Convert the following denary numbers to 8-bit unsigned binary:
1a) 37 00100101
1b) 96 01100000
1c) 219 11011011

Convert the following unsigned binary numbers to denary:
2a) 01100110 102
2b) 10010010 146
2c) 11110000 240

Convert the following denary numbers to hex:
3a) 19 13
3b) 153 99
3c) 999 3E7

Convert the following hex numbers to denary:
4a) A7 167
4b) 9F 159
4c) D3D 3389

Convert the following binary numbers to hex:
5a) 1100 0101 C5
5b) 0111 1011 0011 7B3
5c) 1111 0000 0000 1101 F00D

Convert the following hex numbers to binary:
6a) 8E 1000 1110
6b) 6B 0110 1011
6C) CAB 1100 1010 1011

7a) A person hosts a torrent file and is seeding (sending) the file at 3MB/s on average - how many bits would they transfer in 1 year - give your answer in a sensible unit?
756.9Tb = (3 * 8 * 1,000,000 * 60 * 60 * 24 * 365) / (1,000,000,000,000)

7b) An image is taken with a resolution of 1280x720px and a bit depth of 32 - how many Mebibytes would this image be?
3.52MiB = $(1280 * 720 * 32) / (8 * 2^{20})$

7c) By how much would the size of this image reduce if you reduced the bit depth to 8?
By 4x - i.e. new size = 0.88MiB

7d) A song is recorded with 2 channels at a sampling frequency of 44.1kHz and a sampling resolution of 24 bits - how many mebibits would 3 minutes of audio use?
363.4Mib = $(2 * 44100 * 24 * 3 * 60) / (2^{20})$

7e) Each record in a database requires 200 bytes - a company knows there will never be more than 100 million records. They plan to buy a 32GB USB to store a backup of their database - will the USB have enough capacity to store a copy of a database if it reaches 100 million records?
Yes - (200 * 100,000,000) / (1,000,000,000) = 20GB

8a) Add an appropriate base & power in the form $b^n = x$ as well as the regular number for how many bytes each represent
kilobytes = $10^3$ = 1,000
megabytes = $10^6$ = 1,000,000
gigabytes = $10^9$ = 1,000,000,000
terabytes = $10^{12}$ = 1,000,000,000,000

8b) Add an appropriate base & power in the form $b^n = x$, the correct multiple of 1024 the number represents, as well as the regular number for how many bytes each represent
kibibytes = $2^{10}$ = 1024 = $1024^1$
mebibytes = $2^{20}$ = 1,048,576 = $1024^2$
gibibytes = $2^{30}$ = 1,073,741,824 = $1024^3$
tebibytes = $2^{40}$ = 1,099,511,627,776 = $1024^4$

Write the following denary numbers in 8-bit signed magnitude:
9a) 47 0010 1111
9b) -75 1100 1011

Write the following denary numbers in 8-bit one's complement:
10a) (use unsigned) 142 0111 0001
10b) (use signed) -53 0011 0100

Write the following denary numbers in 8-bit two's complement:
11a) 125 0111 1101
11b) -96 1010 0000

Write the following denary number in packed binary-coded decimal - how would it be different if using unpacked BCD?
12a) 592 0101 1001 0010 - if using packed BCD, each digit would have 4 trailing 0's - i.e. 00000101 00001001 00000010

Write the following packed binary-coded decimal number in denary:
13a) 1001 0111 0101 0011 0001 97531

14) What are two problems with both signed magnitude & one's complement?
Two representations of 0 (+ & -) and arithmetic doesn't work

15) How does two's complement solve the issues mentioned in question 14?
Only 1 representation of zero - two's complement is literally one's complement and add 1 - then the arithmetic works

16) Write the two's complement number for zero - then use a method (e.g. "flip the bits & add 1") to try and convert zero to negative - what happens?
0000 --> 1111 + 0001 --> (1)0000 - since overflow is ignored, even if we try to force 0 to be either negative or positive, it simply goes back to settling on the one value of 0 (0000)

17) Complete the table (assume numbers are stored using 1 byte):

| Number | Denary value | Bit pattern |
|---|---|---|
| Smallest signed magnitude value | -127 | 11111111 |
| Biggest signed magnitude | 127 | 01111111 |
| Smallest one's complement | -127 | 10000000 |
| Biggest one's complement | 127 | 01111111 |
| Smallest two's complement | -128 | 10000000 |
| Biggest two's complement | 127 | 01111111 |

