# Computer Systems

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data representation** | **Describe and exemplify the use of binary to represent positive integers.**  **Convert from binary to denary and vice-versa.**   * Everything stored by a computer is stored using Binary – 0 & 1 * 37 =  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |   **Describe floating point representation of positive real numbers using the terms mantissa and exponent.**   * The mantissa and exponent are stored as separate numbers. The computer recreates the number for calculations when it needs to.   **Describe extended ASCII code (8-bit) used to represent characters.**   * Each character is given an 8-bit ASCII Code. * For example – V = 86 = 01010110   **Describe the vector graphics method of graphic representation for common objects:**   * Vector graphics store a picture by storing each objects attribute – e.g. the instructions to draw the shape. * **Rectangle** - (height, width, x, y, fill colour, line colour) * **Ellipse** - (cx, cy, rx, ry, fill colour, line colour) * **Line** - (x1, y1, x2, y2, line colour) * **Polygon** -*any shape with 3 of more sides* - (x1, y1, x2, y2, x3, y3, fill colour, line colour)   **Describe the bit-mapped method of graphics representation.**   * Computers stores, in binary, each pixel. Number of bits per pixel depends on the number of colours in the image – more colours, more bits. |
| **Computer structure** | **Describe the purpose of the basic computer architecture components and how they are linked together:**  **Processor**   * **Control Unit** – controls the sequencing of fetching, decoding and executing instructions. * **Arithmetic Logic Unit** – performs all calculations and logical operations * **Registers** – temporary memory locations within the processor.   **Memory locations with unique addresses**   * All memory locations within RAM is given a unique address so that the computer can read and write data to that location correctly.   **Buses**   * **Address Bus** –this identifies the memory location that is going to read from or written to. * **Data Bus** – this transfers the data between the processor and memory, and vice-versa.     **Explain the need for interpreters and compilers to translate high-level program code to binary.**   * **Interpreter** – translates the program one line at a time into machine code and executes immediately. Machine code is not retained, so must be translated each time. * Good for testing, errors are highlighted straight away. * **Compiler** – translate whole program, and then save machine code version. Translate code can be run again, and again. Used for programs that are ready for release. |
| **Environmental impact** | **Describe the energy use of computer systems, the implications on the environment**   * It is estimated that two billion computer systems are in use in the world – these all use energy. As user we need to consider how to reduce unnecessary energy use.   **and how these could be reduced through:**   * **Settings on monitors** – reduce brightness, activate efficiency mode, use energy efficient monitors. * **Power down settings** – power-down after a period of inactivity, control setting for single components. * **Leaving computers on standby** – power consumption is reduced. |
| **Security precautions** | **Describe the role of firewalls.**   * A firewall protects against suspicious access to a computer. When an external computer tries to access a computer, the firewall decides if the external computers access should be granted or denied.     **Describe the use made of encryption in electronic communications.**   * Encryption codes the information so that if it is intercepted then they would not be able to make sense of it. It can only be understood by the sender and received. |

# Website Design & Development

|  |  |
| --- | --- |
| **Analysis** | **Identify the end-user and functional requirements of a website problem that relates to the design and implementation at this level.**   * **End-user** – describes the group of people most likely to use the website. E.g. Children, employees, etc. * **Functional requirements** – things that the website must be able to do e.g. Search a database. |
| **Design** | **Describe and exemplify the website structure with a home page, a maximum of four linked multimedia pages, and any necessary external links.**   * **Linear** – users move from one page to another in sequence * **Hierarchical –** pages are accessed dependent upon their position within the hierarchy.     **Wireframing is used to design the visual layout of a web page. Wireframing should clearly show:**   * **navigational links**    + ***Internal*** links take the user to another page within site,  ***External***links take the user to a different site.   + ***Absolute*** address gives the full URL,***Relative***address give URL based on current location. * **consistency across multiple pages:**   All pages of website should follow the same design rules, so that the user knows what to expect.   * **relative vertical positioning of the media displayed** – where text, graphics audio & video will be placed. * **file formats of the media (text, graphics, video, and audio)** – see next page   **Describe & identify the implications for individuals and businesses of the Copyright, Designs and Patents Act**   * The text, graphics, video and audio contained within this website are protected. * The user is prevented from making, using or distributing copies without the owner’s permission. |
