Mark scheme

**Q1.**

(a)  **Marks are for AO1 (knowledge)**

A;

**R.** More than one lozenge shaded

**1**

(b)  **Marks are for AO1 (knowledge)**

C;

**R.** More than one lozenge shaded

**1**

**[2]**

**Q2.**

(a)  **Mark is for AO1 (understanding)**

****

**R**. more than one lozenge shaded

**1**

(b)  **Mark is for AO1 (understanding)**

****

**R**. more than one lozenge shaded

**1**

(c)  **Mark is for AO1 (understanding)**

****

**R**. more than one lozenge shaded

**1**

**[3]**

**Q3.**

(a)     **Mark is for AO2 (apply)**

39;

**A.** #39

**1**

(b)     **Mark is for AO1 (understanding)**

More compact when displayed;

Easier (for people) to understand/remember; **A.** read

Lower likelihood of an error when typing in data;

Saves (the programmer) time writing/typing in data;

**NE** takes up less space

**R.** if answer states that hexadecimal uses less memory/storage

**Max 1**

**1**

(c)     **Marks are for AO2 (apply)**

3 9/16

//

3.5625

**Mark as follows:**

**1 mark** for correct integer part (3)

**1 mark** for correct fractional part (9/16 or .5625)

**Alternative answer**

57/16;;

**2**

(d)     **Mark is for AO2 (apply)**

57;

**1**

(e)     **Mark is for AO1 (understanding)**

9;

**I.** Quotes around answer

**I.** subscript 10 after the answer 9

**1**

(f)     **Marks are for AO1 (understanding)**

The number of 1s (in the other 7 bits) has been counted // there are four 1s (in the 7 bits);;

there are an even number of 1s so the parity bit has been set to 0 (to keep the number of 1s even);

**Alternative answer**

The 7 data bits have been XORed;

The result is a 0 so the parity bit has been set to 0 (so the result of XORing the 8 bits will be 0);

**2**

(g)     **1 mark is for AO1 (knowledge) and 1 mark for AO1 (understanding)**

**AO1 knowledge – 1 mark:**

Each bit is sent multiple times; **A.** A specified (odd) number greater than 2, instead of multiple

**Marking guidance – to get this mark sent/sender must be clear**

**AO1 understanding – 1 mark:**

The receiver checks the bits it has received and if they are not all the same it assumes the one it received the most copies of is the correct value for the bit; **R.** receiver knows that the bit is correct **A.** receiver takes as correct (or similar)

**Marking guidance – to get this mark received/receiver must be clear**

**A. alternative answer using majority voting with a whole byte instead of individual bits**

**AO1 knowledge – 1 mark:**

The bit pattern (**R.** data) is sent multiple times; **A.** A specified number greater than 2, instead of multiple

**Marking guidance – to get this mark sent/sender must be clear**

**AO1 understanding – 1 mark:**

The receiver checks the bit patterns (**R.** data) it has received and if they are not all the same it assumes the one it received the most copies of is the bit pattern (**R.** data) that was sent; **R.** receiver knows that the bit pattern is correct **A.** receiver takes as correct (or similar)

**Marking guidance – to get this mark received/receiver must be clear**

**2**

**[10]**

**Q4.**

(a)  **Marks are for AO1 (understanding)**

|  |  |
| --- | --- |
| **Quantity** | **Position** |
| 3 kilobytes | 3 |
| 2 mebibytes | 5 |
| 2 bytes | 1 |
| 2 megabytes | 4 |
| 20 bits | 2 |

**Mark as follows:**

**1 mark** for bits, bytes and kilobytes in correct positions

**1 mark** for mebibytes and megabytes in correct positions

**2**

(b)  **Marks are for AO2 (apply)**

**1 mark** for correct conversions between representations, allowing follow through for final answer.

2716 = 0010 01112

C916 = 1100 10012

Final answer: F016

**1 mark** for binary addition 111100002 allowing follow through if conversion was incorrect.

**2**

**[4]**

**Q5.**

**Mark is for AO2 (analyse)**

10011.011;

**1 mark** for fixed point clearly between 5th and 6th digits.

**[1]**

**Q6.**

(a)     (i)      54;

**1**

(b)     (i)      ‘4’ / 4 ; ;

*1 mark for ASCII value 52; 2 marks for correct character 4 ; ;*

**Max 2**

(ii)     UNICODE / EBCDIC / EBCD /extended binary coded decimal ;

**A** minor misspelling of EBCDIC

**1**

(c)     (i)      Bit-mapped graphic;

**R** as pixels  
**R** jpeg *etc*

**1**

(ii)     Image broken down into separate pixels;

Each pixel is either black or white / on or off;

Use 2 different values for black and white / 1 for black and 0 for white (or vice versa);

Store in one bit / bits / byte of computer memory;

