control unit manages program control flow following a linear sequence of “fetch-decode-execute”” | |
|  | **1.1.1e** | Structure and function of the processor | | Harvard Architecture | | “A computer architecture with physically separate storage and signal pathways for instructions and data. These early machines had data storage entirely contained within the central processing unit, and provided no access to the instruction storage as data.” | |
|  | **1.1.1e** | Structure and function of the processor | | Contemporary Architecture | | “Any modern set of disciplines that describes the functionality, the organisation and the implementation of computer systems.” | |
|  | **1.1.2a** | Types of processor | | CISC | | **Complex Instruction Set Computer: “**A design that produces a complicated and expensive integrated circuit capable of performing a large variety of complex instructions. Complex instructions can be executed with few machine cycles.” | |
|  | **1.1.2a** | Types of processor | | RISC | | **Reduced Instruction Set Computer:** “A design that produces a simple, cheap integrated circuit with a basic range of machine instructions. Relies on speed as complex instructions take many machine cycles.” | |
|  | **1.1.2b** | Types of processor | | GPU | | **Graphical Processing Unit:** “A specialised electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. Modern GPUs highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel.” | |
|  | **1.1.2c** | Types of processor | | Multicore System | | “Development of CISC architecture with several sets of CPU components in one microprocessor.” | |
|  | **1.1.2c** | Types of processor | | Parallel Processor System | | “The simultaneous use of several processors to perform a single job. A job may be split into a number of tasks each of which may be processed by any available processor.” | |
|  | **1.1.3a** | Input, output and storage | | Input Device | | “Any peripheral device that can accept data, presented in the appropriate machine-readable form, decode it and transmit it as electrical pulses to the CPU” | |
|  | **1.1.3a** | Input, output and storage | | Output Device | | “Any peripheral device that translates signals from the computer into a human-readable form or into a form suitable for reprocessing by the computer at a later stage.” | |
|  | **1.1.3a** | Input, output and storage | | Storage Device | | “Any medium (optical, magnetic, solid state and even paper which holds data or programs” | |
|  | **1.1.3b** | Input, output and storage | | Magnetic Storage | | “Storage medium which uses surfaces coated with a layer of magnetic material on which data can be stored by magnetically setting the arrangement of the magnetic material. This is done by electromagnetic read/write heads.” | |
|  | **1.1.3b** | Input, output and storage | | Flash Storage | | “Collection of memory chips that is controlled by its own software to make the collection of chips act like a disk drive.” | |
|  | **1.1.3b** | Input, output and storage | | Optical Storage | | “Storage medium that uses plastic discs on which the data is stored as patterns on the surface in pits and lans.” | |
|  | **1.1.3c** | Input, output and storage | | RAM | | **Random Access Memory:** “Volatile main memory. Access times are very fast. Often referred to as Main Memory, although RAM can be used in main areas of computing and computing storage. When used as main memory RAM typically can be thought of as containing the Operating System, programs in use and the data those programs are using while the computer is running.” | |
|  | **1.1.3c** | Input, output and storage | | ROM | | **Read Only Memory:** “Memory for which the contents may be read by cannot be written to by the computer system. Software in ROM is fixed during manufacturing. It typically holds system boot up instructions. ROM is non-volatile. More modern types of ROM can be written to in limited ways, these include, PROM, EPROM and EAROM.” | |
|  | **1.1.3d** | Input, output and storage | | Virtual Storage | | “A means of apparently extending main storage by using backing storage (such as a hard disk) as if it were main memory.” | |
|  | **1.2.1a** | Operating systems | | Operating System | | “Collection of systems software that manages the computer. Usually supplied with the computer. Most common operating systems are Windows, Linux, Unix, MacOS, iOS.” | |
|  | **1.2.1b** | Operating systems | | Memory Management | | “The process of organising the flexible use of the computer’s main memory.” | |
|  | **1.2.1b** | Operating systems | | Paging | | “The organisation of memory into fixed-sized units, called pages. The immediate-access store is organised as a number of physical pages. The logical pages used by the SPU can be assigned by the memory management unit to any page in physical memory.” | |
|  | **1.2.1b** | Operating systems | | Segmentation | | “The splitting of a large program unto a number of smaller programs or segments. Each segment is a complete program that is executed separately. The function of the large program is achieved by running segments consecutively. Segmentation allows a large program to be executed on a computer with insufficient memory to store the whole program by carving up memory logically rather than physically.” | |
|  | **1.2.1b** | Operating systems | | Virtual Memory | | “Used when sufficient immediate-access store is not available. Part of a disk drive is allocated to be used as if it were main memory. This is very slow, and the software will attempt to use the immediate-access store if possible.” | |
|  | **1.2.1c** | Operating systems | | Interrupt | | “A signal generated by a source such as an input or output device or a systems software routine that causes a break in the execution of the current routine. Control passes to another routine in such a way that the original routine can be resumed after the interrupt.” | |
|  | **1.2.1c** | Operating systems | | ISR | | **Interrupt Service Routine:** “A software routine that hardware invokes in response to an interrupt. ISRs examine an interrupt and determine how to handle it.” | |
|  | **1.2.1d** | Operating systems | | Scheduling | | “The method by which central processor time is allocated in a multi-access system.” | |
|  | **1.2.1d** | Operating systems | | RR | | **Round Robin:** “A scheduling algorithm that deals with each user or task to be processed in turn.” | |
|  | **1.2.1d** | Operating systems | | FCFS | | **First Come First Serve:** “A scheduling algorithm that deals with each user or task in the order in which they arrive, this can be thought of as a queue.” | |
|  | **1.2.1d** | Operating systems | | MLFQ | | **Multi-Level Feedback Queue:** “A complex scheduling algorithm that deals with tasks based on a set of priorities and rules across different league tables, jobs in a certain table get promoted up or down their table based on these rules and can then end up in totally different tables if they relegated.” | |