| **Design *(cont.)*** | **Compare a range of standard file formats:**   * **Audio:**   + **wav** – uncompressed, high quality, large file size.   + **mp3** – lossy compression, good quality, smaller file size. * **Graphics**   + **jpeg** – lossy compression, transparency & animation not supported   + **gif** – lossless compression, animation & transparency supported, 8-bit colour.   + **png** – lossless compression, animation & transparency supported, 48-bit colour     **Describe the factors affecting file size and quality:**   * **Graphic Resolution** – the more pixels per inch, the better the quality, but the larger the file size. * **Colour Bit Depth** – more bits per colour, more colours, but a larger file size. * **Sampling Rate** - the more samples taken per second, more realistic the audio, but the larger the file size     **Describe the need for compression.**   * Compression is the process of reducing the size of a file. Compression is needed to save backing storage, or the time used to send the file between two computers i.e. over the internet.     **Describe, exemplify and implement prototyping (low-fidelity) from wireframe design at this level.**   * A prototype is a draft of the interface that is produced early in the development process. It should closely resemble the intended look and feel of the final product so user of it get a good idea of the finished product. |
| **Implementation (CSS)** | **Describe, exemplify and implement Cascading Style Sheets (CSS):**  CSS is used to define the colours, fonts and layout of web page content.  **Internal & External CSS**   * **Internal** **Stylesheet** holds the CSS code for the webpage in the <head> section of an HTML file. * **External Stylesheet** holds the CSS code in a separate CSS file, which is liked to an HTML document.      * **Selectors** – the html elements that you want to style. In the example below H1   H1 {color: blue; font-size: 12px;}   * **ID ­**– when writing HTML it is possible to give an element its own unique ID, CSS can then be applied to it. An ID can only be used once in an HTML document.   + HTML - <p id= “intro”> Introduction Paragraph </p>   + CSS - #intro {color: black; font-family: verdana;} * **Class** – worked similar to an ID, but can be applied to multiple HTML tags.   + HTML - <p class = “important”> This is important </p>   + CSS - .important {background-color: yellow; text-align: center;**}** |
| **Implementation (JavaScript)** | **Describe and identify Javascript coding related to mouse events:**  Using JavaScript, functions are created that will change what appears on screen dependant on what the user does with the mouse.  Line 1: <script>  Line 2: function mouseOver() {  Line 3: document.getElementById("demo").style.color = "red";  Line 4: }  Line 5: function mouseOut() {  Line 6: document.getElementById("demo").style.color = "black";  Line 7: }  Line 8: </script>  Line 2-4 – when the mouse it over and element with the ID “demo” the colour will change to red.  Lines 5-6 – when the mouse it moved away from the above element the colour will change to black. |
| **Implementation (HTML)** | **Describe, exemplify and implement HTML code:**     * **<html> … </html>** - tells the browser the rest of the document contains HTML * **<head> … </head>** - contains information about the page itself – including CSS * **<title> … </title>** - title of the webpage, appears on browser tabs. * **<body> … </body>**  - page content is placed between these tags. * **<h1> … </h1>** - creates a heading, there are six different sizes (h1 to h6) * **<p> … </p>** - defines a paragraph * **<div> … </div>** - used so CSS can be applied to a group of elements * **<a href = “link url”> link text </a>** - creates a hyperlink – can take user to different page/ website * **<img src = “image url”>** - inserts an image. * **<audio> <source src = “audio url”> </audio>** - inserts an audio file * **<video> <source src = “video url”> </video>** - inserts a video file * **lists**    + **<ol> … </ol>** - creates an ordered list - numbered   + **<ul> … </ul>**  - created an un-ordered list – bulleted   + **<li> … </li>** - adds a list item     **Describe and implement hyperlinks (internal and external), relative and absolute addressing.**  See above    **Read and explain code that makes use of the above HTML.** |
| **Testing** | **Describe and exemplify testing:**     * **matches user-interface design** – does it match the wireframe design and prototype. * **links and navigation work correctly** – when links are clicked do they take you to the correct location * **media (such as text, graphics, and video) display correctly** – are they in the correct place, do they work * **consistency** – do all pages follow a similar design. External CSS may mitigate this. |