Perform the following sums using 8-bit unsigned binary integers & comment on the correctness of the answer:
18a) 100 + 40     01100100 + 00101000 = 10001100 (correct)
18b) 154 + 125   10011010 + 01111101 = 00010111 (incorrect due to overflow)

Perform the following sums using signed binary integers & comment on the correctness of the answer:
19a) 95 + 52       01011111 + 00110100 = 10010011 (incorrect due to overflow)
19b) -37 + 83      11011011 + 01010011 = 00101110 (correct)

20) What is overflow when performing arithmetic?

When the answer can't be accurately represented with the given number of bits - i.e. adding two positive numbers and the answer becomes negative, or adding two negative number and the answer becomes positive

Perform the following arithmetic operations using BCD:

21a) 7.8 + 4.5

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| | | | | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| | | | | | | | | 1 | 1 | 0 | 1 |
| | | | | | | | | | 1 | 1 | |
| | | | | | | | 1 | 0 | 0 | 1 | 1 |
| | | | | 1 | 0 | 1 | 1 | | | | |
| | | | | | 1 | 1 | | | | | |
| | | | 1 | 0 | 0 | 1 | 0 | | | | |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | | | | 2 | | | | 3 | | | |

21b) 12.7 + 3.4

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| | | | | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| | | | | | | | | 1 | 0 | 1 | 1 |
| | | | | | | | | | 1 | 1 | |
| | | | | | | | 1 | 0 | 0 | 0 | 1 |
| | | | | 0 | 1 | 0 | 1 | | | | |
| | | | | | | | 1 | | | | |
| | | | | 0 | 1 | 1 | 0 | | | | |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | | | | 6 | | | | 1 | | | |

**Without performing the calculations in binary**, write “y” or “n” for whether they would result in overflow (assume using 8-bit two’s complement both operands & answer):

22a) 63 + 64 N
22b) 64 + 64 Y
22c) -100 + -30 Y
22d) -64 + -64 N

Extra (taken from the June 2022 AS exam):

| Statement | Answer |
|---|---|
| The hexadecimal value 11 represented in denary | 17 |
| The smallest denary number that can be represented by an unsigned 8-bit binary integer | 0 |
| The denary number 87 represented in Binary Coded Decimal (BCD) | 1000 0111 |
| The denary number 240 represented in hexadecimal | F0 |
| The denary number -20 represented in 8-bit two’s complement | 11101100 |

---

23) What are the two main character sets & how many bits do each use? Why was the second character set created?

ASCII (7 or 8 bits) & Unicode (8-32 bits). Unicode was created since ASCII wasn't able to represent all the characters (different languages, symbols, emojis etc). Standard/Extended ASCII can represent 128 and 256 different characters respectively - Unicode can represent more than 1 million.

24) Complete the table:

| Character | ASCII Value in Denary |
|---|---|
| 0 | 48 |
| A | 65 |
| a | 97 |
| 5 | 53 |
| G | 71 |
| z | 122 |

25) What are the first 32 (0-31) characters in ASCII called - what kind of things are they used for?
Non-printable characters - used for control, data transmission, formatting etc

26) Why does saving "你好" use more bytes than saving "Hi"?
"Hi" can be represented using ASCII (i.e. 2 bytes total) - while "你好" has to be represented with Unicode, which requires more bits (24 bits per character, or 6 bytes total in this case)

27) What are the two categories of images - can you list some file formats of each?
Bitmap (jpg, gif, png, tiff, bmp, webp etc)
Vectors (svg)

28) Define the following terms:
Pixel: a "picture element" - the smallest area of a screen or image. Each pixel has a colour value and an image is comprised of many pixels
Resolution: the number of pixels in a screen/image/video - represented in terms of the number of horizontal & vertical pixels - e.g. 1920x1080

Colour/bit depth: the number of bits used to represent each pixel - a higher value means more distinct colours can be represented
Bitmap: a picture made up of pixels with a defined resolution - each pixel has a specified colour
Vector: an image defined in terms of mathematical properties/shapes/elements, attributes (position, width, height, radius, fill, stroke etc) and their values (e.g. "20px", "red" etc)

