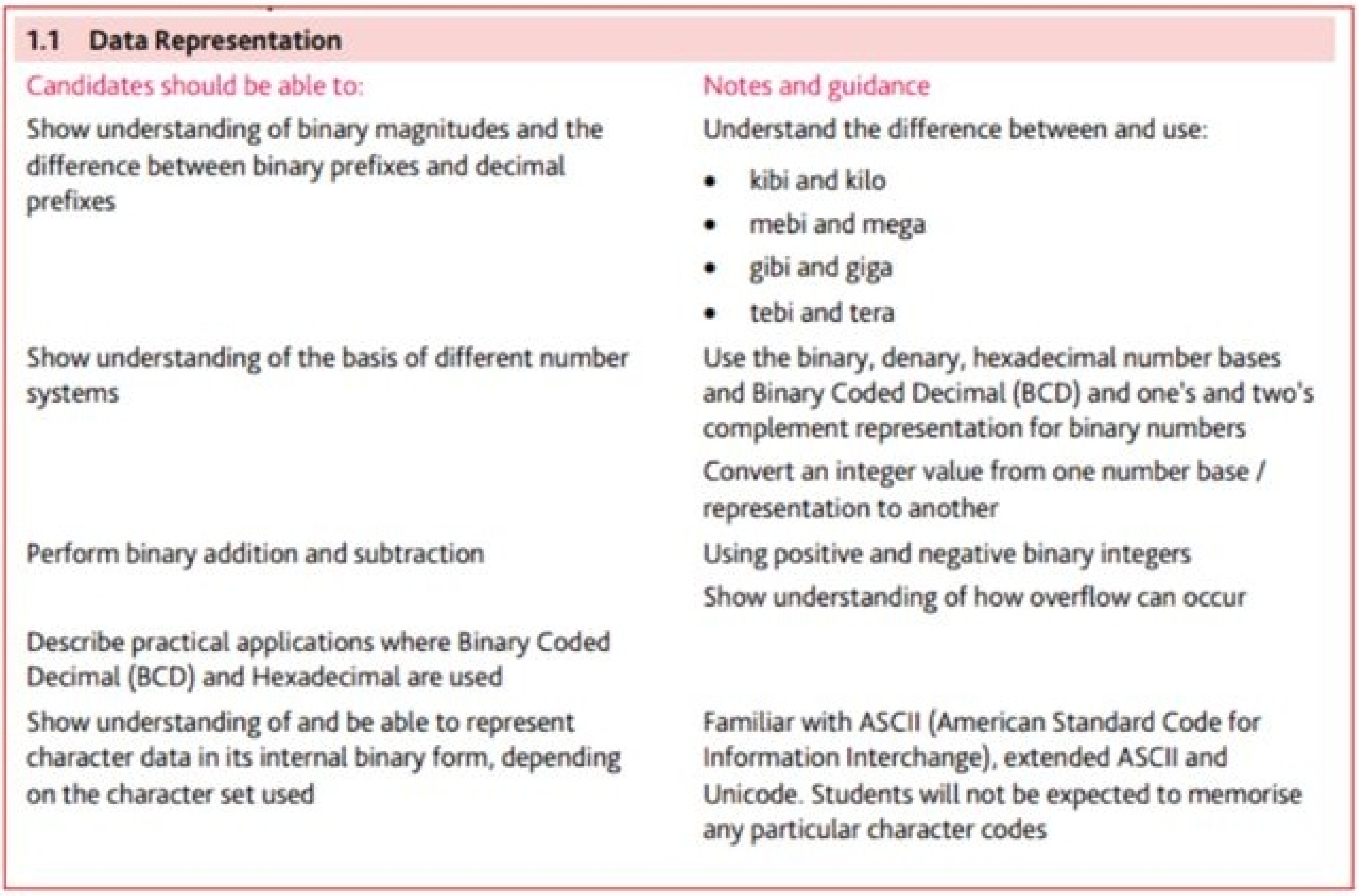
Chapter 1 Information representation

1.1 Data Representation



1.1 Data Representation

Candidates should be able to:	Notes and guidance
Show understanding of binary magnitudes and the difference between binary prefixes and decimal prefixes	Understand the difference between and use: • kibi and kilo • mebi and mega • gibi and giga • tebi and tera
Show understanding of the basis of different number systems	Use the binary, denary, hexadecimal number bases and Binary Coded Decimal (BCD) and one's and two's complement representation for binary numbers Convert an integer value from one number base / representation to another
Perform binary addition and subtraction	Using positive and negative binary integers Show understanding of how overflow can occur
Describe practical applications where Binary Coded Decimal (BCD) and Hexadecimal are used	
Show understanding of and be able to represent character data in its internal binary form, depending on the character set used	Familiar with ASCII (American Standard Code for Information Interchange), extended ASCII and Unicode. Students will not be expected to memorise any particular character codes

Binary | Hexa | Denary

The basis of any number system consists of:

- A base: the number of digits that a number system can use to represent numbers
- Place value for each digit: digits in certain positions have a specific value
- Denary - Base 10 integer digits
- Binary Systems - Base 2
    - Possible bits (binary digits): 0 and 1
    - All data and characters are represented in binary

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	0	0	0	0	0	0	0

E.g. 65 in binary is 0100001

Binary Coded Decimal (BCD)

- Binary representation where each positive denary digit is represented by a sequence of 4 bits (nibble)
- Only certain digits are converted to BCD, because particular digits represent a digit greater than 9.
- Ex. 429 in BCD:
- Convert each digit to their binary equivalents
- 4 = 0100 | 2 = 0010 |9 = 1001
  Concatenate the 3 nibbles (4-bit group) to produce BCD: 0100 0010 1001
- Practical applications
  - A string of digits on any electronic device displaying numbers (e.g. Calculators)