|  | **1.2.1d** | Operating systems | | SJF | | **Shortest Job First:** “A scheduling algorithm that deals with each user or task based on the getting the smaller ones out of the way.” | |
|  | **1.2.1d** | Operating systems | | SRT | | **Shortest Remaining Time:** “A scheduling algorithm that deals with each user or task based on a calculating an estimated time remaking to complete.” | |
|  | **1.2.1e** | Operating systems | | Distributed OS | | “An operating system where the software is spread over a collection of independent, networked, communicating, and physically separate nodes.” | |
|  | **1.2.1e** | Operating systems | | Embedded OS | | “A highly specialised, often quite limited and cut down operating system design to fit inside a certain type of machine. For example computers in cards, tragic lights, cash machines, POS, elevators, drinks machines etc. In contrast to an operating system for a general-purpose computer, an embedded OS is typically quite limited, often running a single application. However, that single application is crucial to the device’s operation.” | |
|  | **1.2.1e** | Operating systems | | Multi-tasking OS | | “Any type of system that is capable of running or appearing to run more than one program at a time.” | |
|  | **1.2.1e** | Operating systems | | Multi-user OS | | “Any operating system designed to allow multiple users to log in and connect to it from different locations / machines at the same time. Specially designed for the sharing of resources such as files, programs or hardware (printers, scanners etc).” | |
|  | **1.2.1e** | Operating systems | | Real Time OS | | “An operating system intended to serve real-time application process data as it comes in, typically without buffering delays.” | |
|  | **1.2.1f** | Operating systems | | BIOS | | **Basic Input / Output System:** “Part of the operating system that handles the input and output of the computer. It enables the operating system to use the particular features of the hardware being used.” | |
|  | **1.2.1g** | Operating systems | | Device Drivers | | “A computer program that operates or controls a particular type of device that is attached to a computer.” | |
|  | **1.2.1h** | Operating systems | | Virtual Machine | | “An emulator of a particular computer system. Virtual machines operated based on the computer architecture and functions of a real or hypothetical computer, and their implementations may involve specialised hardware, software, or a combination of both.” | |
|  | **1.2.1h** | Operating systems | | Intermediate Code | | “Half-way type of code which is standard across machine types. Runs on a virtual machine.” | |
|  | **1.2.2a** | Application generation | | Application | | “Any program, routine or procedures (together with their associated documentation) that can be run on a computer system.” | |
|  | **1.2.2b** | Application generation | | Utilities | | “A systems program that performs some specific task in the operation of the computer, for example file backup, virus checking or a compression program.” | |
|  | **1.2.2c** | Application generation | | Open Source | | “Software for which the original source code is made freely available and may be redistributed and modified. – Open source file types are often able to be opened on many different types of applications.” | |
|  | **1.2.2c** | Application generation | | Closed Source | | “Often called Proprietary is a type of computer program for which the software code is not shared with the public for anyone to look at or make changes to – Closed / Proprietary file types are often only able to be opened if you own a version of the software they were originally made in.” | |
|  | **1.2.2d** | Application generation | | Source Code | | “Original code typed in by the programmer in the native language. This is the code as it appears before it is compilers or interrupted.” | |
|  | **1.2.2d** | Application generation | | Translator | | “A program that translates a program written in assembly language into machine code.” | |
|  | **1.2.2d** | Application generation | | Interpreter | | “Translates and executes a program one statement at a time.” | |
|  | **1.2.2d** | Application generation | | Compiler | | “A program that translates a high-level language program, source code, into a computer’s machine code.” | |
|  | **1.2.2d** | Application generation | | Assembler | | “A program that translates (assembles) a program written in assembly language into machine code.” | |
|  | **1.2.2e** | Application generation | | Compilation | | “The translation process that produces an equivalent program in a low-level language. Compilation involves analysing the language structure of the source program, determining if it is valid, and proceeding suitable machine code.” | |
|  | **1.2.2e** | Application generation | | Lexical Analysis | | “The stage in compilation of a program that puts each statement into the form best suited to the syntax analyser. The standard components of each statement, such as PRINT, IF etc., are replaced by their tokens (a unique fixed length code) and programmer-defined names are entered into a symbol table. The lexical analyser also remove unnecessary charters such as spaces, tabs and coder comments.” | |
|  | **1.2.2e** | Application generation | | Syntax Analysis | | “The stage in compilation where language statements are checked against the rules of the language, errors being reported if a statement is not valid.” | |
|  | **1.2.2e** | Application generation | | Code Generation | | “The stage in compilation that produces a machine-code program that is equivalent to the source program.” | |
|  | **1.2.2e** | Application generation | | Optimisation | | “The stage of compilation that ensures that the executable program if optimised at least as much as the compiler can predict. e.g. removing reference to unused variables or routines.” | |
|  | **1.2.2f** | Application generation | | Linker | | “Software tool that allows already compiled object code files or modules to be combined with the compiled program.” | |
|  | **1.2.2f** | Application generation | | Loader | | “A program that copies an object program held on backing store into main store ready for execution.” | |
|  | **1.2.2f** | Application generation | | Libraries | | “A collection of pre-compiled routines which can be incorporated into a program.” | |
|  | **1.2.3a** | Software development | | SDLC | | **Systems Development Life Cycle:** “The various stages in designing and implementing a new computer system. These stages – typically *analysis, design, implementation, testing* and *installation* ***–*** are part of the systems life cycle performed by the main design team. Not be too confused with the Software Development Cycle which is part of the *implementation* stage of the systems life cycle.” | |
