**Knowledge Organiser:
Logical Thinking and Data Representation**

**Introduction**

This knowledge organiser covers essential topics in logical thinking, logic circuits, algorithms, compression, and data representation as per the 2014 National Curriculum in England for Year 9 Computer Science**. Use this guide to strengthen your understanding and complete the practice activities in preparation for your upcoming assessment in January.**

 **All activities are in RED.**

**Section 1: Logical Thinking and Logical Deduction**

**Key Terms:**

* **Logical Thinking**: The process of using reasoning and logic to solve problems.
* **Logical Deduction**: Drawing conclusions from premises or facts.
* **Algorithms**: A step-by-step procedure for calculations.

**Creating an Algorithm modelled Example:**

Task: Making a Cup of Tea1. Boil water.2. Put a teabag in a cup.3. Pour boiling water into the cup.4. Let the tea steep for 3 minutes.5. Remove the teabag.6. Add milk and sugar if desired.7. Stir and enjoy.

Now, write your own algorithm for a different task (e.g., brushing your teeth, crossing the road, making breakfast).

**Algorithm task:**
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Logical Deduction Modelled Example:**

Premise 1: All cats are mammals. Premise 2: Whiskers is a cat. Conclusion: Therefore, Whiskers is a mammal.

**Logical deduction task:**
Now create your own logical deduction using these premises:
Premise 1: All birds have feathers. Premise 2: A parrot is a bird.

**Conclusion:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Section 2A: Logic Gates**

Logic gates are the fundamental building blocks of digital circuits, processing logical statements to facilitate decision-making in computers. George Boole's pioneering work in Boolean algebra laid the groundwork for these logical systems, establishing how AND, OR, and NOT operations form the basis of computational problem-solving.

**Key Terms**

* **Logic Gates**: Basic building blocks of digital circuits (AND, OR, NOT).
* **AND Gate**: Outputs true only if both inputs are true.
* **OR Gate**: Outputs true if at least one input is true.
* **NOT Gate**: Outputs the opposite of the input.



**Logic Statement Task**

For each of the following statements, identify which logic gate (AND, OR, NOT) would be most suitable:

**Statement 1:** The light should turn on if both the front door and the back door are unlocked.

**Statement 2:** The alarm should go off if either the window or the door sensor is triggered.

**Statement 3:** The heating should be off if the thermostat is set to 'off'.

**Section 2B: Truth Tables**

**What is a Truth Table?**

A truth table is a mathematical table used to determine the output of logic gates based on all possible combinations of their inputs. Each row of the table represents a different combination of inputs using 1 or 0, and the corresponding output is calculated according to the logic gate's operation. **1 represents on, 0 represents off.**

**Modelled Example: Truth table** for an **AND** gate with inputs A and B:

| **Combination** | **Input A** | **Input B** | **Result** |
| --- | --- | --- | --- |
| Combination 1 | 0 | 0 | 0 |
| Combination 2 | 0 | 1 | 0 |
| Combination 3 | 1 | 0 | 0 |
| Combination 4 | 1 | 1 | 1 |

**Truth Table task:
Complete the truth tables to display all possible combinations for an OR and a NOT gate.**

| **Combination** | **Input A** | **Input B** | **Result** |
| --- | --- | --- | --- |
| Combination 1 |  |  |  |
| Combination 2 |  |  |  |
| Combination 3 |  |  |  |
| Combination 4 |  |  |  |

| **Combination** | **Input A** | **Result** |
| --- | --- | --- |
| Combination 1 |  |  |
| Combination 2 |  |  |

**Section 2C: Logic Circuits**

**Key Terms:**

* **Logic Circuit**: An arrangement of logic gates to perform a specific function. This can then be represented via a truth table to show all possible inputs and combinations.
* **Boolean Statement:** An expression made up of variables that represent the output of logic gates, for example, NOT B AND A.

**Modelled Example: The truth table below shows all possible combinations for the circuit’s inputs and logic gates.**



**Logic Circuit Task:
Complete the truth table for the circuit below, the first line has been done for you.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **Y** |
| **0** | **0** | **0** | **1** | **0** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |



**What is a Boolean statement?**A Boolean statement is a rule that checks if something is true or false. In Computer Science, we use Boolean statements to control how circuits or programs behave. Boolean statements use words like AND, OR, and NOT to combine or change conditions.