29) What is the formula for calculating the file size of an image?
resolution * colour depth

30) Fill in the blank: colour codes are often written in hexadecimal format to make it easy for humans to understand

31) What is a file header? And, more specifically, what data would be contained an image's file header (i.e. an image header)?
Metadata about a file - for an image, this could contain the resolution, colour depth, compression algorithms used, author's name, camera model, geolocation etc

32) An image is 500x500px with a bit depth of 8 - how many different colours could this image contain and what would the file size be?
2 ^ 8 = 256 colours. File size = (500 * 500 * 8) / (1000000 * 8) = 0.25MB

33) A 1920x1080px image has a file size of approximately 5.93MiB - what bit depth do you think was used?
(5.93 * 1024 * 1024 * 8) / (1920 * 1080) = 24

34) Define the following terms when talking about vector images:
Element: an individual shape/component of the image - rect, ellipse, text etc
Drawing list: the collection of all the elements in the image
Attribute/property: a specific feature of the element - width, height, radius, fill/stroke colour etc
Value: a specific value for an attribute - e.g. a width of "20px", a fill of "red" etc

35) Complete the table

| Bitmaps | | Vectors | |
|---|---|---|---|
| Advantages | Use cases | Advantages | Use cases |
| High detail - each pixel can be a different colour<br><br>Compatibility - e.g. jpgs/pngs can be rendered/processed by more software than vectors<br><br>Easier to produce complex effects in tools like Photoshop etc | Photos, posters, complex graphics | Scalable (bigger or smaller) without increasing file size or losing quality/graphic becoming pixelated<br><br>Can be smaller in certain situations - e.g. images with simple shapes like many logos<br><br>Editability - since vectors are code, properties can be animated e.g. in websites. In contrast, bitmap pixel values are pre-generated | Graphics with simple shapes like logos, text, large graphics where bitmaps would have a huge file size (e.g. billboards) |

36) Define the terms:

Sample: a measurement of the amplitude (volume) of the sound at a given time

Sampling resolution/bit depth: the number of bits used to store each sample - higher sampling resolution = less quantisation error

Sampling rate/frequency: the number of samples taken per second - higher rate = less quantisation error

Channels: the sound measured at a specific point in space - e.g. an audio file might be recorded with a microphone on both the left and right to create a stereo file. The number of channels could be more than 2 as well - e.g. if you have 8 speakers in different parts of the stage for a music concert, you could have 8 channels with different music/effects, with each channel being output by a different speaker

37) Briefly explain how sound is represented in a computer

Analogue sound waves will be converted to a digital value by an analogue to digital converter (ADC) - each time the amplitude (volume) of the sound is measured and stored, this is referred to as a sample. Each sample uses a given number of bits which is termed the sampling resolution/bit depth, while the number of samples taken per second is called the sampling rate/frequency. Increasing either the sampling resolution or frequency will reduce the quantisation error - i.e. ensuring the sampled digital values will be able to more accurately reconstruct a sound wave close to the original analogue wave

38) What is quantisation?
The process of rounding/truncating each sample to the nearest value that can be stored by the bit values we have available defined by our sampling resolution - e.g. assume we can store decibel values in 0.1 intervals like 25.7, 25.8 ect - if the real analogue volume at this time was 25.74 decibels, this would hence be rounded and stored as 25.7dB. Our quantisation error in this case would therefore be 0.04 - the difference between the original analogue amplitude and the quantised value that we stored digitally

39) What happens to the sound wave we can reproduce & file size when the sampling rate & sampling resolution are increased?
Since we are storing more samples and using more bits per sample (i.e. reducing the gap between quantisation levels), the digital values we are storing will more accurately be able to represent the original analogue values. However, the file size will increase, since we're taking more samples and using more bits for each sample

40) Suppose we want to store 1 million distinct amplitudes - what would be the minimum sampling bit depth we'd need to use?
log2(1000000) = 19.93 - then round up to 20

41) What does Nyquist's thereom state about the sampling rate and hence what is usually consider the lowest sampling frequency you should use for audio designed to be use for human hearing
That in order to be able to accurately reproduce the original analogue signal, we should sample at at least twice the rate of the highest frequency - i.e. if the highest frequency most humans can hear is 20,000Hz, then we should sample at at least 40,000Hz - hence why just above that 44.1KHz is a common sampling rate