|  | **1.2.3a** | Software development | | Waterfall Model | | “An earlier abstract description of the system lifecycle where each identified stage of development flows from the previous one and down to the next one. Feedback from each to the previous takes place independently of the forward flow.” | |
|  | **1.2.3a** | Software development | | Agile Methodologies | | “Describes the responsive development of a system made of small software modules (often web-based applications) by a group of collaborators who work concurrently and closely under a leader who ensures engineering best practice and delivery of the customer requirements.” | |
|  | **1.2.3a** | Software development | | Extreme Programming | | “A software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. It is a form of agile development, and as such advocated frequent releases in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted. Other common elements of XP include: programming in pairs or doing extensive code review, flat management structures and avoiding programming of features until they are actually needed.” | |
|  | **1.2.3a** | Software development | | Spiral Model | | “An abstract description of the systems life cycle where there are four defined quadrants – planning, risk analysis, use of design methods, client and management evaluation. Once one stage of development has gone full circle, the next please takes place, and so on until completion.” | |
|  | **1.2.3a** | Software development | | RAD | | **Rapid Application Development:** “A design strategy that includes online development and repeated prototyping and repeated prototyping and evaluation. It is particularly suited to interactive systems.” | |
|  | **1.2.4a** | Types of programming language | | Programming Paradigm | | “The word ‘paradigm’ means to describe an example or pattern. In a Computing context, this means to describe a computational way of doing things. So a Programming Paradigm is a style or way of programming. E.g. Low-Level languages, High-Level languages, Declarative languages are all examples of different programming paradigms.” | |
|  | **1.2.4b** | Types of programming language | | Procedural Language | | “Any high level language in which program statements can be grouped in self-contained blocks called procedures and functions. These procedures have their own variables, not accessible outside the procedure.” | |
|  | **1.2.4c** | Types of programming language | | Assembly Language | | “A language which is related very closely to the computer’s own machine code. ” | |
|  | **1.2.4c** | Types of programming language | | Machine Code | | “Set of all possible instructions made available by the hardware design of a particular processor. Closest to pure binary.” | |
|  | **1.2.4c** | Types of programming language | | Low Level Language | | “A language which is close to machine code. Related closely to the design of the machine.  A one-to-one language” | |
|  | **1.2.4c** | Types of programming language | | High Level Language | | “A language designed to help a programmer express a computer program in a way that reflects the problem that is being solved, rather than the details of how the computer will produce the solution.  One-to-many language.” | |
|  | **1.2.4c** | Types of programming language | | LMC | | **Little Man Computer:** “An instructional model of a computer, created by Dr. Stuart Madnick in 1965. The LMC is generally used to teach students, because it models a simple von Neumann architecture computer - which has all of the basic features of a modern computer. It can be programmed in machine code or assembly code” | |
|  | **1.2.4d** | Types of programming language | | Immediate Addressing | | “Uses the data in the address field, not as an address, but as a constant that is needed by the program. An example is a routine counting up to 10, which may have the constant ‘10’ supplied in the address field of an instruction. Although the address field cannot hold numbers as large as those that can be stored as data in a memory location, because space has to be left for the operation code field, this is a particularly convenient method of loading constants into the accumulator.” | |
|  | **1.2.4d** | Types of programming language | | Direct Addressing | | “Uses the data in the address field without alteration. This is the simplest method of addressing and also the most common.” | |
|  | **1.2.4d** | Types of programming language | | Indirect Addressing | | “Uses the address field to hold the address of a location that contains the required address. | |
|  | **1.2.4d** | Types of programming language | | Indexed Addressing | | “Modifies the address (either a direct or an indirect address) in the address field by the addition of a number held in a special-purpose registers, called an index register, before the address is used. Index registers are quickly and easily altered providing an efficient way of accessing a range of memory locations, such as in an array.” | |
|  | **1.2.4e** | Types of programming language | | OO | | **Object Orientation:** “Looking at systems by classifying them into real world objects” | |
|  | **1.2.4e** | Types of programming language | | OOP | | **Object Oriented Programming:** “A method of programming which classifies real world objects into classes and encapsulates those objects attributes and behaviours.” | |
|  | **1.2.4e** | Types of programming language | | Class | | “A type definition of an object” | |
|  | **1.2.4e** | Types of programming language | | Object | | “An instance of a class.” | |
|  | **1.2.4e** | Types of programming language | | Base Class | | “Base class contains attributes and methods. It is the highest class and does not inherit from any other class.” | |
|  | **1.2.4e** | Types of programming language | | Superclass | | “A class that has been extended by another class. It allows the extending class to inherit its attributes and methods.” | |
|  | **1.2.4e** | Types of programming language | | Subclass | | “A class that extends another class. The subclass inherits the methods and attributes of the class it extends.” | |
|  | **1.2.4e** | Types of programming language | | Derived Class | | “A derived class is any class that inherits attributes and methods from any other derived class or base class.” | |
|  | **1.2.4e** | Types of programming language | | Instantiation | | “The process of creating an actual named instance of class. The instantiated named copy of the class in an object of that class.” | |
|  | **1.2.4e** | Types of programming language | | Overriding | | “A method in a subclass or derived class which has the same name as a method in one or more of its superclass's. The method supersedes all other versions of the method above it in the inheritance tree.” | |
|  | **1.2.4e** | Types of programming language | | Method | | “A program routine contained within an object designed to perform a particular task on the data within the object. A method can broadly be thought of as a procedure / function from more traditional procedural programming languages.” | |