**Modelled Example and breakdown of a Boolean Statement:**

* A AND B means both A and B must be true.
* A OR B means at least one of A or B must be true.
* NOT C means the opposite of C — if C is true, NOT C is false, and if C is false, NOT C is true.

The Boolean statement **A AND NOT B OR C** means:

A AND NOT B = A must be true **AND** B must be false.

OR C = The whole statement will also be true if C is true, no matter what A and B are.

**Boolean Statement Task:
Create your own Boolean statement and then try to create this as a circuit using logic gates at website:** [**https://logic.ly/demo/**](https://logic.ly/demo/)

**Place a screenshot of your circuit below:**

**Section 3: Data Transmission and Compression**

**Key Terms:**

* **Data Transmission**: The transfer of data between devices.
* **Compression**: Reducing the size of data files to save space and improve transmission speeds.
* **Lossy Compression**: Compression that removes some data permanently, often used in images, video, and audio.
* **Lossless Compression**: Compression that allows the original data to be perfectly reconstructed but marginally reduces the file size.

**Importance of Data Compression**

It is crucial for data to be small when sending over a network (transmission). For instance:

**Streaming:** Services like Netflix and Spotify rely on efficient data transmission to deliver content quickly without buffering.

**Mobile Apps**: Apps need to load quickly and function seamlessly, which is often dependent on the speed and size of data transmitted.

**Computer Games:** In online gaming, rapid data exchange is essential to ensure smooth gameplay—larger files may lead to delays.

**Text Messaging:** SMS relies on small data packets to be sent over mobile networks promptly.

**Compression Techniques**

To make data smaller, there are various compression techniques to use depending on the type of data.

**Images and video can be compressed by reducing colour depth or lowering resolution.**

**Colour Depth:** This refers to the number of bits used to represent the colour of a single pixel. Reducing colour depth decreases the range of colours that can be displayed, thus reducing the overall file size.



**Resolution:** Lowering the number of pixels in an image reduces its resolution, effectively reducing the file size since fewer pixels need to be stored.

**Audio files can be compressed using sample rates or bit depth:**

**Sample Rate:** This determines how many times per second the audio is sampled. A lower sample rate results in smaller files but can lead to a loss in audio quality.

**Bit Depth:** This refers to the number of bits used to represent each audio sample. Reducing bit depth decreases the dynamic range of the audio and reduces the file size as well.



**Modelled Example: Identify a file type and research the compression method it utilises with advantages/disadvantages.**

**Image files:**

* JPEG (lossy, smaller file size, good for photos)
* PNG (lossless, no quality loss, better for graphics)

**Disadvantages:**

* JPEG (quality loss, not suitable for text)
* PNG (larger file size compared to JPEG)

**Compression Task:**

**Research 2 different audio file formats and evaluate which compression method they use and provide advantages and disadvantages for each. There are some suggestions below:**

MP3:
WAV:

**File sizes in images:** Image file sizes are calculated by multiplying width, height, and bits per pixel, then dividing this number by 8 to convert the total bits to bytes. More color combinations mean more bits per pixel, increasing file size.

**File sizes in text:** Text file sizes are calculated by multiplying the number of characters in the text by bits per character and then dividing by 8. Larger character sets like Unicode use more bits, increasing file size.

**File size Task:**

**Calculate the file size of an image with 325 pixels width x325 pixels height and a color depth of 3. Provide the answer in bits and bytes.**

**Section 4: Optimisation in Games**

**Key Terms:**

* **Optimisation**: The process of making a system as effective or functional as possible.
* **Performance**: The efficiency or speed of a game, often influenced by data size and processing power.
* **Graphical Experience**: The visual quality of a game, affected by assets loading times and resolutions.

**Compression and Game Performance:**

Compression can be used in games to selectively **optimise** performance. Developers can reduce the file sizes of textures, sounds, and other assets to ensure faster loading times and smoother gameplay. This might include:

* **Texture Compression:** Reduces the size of image files used in graphics while maintaining visual requirements.
* **Audio Compression:** Smaller audio files result in less strain on processing power, ensuring the game runs more efficiently. Again, this can impact sound quality.
* **Selective Asset Reduction:** Lowering the resolution of distant graphics or background elements can free up resources for more demanding tasks, allowing for enhanced performance but potentially reducing the detail in less focused areas.