| **Evaluation** | **Evaluate solution in terms of fitness for purpose**   * Does the website meet the end-user and functional requirements?   + Yes – then website is fit for purpose   + No – revisit previous phases in the development process |

# Database Design & Development

|  |  |
| --- | --- |
| **Analysis** | **Identify the end-user and functional requirements of a database problem that relates to the implementation at this level.**   * **end-user** – what does the end user want to able to do with the database e.g. search for student data * **functional requirements** – the tasks the database should perform/ data it should hold e.g. first name |
| **Design** | **Describe and identify the implications for individuals and businesses of the Data Protection Act 1998.**  The Data Protection Act (1998) is designed to ensure that personal data is held securely and is not shared with unauthorised third parties. Businesses storing personal data must ensure   * they have prior consent of data subject (the person the data is about) to store personal information * the data is accurate and up-to-day * the data used for limited, specifically stated purposes * the data kept safe and secure   **Describe and exemplify entity-relationship diagrams with two entities.**   * **Entity** – a place where information about a person, thing or concept is collected – a table. * **Attribute** – describe the facts, details or characteristic of an entity – a field.   **An Entity-Relationship diagram shows the relationship that exists between two entities.**  Machine generated alternative text: Primary Key  Uniquely identified each  record in this table.  ACTIVITY  Activity ID  Title  Age Range  Venue  Teacher ID*  Foreign Key  TEACHER  Teacher ID  Name  Teaching Room  Subject  Primary Key  Uniquely identified each  record in this table.  Uses to link this table to the other.  It's the Primary Kev in the other table.  **Describe and exemplify a data dictionary:**  A data dictionary includes the name and description of the attributes in each entity within a database. *Table: Customers*  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Field** | **Key** | **Attribute Type** | **Required** | **Validation** | | Customer\_ID | Primary Key | Number | Yes | Length = 3 | | Title |  | Text | Yes | Restricted Choice = Mr, Mrs, Miss, Ms, | | First Name |  | Text | Yes |  | | Surname |  | Text | Yes |  | | Flight Date |  | Date | Yes |  | | First Class |  | Boolean | No |  | | Flight Number | Foreign Key | Text | Yes | Restricted Choice |   **Attribute Types:**   * **Text** – a combination of text and number e.g. name, address, telephone number. * **Number** – whole numbers and decimal numbers. * **Date** * **Time** * **Boolean** – true or false   **Validation** – ensure data is allowable and sensible, makes it more difficult to make mistakes.   * **Range** – ensures data if between a lower and upper limit * **Restricted Choice** – limits the user to a list of options. * **Length Check** – restricts the number of characters that can be entered. * **Presence Check** – ensures that the field is not left blank – it is required |
| **Implementation** | **Implement relational databases with two linked tables, to match the design with referential integrity.**  ***Referential integrity*** ensures that the foreign key value has a matching value in the corresponding primary key.    **Describe, exemplify and implement SQL operations for pre-populated relational databases.**  SQL standard for Structured Query Language, it is used to interrogate the data held in a database.  **SELECT … FROM …** – is used to select data from the database – SELECT name, gender FROM student;  **… WHERE** – where is used to ensure that only records that meet a specific condition are returned.  SELECT name, gender FROM student WHERE regClass = “1S1”;  **... ORDER BY** – used to order the results in either ascending or descending order. ORDER BY age ASC  SELECT firstname, surname, gender  FROM student  WHERE regClass = “1S1”  ORDER BY surname ASC;  **INSERT INTO** – is used to add a record to a table. INSERT INTO Pupil  VALUES (Ben, Simon, Male, 10/10/05, 1S1);  **UPDATE** – is used to update a current record held in the database. The WHERE identifies specific records.  UPDATE pupil  SET Reg Teacher = “Dr. G. Wilson”  WHERE Reg Teacher = “Mr E. Neil”  **DELETE** – delete a record from the database. No WHERE will mean all content is deleted from the table.  DELETE FROM pupil  WHERE firstName = “Eva” AND surname = “Forrest”;  **EQUI-JOIN** – used to display result from two different tables. Requires Primary Key & Foreign Key to be linked  SELECT firstname, surname, instrument  FROM student, musicTution  ***WHERE student.studentCode = musicTution.studentCode AND year < 4:***  *List the name and instrument played by all pupils in S1-3*  **Operators**   * **AND** – both conditions must be true - WHERE regClass = “1S1” AND gender= “female”; * **OR** – either condition must be true - WHERE year = “1” OR year= “2”; * **<** - less than - WHERE age < 18; * **>** - greater than - WHERE year > 3;   **Read and explain code that makes use of the above SQL.** |