  - Accurately measuring decimal fractions
  - Electronically coding denary numbers

Two's Complement

- We can represent a negative number in binary by making the most significant bit (MSB) a sign bit, which indicates whether the number is positive or negative.
- Converting negative denary into binary (ex. -42)
- Find the binary equivalent of the denary number (ignoring the -ve sign) | 42 = 101010
- Add extra 0 bits before the MSB, to format binary number to 8 bits | 00101010
- Convert binary number to one's complement (flip the bits) | 11010101
- Convert binary number to two's complement (add 1) |1010101 + 1 = 11010110
- Converting binary two's complement into denary (ex. 11010110)
  - Flip all the bits | 00101001
  - Add 1 | 00101010
  - Convert binary to denary and put a –ve sign) | -42
- Range is -127 to 128

Hexadecimal Systems - Base 16

- Possible digits: 0 to 9 and A to F, where A to F represent denary digits 10 to 15
- Practical applications:
  - Defining colors in HTML
  - Defining Media Access Control (MAC) addresses
  - Assembly languages and machine code
  - Debugging via memory dumps
- E.g. A5 in Denary = (16×10) + (1×5) = 165
- E.g. 65 in Hexadecimal = 65÷16=4 Remainder 1∴ = 41

Binary Addition | Subtraction

Binary addition and binary subtraction are two fundamental operations in binary arithmetic, which is the mathematical system used in computers and other digital devices.

Binary Addition:

In binary addition, two binary numbers are added together to produce a third binary number. The process of binary addition is similar to decimal addition, but with only two digits, 0 and 1. Here's an example of how binary addition works:

In this example, we are adding two 8-bit binary numbers: 10110110 and 01011101. We start by adding the least significant bits (LSBs), which are the rightmost digits, and then carry over any remainders to the next column if needed. In this case, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 0 with a carry of 1, and so on.

Once we have added all the bits, we get the final result of 11000100 in binary, which is equivalent to the decimal value of 196.