|  | **1.2.4e** | Types of programming language | | Attribute | | “A single bit of data within a record” | |
|  | **1.2.4e** | Types of programming language | | Inheritance | | “When a derived class is defined it also has all the attributes and methods of the base class.” | |
|  | **1.2.4e** | Types of programming language | | Encapsulation | | “All of the object's attributes are contained and hidden in the object and access to them is restricted to operations of that class.” | |
|  | **1.2.4e** | Types of programming language | | Polymorphism | | “A specialised form of overloading which allows us to create very general object structures, which can be used with a wide range of data types.” | |
|  | **1.3.1a** | Compression, encryption and hashing | | Lossy Compression | | “A compression scheme where their generally involves a loss of resolution in parts of the image where experiences shows that it will be least noticed.” | |
|  | **1.3.1a** | Compression, encryption and hashing | | Lossless Compression | | “A compression scheme that allows the original images to be recreated.” | |
|  | **1.3.1b** | Compression, encryption and hashing | | Length Encoding | | “A very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.” | |
|  | **1.3.1b** | Compression, encryption and hashing | | Dictionary Coding | | “A class of lossless data compression algorithms which operate by searching for matches between the text to be compressed and a set of strings contained in a data structure (called the ‘dictionary’) maintained by the encoder.” | |
|  | **1.3.1c** | Compression, encryption and hashing | | Symmetric Encryption | | “The oldest and best-known encryption technique. A secret key, which can be a number, a word, or just a string of random letters, is applied to the text of a message to change the content in a particular way. This might be as simple as shifting each letter by a number of places in the alphabet.” | |
|  | **1.3.1c** | Compression, encryption and hashing | | Asymmetric Encryption | | “This method of encryption involves using a pair of keys to encrypt and decrypt a message so that it arrives securely. Initially, a network user receives a public and private key pair from a certificate authority.” | |
|  | **1.3.1d** | Compression, encryption and hashing | | Hashing | | “The process of calculating a numeric value from one or more data items. While this value obviously depends on the value of the data items, it need not depend on the meaning attached to themn, simply producing a number that is used within the computer.” | |
|  | **1.3.1a** | Databases | | Relational Database | | “Allows the user to specify information about multiple tables and the relationship between those tables” | |
|  | **1.3.1a** | Databases | | Flat File | | “A database that allows the user to specify data attributes (columns, databases etc.) for only one table at a time, storing those attributes independently” | |
|  | **1.3.1a** | Databases | | Primary Key | | “A field that uniquely identifies a record in a table” | |
|  | **1.3.1a** | Databases | | Foreign Key | | “The linking field in the foreign table formed when a relationship is made. The FK becomes by default the PK of the new table.” | |
|  | **1.3.1a** | Databases | | Concatenated Primary Key | | “When more than one field is added together to form a unique primary key for a table.” | |
|  | **1.3.1a** | Databases | | Secondary Key | | “A key field which can be used to access a table in a different way” | |
|  | **1.3.1a** | Databases | | ERM | | **Entity Relationship Modelling:** “The process of producing a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a data such as a relation database.” | |
|  | **1.3.1a** | Databases | | Normalisation | | “The process of arranging data in tables and setting their relationships to move them through normal forms” | |
|  | **1.3.1a** | Databases | | Indexing | | “The process of creating a database index, which is a data structure that improves the speed of data retrieval operations on a dataset table at the cost of additional writes and storage space to maintain the index data structure.” | |
|  | **1.3.1c** | Databases | | 0NF | | “A table with no normalisation. All data and all fields in one table” | |
|  | **1.3.1c** | Databases | | 1NF | | **First Normal Form:** “A relationship with repeating groups removed. That is a relation in which the intersection of each tuple and attribute contains one and only one value.” | |
|  | **1.3.1c** | Databases | | 2NF | | **Second Normal Form:** “A relation that is in 1NF and every non-primary key attribute is fully dependant on the primary key. That is, all the incomplete dependencies have been removed” | |
|  | **1.3.1c** | Databases | | 3NF | | **Third Normal Form:** “A relation that is in 1NF and 2NF, and in which no non-primary key attribute is transitively dependant on the primary key. That is, all non-key elements are fully dependant on the primary key” | |
|  | **1.3.1c** | Databases | | Normal Forms | | “A way of structuring the data in a relational database according to formal rules, in order to avoid problems of efficiency and security in accessing and maintain the data.” | |
|  | **1.3.1d** | Databases | | SQL | | **Structured Query Language:** “The language and syntax used to write and run database queries” | |
|  | **1.3.1e** | Databases | | Referential Integrity | | “A measure of the consistency of the data in a database. It is violated when the relation to which a foreign key refers to no longer exists.” | |
|  | **1.3.1f** | Databases | | Transaction Processing | | “Information processing that is divided into individual, indivisible operations, called transactions. Each transaction must succeed or fail as a complete unit, it can never be only partially complete.” | |
|  | **1.3.1f** | Databases | | ACID | | **Atomicity, Consistency, Isolation, Durability:** “A set of properties that guarantee that database transactions are processed reliably. In the context of databases, a single logical operation on the data is called a transaction.” | |
|  | **1.3.1f** | Databases | | Record Locking | | “A technique of preventing simultaneous access to data in a database, to prevent inconsistent results. The classic example is demonstrated by two bank clerks attempting to update the same bank account for two different transactions.” | |
|  | **1.3.1f** | Databases | | Redundancy | | “Redundancy occurs in database systems which have a field that is repeated in two or more tables For instance, when customer data is duplicated and attached with each product bought, then redundancy of data is a known source of inconsistency since customer might appear with different values for given attributes.” | |
|  | **1.3.2a** | Networks | | Protocol | | “A set of rules that allow two devices to communicate.” | |