**Optimisation Activity:**

Consider a video game you enjoy. Discuss how the developers might have used compression techniques to ensure the game runs smoothly. What compromises in graphical quality might you be willing to accept for better performance?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Section 5: Abstraction and Decomposition**

**Key Terms:**

* **Abstraction** simplifies a complex problem by focusing on the main ideas, removing unnecessary details.
* **Decomposition** breaks down a complex problem into smaller, easier-to-solve parts, making it more manageable and structured.

**Modelled Example for Abstraction:**

The image only includes critical components like the bed, desk, chair, wardrobe, and door. These are essential to understand the layout and purpose of the room.

Complex details, such as textures, materials, colors, and decorative objects, are removed. This makes the image easier to interpret and more efficient for its intended purpose (e.g., planning or designing the layout).

**Modelled Example for Decomposition:**

Consider planning a holiday as a complex problem. Some components could be broken down as follows:

**Problem: Planning a Holiday**

1. Choose Destination
2. Book Flights
3. Find Accommodation
4. Plan Itinerary - Activities for each day - Meals
5. Arrange Transportation

**Activity:**

Think of a real-world challenge like organizing a school event. Break it down into its components, such as location, budget, participants, and timeline.

**Problem:**

1.
2.
3.
4.
5.

**Section 6: Binary and Denary Systems**

**Key Terms**

* **Binary**: A base-2 number system that uses only 0s and 1s.
* **Denary**: A base-10 number system, our everyday counting system.



All data is stored within a computer system using binary.

* A singular binary digit is known as a bit.
* A group of 4 binary digits is known as a nibble.
* A group of 8 binary digits is known as a byte.

Whenever you are using data, whether sending a text message, playing a computer game, watching TV, or using a laptop – you are sending, receiving and processing binary data.

Letters, numbers, symbols and characters all have unique binary codes, when these are received and processed its graphic is then displayed on the screen.

**ASCII:**
ASCII uses 8 bits to represent a character, however 8 bits is limited to 255 combinations. There are more than 255 different characters when we consider all different languages, emojis, and symbols and therefore this is now outdated.

**Unicode:**

Unicode is the new alternative character set and instead uses 16 bits to overcome the limitation of ASCII. With 16 bits used to represent each character, it can store 65,000-character possibilities, which include emojis, specialist symbols and all different languages.

**Conversion Processes:**

At times, we may want to convert a binary number to denary or to hexadecimal.

**Binary to denary:**Using the binary grid, we can convert binary to denary by placing in our values and then adding together any values with a 1 within them.


**Modelled Example for binary to denary:

10111111** = **128 + 32 + 16 + 8 + 4 + 2 + 1 = 191**

**Denary to binary:**

We can convert a denary (decimal) number to binary by subtracting column values from the decimal number. We start from the left and if the value is larger than the required decimal number, we place a 0.

**Modelled Example for denary to binary:**

40 in decimal = **(start from the left!)**

 **128, 64: Too large, place 0s.**

**32: Fits. Subtract 32 from 40 (40 - 32 = 8), place 1.**

**16: Too large, place 0.**

**8: Fits. Subtract 8 (32 + 8 = 40), place 1.**

**Remaining columns: 0s.
40 Decimal = 00101000 Binary.**

**Binary Addition:**

Adding binary numbers together works in a very similar way to adding denary (decimal) numbers together using the column method.

**There are 4 rules to follow:**
0 + 0 = 0
0 + 1 = 1

1 + 1 = 0 and carry a 1 to the left column.

1 + 1 = 1 and carry a 1 to the left column.

*The rules to carry to the left work similarly to decimal when we have 9 + 1 or 99 + 1.*

**Carries**

Modelled Example for binary addition:

**Overflow Errors:**

If a binary value carries over and goes beyond the 8 bits, this is known as an overflow error. This is because it cannot be represented in 8 bits with its maximum storage capacity being 255.