1 0 1 1 0 1 1 0

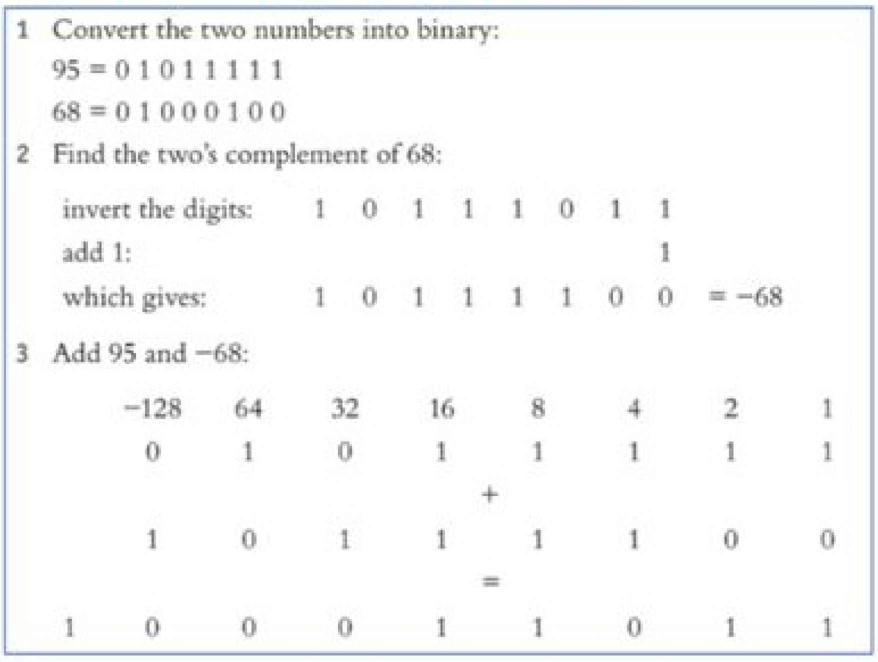
1 1 0 0 0 1 0 0

1 1 0 0 0 1 0 0

Binary Subtraction:

Binary subtraction using the two's complement method is a common technique used in computers to perform subtraction of binary numbers. It works by representing negative numbers using the two's complement representation.

Carry out the subtraction 95 – 68 in binary



1 Convert the two numbers into binary:

95 = 0 1 0 1 1 1 1 1

68 = 0 1 0 0 0 1 0 0

2 Find the two's complement of 68:

invert the digits:	1	0	1	1	1	0	1	1	
add 1:								1	
which gives:	1	0	1	1	1	1	0	0	= −68

3 Add 95 and −68:

	−128	64	32	16	8	4	2	1
	0	1	0	1	1	1	1	1
					+			
	1	0	1	1	1	1	0	0
					=			
1	0	0	0	1	1	0	1	1

Measurement of the size of computer memories

Name of memory size	Equivalent denary value (bytes)
1 kilobyte (1 KB)	1 000
1 megabyte (1 MB)	1 000 000
1 gigabyte (1 GB)	1 000 000 000
1 terabyte (1 TB)	1 000 000 000 000
1 petabyte (1 PB)	1 000 000 000 000 000

Name of memory size	Number of bytes	Equivalent denary value (bytes)
1 kibibyte (1 KiB)	2^{10}	1 024
1 mebibyte (1 MiB)	2^{20}	1 048 576
1 gibibyte (1 GiB)	2^{30}	1 073 741 824
1 tebibyte (1 TiB)	2^{40}	1 099 511 627 776
1 pebibyte (1 PiB)	2^{50}	1 125 899 906 842 624

- **kibibyte/mebibyte/gibibyte/tebibyte/pebibyte** has a **binary** prefix,
- **kilobyte/megabyte/gigabyte/terabyte/petabyte** has a **denary** prefix