|  | **1.3.2b** | Networks | | TCP/IP Stack | | **Transmission Control Protocol / Internet Protocol:** “The most common general-purpose standard protocol that allows any networked computers (including those on The Internet) to communicate with each other whatever their equipment.” | |
|  | **1.3.2b** | Networks | | DNS | | **Domain Name System:** “The Internet’s equivalent of a phone book. They maintain a directory of domain names and translate them to Internet Protocol (IP) addresses. This is necessary because, although domain names are easy for people to remember, computers or machines access websites based on IP addresses.” | |
|  | **1.3.2b** | Networks | | Protocol Layering | | “The concept of a protocol not simply being a set of rules but those rules being built up into very specific layers and those rule layers behind built on top of each other in a deliberate order creating a layered protocol stack. This results in the rules of a protocol being executed in a specific sequence as you move through the protocol stack.” | |
|  | **1.3.2b** | Networks | | LAN | | **Local Area Network:** “A collection of computers / computing devices on the same network which are physically close together, for example, all located within one building or site e.g. a home or school network.” | |
|  | **1.3.2b** | Networks | | WAN | | **Wide Area Network:** “A collection of computers / computing devices on the same network which are spread out over a geographically large area, for example a university across several campuses, or a multinational corporation with offices / sites in different cities or even different countries.” | |
|  | **1.3.2b** | Networks | | Packet Switching | | “A method of sending data over a wide area network in which the message is broken into a number of parts which are sent independently, over whatever route is optimum for each packet, and reassembled at the destination.” | |
|  | **1.3.2b** | Networks | | Circuit Switching | | “A method of sending data over a wide area network in which two network nodes establish a dedicated communications channel through the network before the nodes may communicate. All data then follows this same path for the duration of the data transfer.” | |
|  | **1.3.2c** | Networks | | Firewall | | “A computer application used in a network to prevent external users gaining unauthorised access to a computer system.” | |
|  | **1.3.2c** | Networks | | Proxies | | “A computer application that accesses data on a different computer system or network. It controls the access of authorised users to data and allows the operation of the system to be isolated from control by external users.” | |
|  | **1.3.2c** | Networks | | Encryption | | “The process of making data in a computer system unintelligible.” | |
|  | **1.3.2e** | Networks | | Client-Server | | “A method of network organisation in which network stations make use of resources available at one or more servers.” | |
|  | **1.3.2e** | Networks | | Peer to Peer | | “A method of network organisation in which network stations can share resources on other network stations, so one station can use a printer on another station or save data on another station’s local storage.” | |
|  | **1.3.4a** | Web technologies | | HTML | | **HyperText Markup Language:** “A mark-up language developed for multimedia documents, such as World Wide Web Pages.” | |
|  | **1.3.4a** | Web technologies | | CSS | | **Cascading Style Sheets:** “A definition of the formatting and layout of elements of an HTML document. The stylesheet may be part of the HTML document, or stored as a separate file linked to the document. The use of different stylesheets linked to the same document allows appropriate layout of the same content on, for example, mobile devices, display screens.” | |
|  | **1.3.4a** | Web technologies | | JavaScript | | “An object-oriented computer programming language commonly used to create interactive effects within web browsers.” | |
|  | **1.3.4b** | Web technologies | | Search Engine Indexing | | “The method of collecting, parsing and storing data to facilitate fast and accurate information retrieval.” | |
|  | **1.3.4c** | Web technologies | | PageRank Algorithm | | “An algorithm used by Google Search to rank websites in their search engine results. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.” | |
|  | **1.3.4d** | Web technologies | | Sever Side Processing | | “Server-side processing refers to operations that are performed by the server in a client-server relationship in computer network. Typically, a server is a computer program, such as a web server, that runs on a remote server, reachable from a user’s local computer.” | |
|  | **1.3.4d** | Web technologies | | Client Side Processing | | “Client-side processing refers to operations that are performed by the client in a client-server relationship in a computer network. Typically, a client is a computer application, such as a web browser, that runs on a user’s local computer and connects to a server as necessary.” | |
|  | **1.4.1a** | Data types | | Primitive Data Types | | “The basic data types provided by a programming language as building blocks. Most languages allow more complicated composite types to be recursively construction starting from basic types. E.g. char, integer, float, Boolean. As an extension a ‘string’ data type is constructed behind the scenes of many char data types.” | |
|  | **1.4.1a** | Data types | | Integer | | “A data type used to store positive and negative whole numbers | |
|  | **1.4.1a** | Data types | | Real | | “A data type used to store an approximation of a real number in a way that can support a trade-off between range and precision. A number is, in general, represented approximately to a fixed number of significant digits and scaled using an exponent.” | |
|  | **1.4.1a** | Data types | | Floating Point | | “A data type used to store an approximation of a real number in a way that can support a trade-off between range and precision. A number is, in general, represented approximately to a fixed number of significant digits and scaled using an exponent.” | |
|  | **1.4.1a** | Data types | | Character | | “A single alphanumeric character or symbol.” | |
|  | **1.4.1a** | Data types | | String | | “A sequence of alphanumeric characters and or symbols. e.g. a word or sentence.” | |
|  | **1.4.1a** | Data types | | Boolean | | “Used to store the logical conditions TRUE / FALSE. Often translated to On/Off, Yes/No etc.” | |
|  | **1.4.1b** | Data types | | Binary | | “Binary describes a numbering scheme in which there are only two possible values for each digit: 0 and 1. The term in computing refers to any digital encoding system in which there are exactly two possible states. E.g. in memory, storage, processing and communications, the 0 and 1 values are sometimes called “low” and “high”, respectively.” | |
|  | **1.4.1c** | Data types | | Sign and Magnitude | | “A method in computing of being able to store and represent floating point real numbers (both positive and negative) as a string of pure binary digits. Uses the concepts of two’s complements, mantissa and exponent.” | |