| **Testing** | **Describe and exemplify testing SQL operations work correctly at this level**   * When testing queries, it is good practice to document expected results and actual results.   A comparison can then be made, with testing passed if the expected result and actual result is the same. |
| **Evaluation** | **Evaluate solution in terms of fitness for purpose & accuracy of output**   * A database is deemed fit for purpose if it meets the requirements determined at the analysis stage. If the database is not fit for purpose, it will be necessary to revisit previous phases of the development process. |

# Software Design & Development

|  |  |
| --- | --- |
|  | **Describe and implement the phases of an iterative development process:**   * **Analysis** – production of a software specification that will outline what the software must do * **Design** – plan the design of the program, and the user interface * **Implementation** – write the code for the program/ software * **Testing** – fully test the program/ software to ensure that all aspect work correctly * **Documentation** – technical guide – how to install software - and user guide – how to work the software. * **Evaluation** – is the program fit for purpose and robust   This is an iterative process – each stage may have to be revisited as a result of new information coming to light. |
| **Analysis** | **Identify the purpose and functional requirements of a problem in terms of:**   * **Inputs** – data that is entered into the program. * **Processes** – what the program does with the inputted data. * **Output** – data that is outputted to the user. |
| **Design** | **Identify the data types and structures required for a problem that relates to the implementation at this level, as listed below.**    **Describe, identify, and be able to read and understand:**     * **Structure diagrams** – breaks the problems down into smaller sections. * **Flowcharts** – shows what is going on in a program and how data flows around it.   Image result for bbc bitesize flow charts   * **Pseudocode** – structured English for describing a program, read by a human   **get age from user**  **if age < 15?**  **Send discount to user**  **else**  **Send no discount to user**  **end if**    **Exemplify and implement one of the above design techniques to design efficient solutions to a problem.**  Below are the main flow chart symbols.  Image result for bbc bitesize flow chartsImage result for bbc bitesize flow charts  **Describe, exemplify, and implement user-interface design, in terms of input and output, using a wireframe.**  A wireframe is designed to provide a visual draft of how the interface could look. A wireframe could be anything from a rough paper-based sketch to a full and detailed image. |
| **Implementation (Data Types & Structures)** | **Describe, exemplify, and implement appropriately the following:**   * **Data types**   + **string** – stores text and alphanumeric combinations e.g. Sausage, AB12 4RT   + **character** – stores a single character which can be a letter, number or symbol e.g. Q. #, 6   + **integer** – stores positive and negative whole numbers e.g. 17. -45. 124   + **real** – stores numbers with decimal places, can also store integers e.g. 3.14, - 5.67   + **boolean** – stores TRUE or FALSE only. * **Data structures -**    + **arrays** – stores a list of related data e.g. names of pupils in a class. Image result for arrays programming |
| **Implementation (Computational Constructs)** | **Describe, exemplify, and implement the appropriate constructs in a high-level language:**     * **Assigning values to variables**.   Variables are locations in memory where programs store and retrieve information.  SET score TO 5  SET score TO score + 12  SEND score TO DISPLAY → 17 will be outputted to the user.   * **Arithmetic operations**   + **addition** – SET answer TO a + b   + **subtraction** – SET answer TO a – b   + **multiplication** – SET answer TO a \* b   + **division** – SET answer TO a – c   + **exponentiation** – *to the power off* ***–*** SET answer TO a^b *e.g. 2^3 = 8* * **Concatenate strings -** concatenation is the process of joining two or more strings together to make a new text string. SET displayName to firstName **&** surname. * **Selection -** The program selects an action dependant upon the value of variables held in the program   + **If … Then** – IF *condition* THEN *action* END IF   + **If …** **Then … Else** – IF *condition* THEN *action\_a* ELSE *action\_b* END IF   + **If …** **Then … Else If** – IF *condition\_1* THEN *action\_a* ELSE IF condition\_2 THEN action\_bEND IF * **Conditional Statements**   + **Simple Conditions** – depends on one conditional statement being true.  |  |  |  | | --- | --- | --- | | Operator | Explanation | Example | | < | Less than | age < 18 | | > | More than | age > 18 | | <= | Less than or equal to | age <= 18 | | >= | More than or equal to | age >= 18 | | = or == | Equality | age == 18 | | ≠ or != | Inequality/Not equal to | age != 18 |      * + **Complex Conditions**     - **AND** – both conditions must be true e.g. gender = “M” AND age > 18     - **OR** – either of the conditions must be true e.g. class = 1 OR class = 3     - **NOT** – the condition should not be true e.g. NOT (gender = “M”) |