ASCII Code and Unicode

Character Set

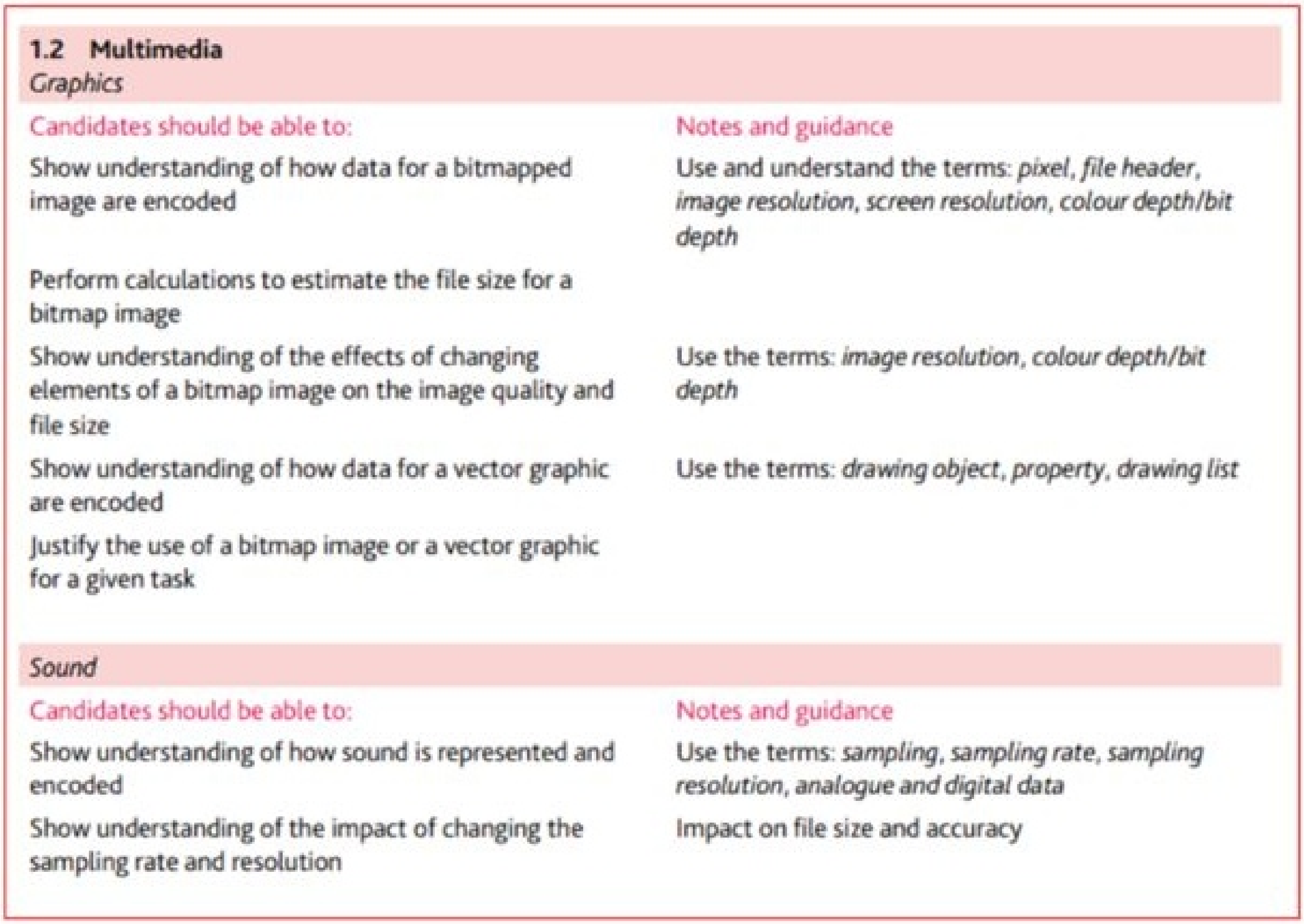
Set of characters defined by the computer hardware and software, so that the computer can understand human characters. Symbols that the computer recognizes. Each character has a unique binary value

ASCII Code

A coding system used to represent all characters on a keyboard, including denary numbers, alphabets, characters and 32 control codes. Standard ASCII is a 7-bit binary code which supports up to 128 characters, and extended ascii is an 8-bit code supporting up to 256 characters.