**Binary/Denary/Addition Activity:**

**Activity 1:**

Convert the following **binary values to denary** using the grid below:

1. **01010101 =**
2. **11110000 =**
3. **00001010 =**

**Activity 2:**

Convert the following **denary values to binary** using the grid below:

1. **45 =**
2. **99 =**
3. **242 =**

**Activity 3:**

Add together the following binary values:

1. **00000001 + 11000000 =**
2. **01010101 + 00000010 =**
3. **01011010 + 00011111 =**

**Challenge Activity:**

A computer uses 8 bits to store numbers. What happens if you try to add the binary numbers 11110000 and 10100000?

**Section 7: Binary and Hexadecimal Systems**

**Key Terms**

* **Binary**: A base-2 number system that uses only 0s and 1s.
* **Hexadecimal**: A base-16 number system used in computing to represent binary numbers in a more readable way using numbers and letters.

Hexadecimal values are always in pairs, with each Hexadecimal digit representing 4 bits (a nibble), and a pair representing an 8-bit (byte) binary number.

Hexadecimal is a base-16 system because it uses 16 unique values: 0-9 for numbers and A-F for 10-15. In this system, values above 9 are replaced with letters to simplify representation.

**Hex’s purpose is to make this: 11001100 11110000 10101010 easier to read with: CCF0AA**

**Modelled Example: Binary to Hex Conversion Process**

To convert a binary number to hexadecimal, we start by splitting the byte into two pairs:

00110111 = **0011 0111**

We then place these values into our binary grid, treating them as *separate* numbers.



Once we have placed them into the grid, we then add them together to find the total for each hexadecimal value:


**2 + 1 = 3 4+2+1 = 7**

The answer is then the numbers placed together. In this example, 00110111 = 37 Hexadecimal.

If the value was ever above 9, such as in the example below, we would then use letters to represent the value:


**8 + 4 = 12 or (C in Hex) 4+2+1 = 7**



In this example, 11000111 = C7 Hexadecimal.

**Binary to Hex Activity:**

**Activity 1:**

Convert the following **binary values to Hex** using the grid below to help you:

1. **10101100 =**
2. **11010011 =**
3. **11100011 =**
4. **10101010 =**

|  |
| --- |
| **Nibble 1** |
|  |  |  |  |

|  |
| --- |
| **Nibble 2** |
|  |  |  |  |

**Hex to Denary Conversion Process:**

To convert hexadecimal values to denary, we use a 16 | 1 grid because hexadecimal is based on 16 values (0-F, with F being 15)

We place the values into each part of the grid.

The first digit is multiplied by 16.

The second digit is multiplied by 1.

**Modelled Example: A8 Hexadecimal to denary.**

Step 1. A = 10
Step 2. 10 X 16 = 160.

Step 3. 8 = 8

Step 4. 8 X 1 = 8.

Step 5 160 + 8 = 168.

A8 Hexadecimal = 168 Denary

**Hexadecimal to Denary Activity:**

**Activity 1:**

Convert the following **Hexadecimal values to denary** using the grid to help you:

|  |  |
| --- | --- |
| **Hexadecimal Value 1** | **Hexadecimal Value 2** |
| **16’s** | **1’s** |
|  |  |

1. **1A =**
2. **3F =**
3. **4C =**
4. **7E =**
5. **B2 =**

**Modelled Example: Hexadecimal to Binary Conversion Process:**

Convert a Hexadecimal value to binary is a much simpler process. Firstly, we take each hex digit separately.

Example: A3 = A | 3

We then convert each digit to its 4-bit binary equivalent, remembering that above 9 uses letters.

A = 10. 10 can be represented with 1010

3 = 3. 3 can be represented with 0011

A3 = 1010 0011

**Hexadecimal to Binary Activity:**

**Activity 1:**

Convert the following **Hexadecimal values to Binary** using the grid to help you:

1. **FF =**
2. **7E =**
3. **AB =**
4. **1D =**
5. **6C =**

|  |
| --- |
| **Hexadecimal Value 1 =**  |
| 8 | 4 | 2 | 1 |
|  |  |  |  |

|  |
| --- |
| **Hexadecimal Value 2 =**  |
| 8 | 4 | 2 | 1 |
|  |  |  |  |