|  | **1.4.1c** | Data types | | Two’s Complement | | “A method in computing of being able to store negative numbers as string of pure binary digits. It works by turning the MSB into a sign bit, where 0 represents a positive number and 1 represents a negative.” | |
|  | **1.4.1e** | Data types | | Hexadecimal | | “A numerical system of notation which uses 16 rather than 10 as its base. The 16 Hex base digits are 0-9 and the letters A-F.” | |
|  | **1.4.1f** | Data types | | Denary | | “A numerical system of notation which uses 10 as its base. The 10 Decimal base digits are 0-9.” | |
|  | **1.4.1h** | Data types | | Floating Point Arithmetic | | “The mathematical process of performing simply calculations on more than one floating-point number stored in binary notation.” | |
|  | **1.4.1i** | Data types | | Bitwise Manipulation | | “The act of algorithmically manipulating bits or other pieces of data shorter than a word. Programming tasks that require a bit manipulation include low-level device control, error detection and correction algorithms, data compression, encryption algorithms, and optimisation.” | |
|  | **1.4.1i** | Data types | | Shifts | | “An operation that moves the bits held in a register, called the *shift register*, either to the left or the right. There are three different types of shift: arithmetic shift, logical shift and cyclic shift. They are distinguished by what happens to the bits that are shifted out of the register at one end and what is moved in to fill the vacant space at the other end.” | |
|  | **1.4.1i** | Data types | | AND | | “A logical operator used within a program. AND works by only returning TRUE if both values being compared are TRUE.” | |
|  | **1.4.1i** | Data types | | OR | | “A logical operator used within a program. OR works by returning TRUE as long as either value being compared is TRUE.” | |
|  | **1.4.1i** | Data types | | XOR | | “A logical operator used within a program. XOR stands for exclusive OR. It will return TRUE if the two items being compared are different.” | |
|  | **1.4.1j** | Data types | | Character Sets | | “The set of symbols that may be represented in a computer at a particular time. These symbols, called characters, can be letters, digits, spaces or punctuations marks, the set includes control characters.” | |
|  | **1.4.1j** | Data types | | ASCII | | **America Standard Code for Information Interchange:** “A character set devised for early telecommunication systems but proved to be ideal for computer systems. ASCII codes use 7-bits giving 32 control codes and 96 displayable characters (the 8th bit is often used for error checking).” | |
|  | **1.4.1j** | Data types | | UNICODE | | “Standard character set that replaces the need for all the different character sets. It incorporates characters from almost all the world’s languages. It is a 16-bit extension of ASCII.” | |
|  | **1.4.2a** | Data structures | | Array | | “A set of data items of the same type grouped together using a single identifier. Each of the data items is addressed by the variable name and a subscript.” | |
|  | **1.4.2a** | Data structures | | Records | | “A data structure which consists of a collection of elements, typically in fixed number and sequence and typically indexed by names. The elements of records may also be called fields.”    “The record type is a data type that describes such values and variables. Most modern computer languages allow the programmer to define new record types. The definition includes specifying the data type of each field and an identifier by which it can be accessed.” | |
|  | **1.4.2a** | Data structures | | Lists | | “An abstract data type that represents a sequence of values, where the same value may occur more than once. The name list is also used to cover several concrete data structures that can be used to implement abstract lists especially linked lists.” | |
|  | **1.4.2a** | Data structures | | Tuple | | “Another name for a record.” | |
|  | **1.4.2b** | Data structures | | Linkedlist | | “A list where each item contains the data together with a pointer to the next item. There may be an additional pointer to the previous item. This means the items can be accessed in order even if they are not stored in order; they do not have to be stored in adjacent memory locations.” | |
|  | **1.4.2b** | Data structures | | Directed Graph | | “In a directed graph, the order of the vertices in the pairs in the edge set matters.” | |
|  | **1.4.2b** | Data structures | | Undirected Graph | | “In an undirected graph, the order of the vertices in the pairs in the Edge set doesn’t matter.” | |
|  | **1.4.2b** | Data structures | | Stack | | “A stack is a dynamic data structure of the form Last In First Out (LIFO).” | |
|  | **1.4.2b** | Data structures | | Queue | | “A queue is a dynamic data structure of the form First In First Out (FIFO).” | |
|  | **1.4.2b** | Data structures | | Tree | | “A tree is a non-linear dynamic data structure where data items can be thought of as occurring at different levels. There are links between items at one level and their descendants at the next. Each data item has data that relates in some way to its unique parent node. The data items are usually called nodes with the links known as branches. The top level nodes is called the root node.” | |
|  | **1.4.2b** | Data structures | | Binary Search Tree | | “A data structure very similar to a tree with the following additional restrictions. Each node can have only 0, 1 or 2 leaf nodes. All left nodes and all of its descendants have smaller values that the root node, while all right nodes and all of its descendants have larger values than the root node.” | |
|  | **1.4.2b** | Data structures | | Hash Table | | “A data structure where the calculated value is used to mark the position in the table where the data item should be stored, enabling it to be accessed directly, rather than forcing a sequential search.” | |
|  | **1.4.3a** | Boolean algebra | | Boolean Logic | | “Named after the nineteenth-century mathematician George Boole, Boolean logic is a form of algebra in which all values are reduced to either TRUE or FALSE.” | |
|  | **1.4.3b** | Boolean algebra | | Karnaugh Maps | | “Also know as the K-Map, is a method to simplify Boolean algebra expressions. The Karnaugh map reduces the need for extensive calculations by taking advantage of humans’ pattern-recognition capability. They are used to simplify real-word logic requirements so that they can be implemented using a minimum number of physical logic gates.” | |
|  | **1.4.3c** | Boolean algebra | | Boolean Algebra | | “A set of rules for manipulating truth values according to truth tables. Very important in computing as truth values in Boolean algebra are True and false, and can thus easily be represented as the binary digits 1 and 0.” | |