Issues with ASCII coding system	How Unicode overcomes them
Only supports 128/256 characters	Uses two, three or four bytes per character
Uses 1 byte to represent each character only	
Doesn't support all languages in world	Supports all languages around the world
Extended ASCII is not standardized, there's variation in different ASCII Coding system after first 128 characters	It's standardized First 128 characters are same

1.2 Multimedia



1.2 Multimedia
Graphics

Candidates should be able to:	Notes and guidance
Show understanding of how data for a bitmapped image are encoded	Use and understand the terms: *pixel*, *file header*, *image resolution*, *screen resolution*, *colour depth/bit depth*
Perform calculations to estimate the file size for a bitmap image	
Show understanding of the effects of changing elements of a bitmap image on the image quality and file size	Use the terms: *image resolution*, *colour depth/bit depth*
Show understanding of how data for a vector graphic are encoded	Use the terms: *drawing object*, *property*, *drawing list*
Justify the use of a bitmap image or a vector graphic for a given task	

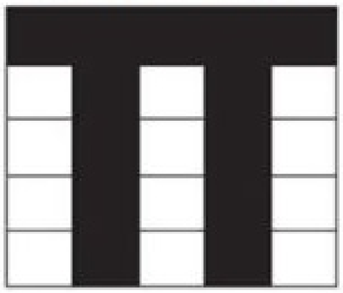
Sound

Candidates should be able to:	Notes and guidance
Show understanding of how sound is represented and encoded	Use the terms: *sampling*, *sampling rate*, *sampling resolution*, *analogue and digital data*
Show understanding of the impact of changing the sampling rate and resolution	Impact on file size and accuracy

Bitmap Image

- A bit map image is made up of a 2D representation of pixels
- Smallest addressable element of a bit map image is a pixel
- Each pixel has a color
- Each pixel has a binary value corresponding to its color
- Number of bits used to represent each pixel is called the bit depth

How can a computer store this bitmap image?



Since it's a monochrome image, each pixel can have one color, so two bit per pixel will be the color depth, black pixel can be represented as 00 and white can be represented as 01, Each pixel will have a color either black or white, each pixel will have a binary value. Bits are stored in sequence.

00000 10101 10101 10101 10101

Pixel : Smallest addressable element of a bitmap image

File Header : Contains meta data of the image

What does file header store?

- Confirmation that file is BMP
- File size
- File location
- Type of compression used
- File name
- Color depth
- Image resolution
- File extension

Image Resolution : number of pixels that make up an image, for example, an image could contain 4096 × 3192 pixels (12738656 pixels in total).

Screen Resolution : number of horizontal and vertical pixels that make up a screen display. If the screen resolution is smaller than the image resolution, the whole image cannot be shown on the screen, or the original image will become lower quality

Color Depth/Bit Depth : number of bits used to represent the smallest unit in, for example, a sound or image file – the larger the bit depth, the better the quality of the sound or color image.

File Size Calculation for Bitmap Image

File Size = (Number of Pixels Horizontal x Number of pixels vertical) × color depth → *Answer would be in bits*
File Size = (Total Pixels) × color depth
Bits → Bytes **(Divide by 8)**
Bytes → Kilobytes/Kibibytes **(Divide by 1000/1024)**
Kilobytes/Kibibytes → Megabyte/Mebibyte **(Divide by 1000/1024)**

Vector image

- A vector image is made up of geometric shapes
- These shapes are joined by 2D points, and coordinates
- Each shape has its own drawing list
- Which stores the commands and instructions on how to the draw the image as well its properties/attributes
- Such as radius, thickness, fill color etc.

Differences between bitmap and vector images

Bitmap Image	Vector Image
Bitmap is made up pixels	Vector graph store a set of instructions about how to draw the same
Bitmap files are usually bigger than vector graphics files, each pixel is stored, which has a binary value. File size greater compared to vector image.	Vector files are usually smaller in size than vector graphics files because only the instructions and commands required to draw the image are stored
Enlarging a bitmap means the image is pixelated	Vector graphic can be enlarged without the image becoming pixelated
Bitmap images can compress, vector graphic images do not compress well	Vector graphics cannot be compressed well
Bitmap images are more suitable for photographs	Vector graphics are more suitable for geometric shapes
Individual elements of bitmap cannot be grouped	Individual elements on vector can be grouped

Advantages of Vector Graphics over bitmap image

- Vector graphics are smaller in size since they only store the commands and instructions on how to draw the image
- Vector graphics can be scaled up and down Static dimensions are not stored, relative positions of drawing objects are stored, position and dimensions are recalculated with adjustment in size