**A** diagram which maps onto above points

**A** follow through from (i) a .gif or .jpeg image:

**Max 2**

**[7]**

**Q7.**

(a)     167;;

If final answer is incorrect **Max 1** can be awarded for some correct working out being shown by the candidate:

1010 0111;  
10 \* 16 // 160 // A \* 16;  
A = 10;  
Multiplying a value by 16 and adding on 7;

**2**

(b)     0111.1010 // 01111010

**Mark as follows:**4 bits before binary point are 0111;  
4 bits after binary point are 1010;

**2**

(c)     1;110 1110;

**R** if not 8 bits

**2**

(d)     127;

**1**

(e)     The number to subtract is converted into a negative number;  
**NE** Convert into twoߣs complement  
This is then added to the first number;

Two marks for example:

23 = 00010111  
-48 = 11010000;  
          -------------  
          11100111; (= -25)

**A** if not used 8 bits in examples  
**A** 23 + - 48 is worth 1 mark only (if there is no description)

**Note:** for the first mark in the example to be awarded the two bit patterns must be correct. For the second mark in the example accept an incorrect answer as long as it is a correct addition using one of the two correct bit patterns.

**4**

**[11]**

**Q8.**

(a)     The number of pixels/dots; per cm/inch/unit of measurement;

//

Width × height; in pixels;

**2**

(b)     24 // 16;

**1**

(c)     512;;; //

16\*16;\*16;÷8; //

4096;; ÷8;

**MAX 2** if final answer not correct

**3**

(d)     (For geometric images) less storage space /memory likely to be needed;

**NE.** less space

(For geometric images) will load faster from secondary storage;

(For geometric images) will download faster;

Can be scaled/resized/zoomed without distortion // can be scaled/resized/zoomed without loss of quality;

Image can be (more easily) searched for particular objects;

Can (more easily) manipulate individual objects in an image;

Can preserve the background so that it can be recreated if an object is deleted;

**MAX 2**

**2**

**[8]**

**Q9.**

(a)     16 (bit);

**A** 2 bytes

**1**

(b)     8,800,000 // 100 \* 2 \* 44,000;;;  
//  
100;  
2; **A** 16 ÷ 8; **A** different value for the sampling resolution (16) being used in the calculation but only if matches answer to part 15 44,000;

**Max 2** if final answer incorrect

**3**

(c)     Because of Nyquist's theorem // Because we should sample at least double the highest frequency in the original sound;  
Some people can hear higher frequencies than the average (so more than double has been chosen);  
There is no need to sample at a higher rate as humans won't notice any difference in quality above this level // sampling at a lower rate would mean that some people would notice the lower quality of the recording // sampling at a lower rate would mean that some meaningful changes in the analogue signal could be missed;  
higher rate would require more, unnecessary, storage space;

**Max 2**

(d)     Compression has been used;

**A** Explanation of a particular compression method that could have been used on the recording e.g. lower sampling frequency used // lower sampling resolution used;

**1**

**[7]**

**Q10.**

**2 marks for AO1 (knowledge) and 2 marks for AO1 (understanding)**

**AO1 (knowledge): Representation (MAX 2):**

Music represented as sequence of MIDI (event) messages;

**A**. Music represented as sequence of instructions

**R**. Music represented as sequence of notes

One example of data that might be contained in a message:

•   Channel

•   Note on / note off

•   Pitch / frequency / note number

•   Volume / loudness

•   Velocity

•   Key pressure / aftertouch

•   Duration / length

•   Timbre

•   Instrument

•   Pedal effects

•   Pitch bend

•   Note envelope;

MIDI messages are usually two or three bytes long;

First byte of each MIDI message is a status byte (others are data bytes);

Bit rate is 31,250 bits per second;

MSB value of 1 indicates status byte, 0 indicates data bytes;

Status bytes are divided into a command and a channel number (4 bits for each);

Sixteen channels are supported;

**AO1 (understanding): Advantages of MIDI (MAX 2):**

More compact representation;

Easy to modify / edit notes // Easy to change values eg octave for entire score //easy to change instruments;

Simple method to compose algorithmically;

Musical score can be generated directly from a MIDI file;

No data lost about musical notes // through sampling; **A**. “better quality” but only if there is some explanation of this eg “no error introduced during sampling”, “no background noise recorded”

**[4]**

**Q11.**

**Mark is for AO2 (apply)**

KAITLEN;

**I**. Case

**[1]**

**Q12.**

**Marks are for AO2 (apply)**

**1 mark** for identifying 1001000 1001111 1000111 as the binary representation of ‘HOG’

**1 mark** for final result being 21 bits long;

**R.** if result is the same as HOG (1001000 1001111 1000111) or SON (1010011 1001111 1001110)

**1 mark** for correct application of XOR;

  0011011 0000000 0001001

**A.** follow through mistakes

**[3]**