42) What is the formula for calculating the file size of an audio file?
sampling rate * sampling resolution * number of channels * duration

43) A singer records a 60 minute album with 2 channels, at a sampling rate of 44.1kHz and a sampling resolution of 24 - what would be the file size in Mebibytes?
908.43MiB (60 * 60 * 24 * 44100) / (8 * 1024 ^ 2)

44) A CD has a capacity of 700MB and will be used to store interviews - only 1 channel will be required, with a bit depth of 8 and a sampling frequency of 44,100Hz - how many minutes of audio will the CD be able to store?
33 minutes (((700 * 1000000) / (8 * 44100)) / 60) and 4 seconds (((700 * 1000000) / (8 * 44100)) % 60)

45) progressive encoding displays the entire video frame at once, while interlaced encoding alternates between displaying only the even rows, then only the odd rows

46) Define the terms lossy & lossless compression:
Lossy: reduces the file size, but when decompressing, we can’t restore the original exactly - i.e. some data has been lost
Lossless: reduces the file size less, but we are able to restore the original file perfectly without data loss

47) Complete the table:

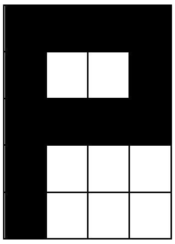
| | Lossy compression technique(s) | Lossless compression technique(s) |
|---|---|---|
| Images | Reduce resolution or colour/bit-depth | Run-length encoding |
| Audio | Reduce sampling rate, sampling resolution or number of channels<br><br>Perceptual music shaping - remove quiet sounds and those with frequencies outside range of human hearing | Huffman coding |
| Video | Image & audio lossy techniques<br><br>Reduce frame rate | Image & audio lossless techniques<br><br>Store only changed pixels, not whole frame |
| Text | [generally we don’t want to lose | Huffman coding |

| | | |
|---|---|---|
| | data when compressing text…since it would be unintelligible] | LZW coding |

48) What are 3 benefits of compression?
Use less storage on disk
Faster to send
Uses less bandwidth
Can reduce file size so it is below file size limit (e.g. 25MB for many email attachments)

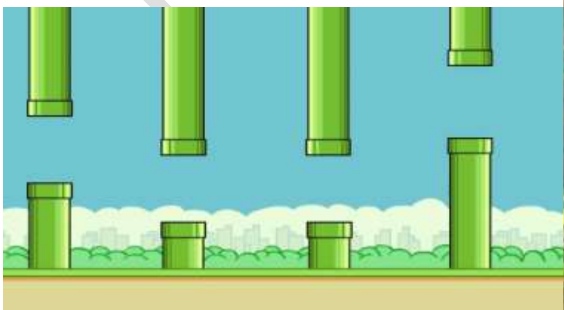
49) Briefly explain how run-length encoding works and calculate the run-length encoded representation of the following black and white 5x4px image (you can write your answer in denary):

Run-length encoding calculates the number of consecutive pixels of the same colour, then rather than storing each pixel value individually, it stores a count of the number of same-colour consecutive pixels, then the colour itself - for example, in the following black and white image, we can represent black as 0 and white as 1:



5 0 2 1 6 0 3 1 1 0 3 1

50) Can run-length encoding be used effectively on photos? Between the two images, which would RLE before more effective for and why?




Run-length encoding would be more effective for the first image (Flappy Bird), since this has many consecutive pixels that are identical in colour the sky, ground, parts of the tubes etc) - in contrast, the photo of the sweets, almost no consecutive pixels will be the identical; even look at e.g. those red sweets - consecutive pixels might look similar to the human eye, but with a typical 24-bit photo being able to store more than 16 million different colours, it’s unlikely that thee shades of red will be completely the same

51) Briefly explain how Huffman Coding works

A frequency table is made representing how many times each character appears in the text - the most common characters are given the shortest Huffman Codes. These codes are used instead of the ASCII values and are generated in such a way that there is only one possible decoding.

52) Suppose you have the following Huffman dictionary and encoded sequence of bits - what is the original string?

11101011111000001010110110011101100

Compression

| | |
|---|---|
| o | 101 |
| p | 000 |
| r | 001 |
| e | 010 |
| s | 110 |
| i | 011 |
| n | 100 |
| C | 1110 |
| m | 1111 |