Sound

Sampling

- The analogue value of sound wave measured at set time intervals
- The amplitude of sound wave taken at regular time intervals

Sampling Resolution

- Number of distinct values available to encode the sound wave
- Number of bits used to encode each sample i.e. bit depth
- Higher the sampling resolution is
    - ✓ better the quality
    - ✓ less quantization errors
    - ✓ less distortion in sound.
    - ✓ Greater accuracy of sound
    - ✓ Greater file size as well
    - ✓ Longer to transmit
    - ✓ Greater bandwidth required when uploading
    - ✓ Greater processing power required

Sampling Rate

- Number of samples taken per unit time
- Number of times the analogue value of sound is measured per unit time, per second
- Higher sampling rate results in more accurate representation of sound.

How sampling is used to record sound clops

- Analog value of the sound wave is measured at set time intervals
- An approximation of sound wave is achieved.
- the sample is then encoded into binary values and stored.
- Increasing the sampling rate improves the accuracy of sound.

1.3 Compression

1.3 Compression	
Candidates should be able to:	Notes and guidance
Show understanding of the need for and examples of the use of compression	
Show understanding of lossy and lossless compression and justify the use of a method in a given situation	
Show understanding of how a text file, bitmap image, vector graphic and sound file can be compressed	Including the use of run-length encoding (RLE)

Compression Techniques

Lossless Compression

A compression algorithm is used, the data is not permanently deleted, the original file can be reconstructed back to the original. This type of compression is mostly used in text documents, spreadsheets, code files where it's essential to decompress the file. It operates by identifying repeating bits of data, indexes/collates them and replaces the repeating data with the index value. Examples of compression algorithms based on lossless compression are Run Length Encoding, and Huffman coding. *Include example of RLE on text as well*

Run Length Encoding

- Run Length Encoding is a lossless compression algorithm that operates on repeating data.
- Data is not permanently deleted and can be reconstructed back.
- It's most suitable for files with long run of repeated bits of data
- As the algorithm identifies repeating data, it encodes them into to two values
- The first value represents the number of repetitions
- The second value represents the data, or its index, e.g. ASCII code, color code, or the index with which it's stored in a dictionary
- In some cases, the second value is preceded by a control character as well.

Take example of this text:

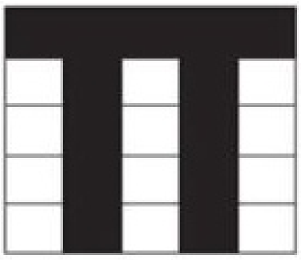
AAAAAA**BBBB**CCCCCCCCCC

The RLE will compress this text in the following way:

6 65 4 66 10 67

Where 65, 66. 67 is the ascii code of the characters

6, 4. 7 are the number of repetitions of the corresponding data.



The algorithm identifies the long run of repeating pixels, and encodes them into two values, one value represents the number of repetitions whereas the second value represents the color code/binary value of color. The following image will be compressed as following

B5 W1 B1 W1 B1 W1 W1 B1 W1 B1 W1 W1 B1 W1 B1 W1 W1 B1 W1 B1 W1

Lossy Compression

A compression algorithm is used, the data is permanently deleted, and the original file cannot be reconstructed back, It decides what part of the file is important, and discards other data that's not important enough. It's usually used to compress audio, and video files; it operates in a way that there's no significant compromise in the quality of the file compressed e.g. a music file, or a movie file. The compression algorithm identifies bits of data that aren't important or not detected by a human, and deletes them. Example, perceptual music shaping used on audio files. Files using lossy compression are JPEG, MP3 etc.

MP3 Compression

Lossy compression algorithm is used, perceptual music shaping, the data is permanently deleted and the original audio file cannot be reconstructed back, the algorithm basically removes sound with low amplitude, if two sounds are played at the same it removes the softer sound. Removes background sound, or sounds not noticed by human ear. Since the algorithm only removes bits of data not identified by a human, it still retains most of the quality of the original file.