|  | **1.4.3c** | Boolean algebra | | De Morgan’s Law | | “Two laws in Boolean algebra which state that AND and OR, or union and intersections, are duel. The rules can be expressed in English as 1) ‘The negation of a conjunction is the disjunction of the negations.’  2) ‘The negation of a disjunction is the conjunction of the negations.’ Or more informally as 1) "not (A and B)" is the same as "(not A) or (not B)" and also 2) "not (A or B)" is the same as "(not A) and (not B)".  The purpose of De Morgan’s laws is to simplify the design of electronic circuits.” | |
|  | **1.4.3c** | Boolean algebra | | Distribution | | “A rule or law in Boolean algebra which permits the multiplying or factoring out of an expression.” | |
|  | **1.4.3c** | Boolean algebra | | Association | | “A rule or law in Boolean algebra which permits the removal of brackets from an expression and regrouping of the variables.” | |
|  | **1.4.3c** | Boolean algebra | | Commutation | | “A rule or law in Boolean algebra which stats that the order of application of two separate terms is not important: A AND B = B AND A.” | |
|  | **1.4.3c** | Boolean algebra | | Double Negation | | “A rule or law in Boolean algebra where if you invert a term twice it is equal to its original term: (NOT NOT A) = A” | |
|  | **1.4.3d** | Boolean algebra | | Logic Gate Diagram | | “A method of expression Boolean Logic in a diagrammatic form using a set of standard symbols representing the various Logic Gates such as AND NOT OR NAND etc.” | |
|  | **1.4.3d** | Boolean algebra | | Truth Table | | “A notation used in Boolean algebra for defining the output of a logic gate or logic circuit for all possible combinations of inputs.” | |
|  | **1.4.3e** | Boolean algebra | | D Type Flip Flops | | “Also known as a data or delay flip flop. This is a circuit or logic design which can be viewed as a memory cell. It has two stable states. Using appropriate input signals, you can trigger the flip-flop from one state to the other.” | |
|  | **1.4.3e** | Boolean algebra | | Half Adders | | “A unit which adds together two input variables. A half adder can only add the inputs together.” | |
|  | **1.4.3e** | Boolean algebra | | Full Adders | | “A unit which adds together two input variables. A full adder can a bit carried from another addition as well as the two inputs.” | |
|  | **1.5.1a** | Computing related legislation | | DPA | | **The Data Protection Act 1998:** “Legislation which protects individuals from unreasonable use of their store personal data.” | |
|  | **1.5.1b** | Computing related legislation | | CMA | | **The Computer Misuse Act 1990:** “Legislation which defines electronic vandalism, unauthorised access to computer systems and theft of information.” | |
|  | **1.5.1c** | Computing related legislation | | CDPA | | **The Copyright Design and Patents Act 1988:** “Legislation which gives creators of literacy, dramatic, musical and artistic works the right to control the ways in which their material may be used.” | |
|  | **1.5.1d** | Computing related legislation | | RIPA | | **The Regulation of Investigatory Powers Act 2000:** “Legislation which regulates the powers of public bodies to carry out surveillance and investigation, and covering interception of communications.” | |
|  | **2.1.1a** | Thinking abstractly | | Abstraction | | “The process of separating ideas from specific instances of those ideas at work. Computational structures are defined by their meanings, while hiding away the details of how they work. Abstraction tries to factor out details from a common pattern so that programmers can work close to the level of human thoughts, leaving out details which matter in practice, but are immaterial to the problem being solved.” | |
|  | **2.1.1d** | Thinking abstractly | | Abstract Model | | “Any model of a system which has been taken from, or based on (abstracted) a real life situation or problem.” | |
|  | **2.1.2a** | Thinking ahead | | System Inputs | | “Anything which will be required to go into a system in any form in order for it to operate as intended. These inputs are then processed by the system following the rules or algorithms of the system.” | |
|  | **2.1.2b** | Thinking ahead | | System Outputs | | “Anything which exits a system in any form in order for it to operate as intended. These outputs are often the result of complex processing of inputs along with other data.” | |
|  | **2.1.2b** | Thinking ahead | | Solution Preconditions | | “Any condition which pre-exists the suggested solution. For example for a road traffic speed camera program their would be pre-conditions which could be as simple as requiring a road in place and a vehicle traveling along it.” | |
|  | **2.1.5b** | Thinking concurrently | | Concurrent Processing | | “Any situation in the design or programming of a system when you would want more than one thing happening at the same time.” | |
|  | **2.2.1a** | Programming techniques | | Sequence | | “One of the 3 basic programming constructs. Instructions happening one after the other in order is sequence.” | |
|  | **2.2.1a** | Programming techniques | | Iteration | | “One of the 3 basic programming constructs. A selection of code which can be repeated either a set number of times (count controlled) or a variable number of times based on the evaluation of a Boolean expression (condition controlled) is iteration.” | |
|  | **2.2.1a** | Programming techniques | | Branching / Selection | | “One of the 3 basic programming constructs. Instructions which can evaluate a Boolean expression and then branch the code to one or more alternatives paths is branching / selection.” | |
|  | **2.2.1b** | Programming techniques | | Recursion | | “An advanced programming construct in which a block of code (often a function) is able to call itself. Recursion algorithms must be designed carefully so that terminating condition will be met.” | |
|  | **2.2.1c** | Programming techniques | | Global Variable | | “A variable which can be used anywhere in the program.” | |
|  | **2.2.1c** | Programming techniques | | Local Variable | | “A variable which is defined and can only be used within one part of the program (normally a single function or procedure).” | |
|  | **2.2.1d** | Programming techniques | | Modularity | | “A technique of code design where a solution is broken down into a number of small self-contained and manageable chunks. Each Module of code should be designed to perform one set specific purpose. If during design it is found that the module starts to grow and performs more than one task then the additional functionality should be split out into a new module.” | |
|  | **2.2.1d** | Programming techniques | | Functions | | “A block of code given a unique identifiable name within a program. A function can take either zero or more parameters when it is called and should return a value. The function should be designed and written to perform one task or action which is clearly indicated by its name.” | |