| **Implementation (Computational Constructs) – *cont.*** | * **Iteration and repetition -** Iteration is the process where program repeat a group of instructions.   + **Fixed Loop** – when a group of instructions is repeated a pre-determined number of times.   FOR counter 0 TO 5 DO  SEND “I must behave” TO DISPLAY  End For   * + **Conditional Loop** – will keep repeating a group of instructions until a specific conditional is met.   GET password FROM (STRING) KEYBOARD  WHILE password != “letmein” DO  SEND “In-correct” TO DISPLAY  GET password FROM (STRING) KEYBOARD  END WHILE   * **Predefined functions (with parameters):** A pre-defined function is a segment of code that can be referred to and it will do something useful.   + **Round** – rounds a value to a specific number of decimal places.   SET pi TO 13.141592  SET pi TO ROUND (pi, 2) *→ the new value of pi will be 13.14*   * + **Length** – give the number of character present in a variable.   SET word TO “Mississippi”  SET numWord TO LENGTH (word) *→ the value of numWord would be 11*   * + **Random** – generates a random number, between a lower and upper range.   SET bonusBall to RANDOM (1,59) |
| **Implementation (Computational Constructs)** | **Describe, exemplify, and implement standard algorithms:**     * **Input validation ­**- used to check that the input is acceptable.   *A program must ensure the user enters a number between 1 & 100.*   |  |  | | --- | --- | | GET score FROM (INTEGER) KEYBOARD  WHILE score < 1 or score > 100 DO  SEND “In-Valid” TO DISPLAY  GET score FROM (INTEGER) KEYBOARD  END WHILE | *Get initial score from the user*  *Check if data if valid, if invalid then enter loop*  *Display error message*  *Get new score from user*  *Repeat loop until data is valid* |  * **Running total within loop**   *A program must calculate the total cost of 5 items.*   |  |  | | --- | --- | | SET total TO 0  FOR counter 0 TO 5  GET itemPrice FROM (REAL) KEYBOARD  SET total = total + itemPrice  END FOR | *Set the initial total to 0*  *Fixed loop, 5 times.*  *Receive item price from the user.*  *Add the item price to the total*  *End Loop* |  * **Traversing a 1-D array** – to transverse an array means to access each element stored in the array so that the data can be checked or used as part of a process.   *A program is required to count all A grades.*   |  |  | | --- | --- | | SET grade TO [A, B, C, A, A, B, A, C]  SET numA TO 0  FOR counter 0 TO 7  IF grade[counter] == “A” THEN  numA = numA + 1  END FOR | *Array with all grades*  *Number to A grades counted*  *Fixed loop, loop for each item in array*  *Check the grade at index if it an a*  *Then count the A*  *End Loop* | |
| **Testing** | **Describe, identify, exemplify, and implement normal, extreme, and exceptional test data.**  The purpose of testing is to detect and remove errors in a program. Programs should be comprehensively tested to see whether they give the correct results when dealing with normal, extreme and exceptional test data.   * **Normal** – data that you would expect to work or be accepted * **Extreme** – data at the lower and upper limits of what is expected. * **Exceptional** – data that should not be accepted by the program.      * **Test Table** – provide a structure to testing, and evidence that testing has taken place.   N5 Computing Science test table  As one test has failed, the programmer should now review the code to identify why this test has failed  – and rectify the error.  **Describe and identify syntax, execution, and logic errors.**   * **Syntax** – This is an error in the spelling or grammar used when coding.  For example, misspelt command words, missing brackets, commands or colons.  Syntax errors are identified by the interpreter, as the code will not run. * **Execution** – This type of error occurs when the program is asked to do something that it cannot, resulting in a crash. For example, dividing by zero or trying to store a string a integer variable. * **Logic** – This is an error in the logic of the code e.g. using a < instead of >. The program will run, but it will produce unexpected results. Logic errors are usually only resolved by carrying out testing. |
| **Evaluation** | **Describe, identify, and exemplify the evaluation of a solution in terms of:**     * **Fitness for purpose** – does the program do what it is supposed to do? * **Efficient use of coding constructs** – a program is efficient if the length of the code is proportional to the scale of the project.   + **use of loops** –repeating instructions is more efficient than a sequence of individual instructions.   + **arrays** – using arrays to store related data is more efficient than using multiple variables.   + **nested IFs** – using IF … ELSE IF … is more efficient that using multiple IF … THEN * **Robustness** – can the program cope with unexpected input or mishaps without crashing? * **Readability:**    + **internal commentary** – add descriptions, notes and explanation to the code.   + **meaningful identifiers** – use variable names that describe what the variable contains.   + **indentation –** makes it easier to identify constructs. Mandatory in Python.   + **white space** – help makes it clearer where code is placed. |