|  | **2.2.1d** | Programming techniques | | Procedures | | “A block of code given a unique identifiable name within a program. A procedure can take either zero or more parameters when it is called. The procedure should be designed and written to perform one task or action which is clearly indicated by its name.” | |
|  | **2.2.1d** | Programming techniques | | Parameters | | “Data structures passed into a Procedure or Function when they are initially called.” | |
|  | **2.2.1d** | Programming techniques | | Parameter Passing | | “The process of providing a procedure, function or module with values at the point when you call it.” | |
|  | **2.2.1d** | Programming techniques | | * By Value | | “If a data item is passed by value, a (local) copy of the data is used, which is discarded when the subprogram exits.” | |
|  | **2.2.1d** | Programming techniques | | * By Reference | | “If a data item is passed by reference, the location (in memory) of the data is used. This means that any changes are retained after the procedure or function has been completed.” | |
|  | **2.2.1e** | Programming techniques | | IDE | | **Integrated Development Environment:** “Software that performs the various stages of software design and implementation in a single integrated system. It will usually include facilities for project management, design, graphical design, programming, testing and producing documentation.” | |
|  | **2.2.1e** | Programming techniques | | Debugging | | “The process of removing syntactical, semantic (logical) and run-time errors from computer code.” | |
|  | **2.2.2a** | Computational methods | | Computational Methods | | “Describes any method which does something related to computation, such as sorting, merging, searching, indexing etc.” | |
|  | **2.2.2b** | Computational methods | | Problem Recognition | | “The acknowledgement and definition of an issue that does or may arise during the performance of a process.” | |
|  | **2.2.2c** | Computational methods | | Problem Decomposition | | “The process by which a complex problem or system is broken down into parts that are easier to conceive, understand, program and maintain.” | |
|  | **2.2.2d** | Computational methods | | Divide and Conquer | | “Another term for decomposition which is the process by which a complex problem or system are broken down into parts that are easier to conceive, understand, program and maintain.” | |
|  | **2.2.2f** | Computational methods | | Backtracking | | “An algorithm for finding a complete solution. This is a refined brute force methodology. It is a very general algorithm for finding all (or some) solutions to computational problems. It incrementally builds to the solution, and abandons each partial success (“backtracks”) as soon as it determines that the partial solution cannot possible be completed to a valid solution.” | |
|  | **2.2.2f** | Computational methods | | Data Mining | | “The analysis of a large amount of data in a data warehouse to provide new information. For example, by using loyalty cards, which connect purchases to a particular customer, supermarkets can gathering information about the buying habits of individual customers.” | |
|  | **2.2.2f** | Computational methods | | Heuristics | | “Rules that are not derived purely from logic but are derived from the experience (of a person). These are also known as ‘rules of thumb’.” | |
|  | **2.2.2f** | Computational methods | | Performance Modelling | | “This is the process of carrying out mathematical approximations of how well models perform. e.g. How efficient they are.” | |
|  | **2.2.2f** | Computational methods | | Visualisation | | “The process of turning data in a visual representation which is easier for humans to understand, interrupt and work with.” | |
|  | **2.3.1a** | Algorithms | | Algorithm | | “A sequence of steps designed to perform a particular task. An algorithm may be constructed to describe the operation of a complete system or to describe a particular part of it.” | |
|  | **2.3.1c** | Algorithms | | Big O Notation | | “Used in computer science to describe the performance or complexity of an algorithm. Big-O specifically described the **worst-case** scenario, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.” | |
|  | **2.3.1f** | Algorithms | | Bubble Sort | | “A simple algorithm popular with inexperienced programmers. It is inefficient when sorting large amounts of data as the time taken is related to the square of the number of items. If 10 items take 1ms then 100 times will take 100ms (this is 10 times the number of items and so the time will be 102 or 100 times longer).” | |
|  | **2.3.1f** | Algorithms | | Insertion Sort | | “A simple sorting algorithm that builds the final sorted array (or list) one item at time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.” | |
|  | **2.3.1f** | Algorithms | | Merge Sort | | “A type of divide and conquer algorithm that was incited by John von Neumann. First the list is divided into the smallest unit (1 element), then each element is compared with the adjacent list to sort and merge the two adjacent lists. Finally all elements are sorted and merged.” | |
|  | **2.3.1f** | Algorithms | | Quick Sort | | “A type of divide and conquer algorithm which sorts the given sequence in place meaning that it doesn’t require extra storage as would be needed in a merge sort. The basic idea is dividing the sequence into two sub-lists around an element which is called the pivot such that all elements in the lower sub-list are less than the value of the pivot element and all elements in the higher sub-list are greater than the pivot element.” | |
|  | **2.3.1f** | Algorithms | | Dijkstra’s Shortest Path | | “A graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms. In practice in picks an unvisited vertex with the lowest-distance, calculates the distance through it to each unvisited neighbour, and updates the neighbours distance if smaller. It the marks the visited when done with neighbours.” | |
|  | **2.3.1f** | Algorithms | | A\* Algorithm | | “Widely used in pathfinding and graph traversal, the process of plotting an efficiently traversable path between points, called nodes. A\* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals). As A\* traverses the graph, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments along the way.” | |
|  | **2.3.1f** | Algorithms | | Binary Search | | “A particularly efficient search method. It only works if records in the file are in sequence. A binary search involvers accessing the middle record in the file and determining if the target record has been found or, if not, if it is before or after in the sequence. This process is repeated on the part of the file where the target record is expected, until it is found.” | |
|  | **2.3.1f** | Algorithms | | Linear Search | | “Involves examining each entry in turn in the file until the time is found or the end of the file is reached. Unless the file is in some useful order a serial search has to be used.” | |