ASSESSMENT REPORT 2020

ITC315118 – COMPUTER SCIENCE

SECTION A – CRITERION 1

QUESTION 1

(a) if number_of_items not equal to 1

(b) 6 was the preferred answer, 19 or 7 were also acceptable

(c) Change: 20. if transaction = "add"
or swap lines 21 and 23.

(d) 25.1 transaction = "add"
25.2 number_of_items = 1

Examiner Comments

1 (a) Less than half the students got this question correct.

1 (b) The choice of line 6 follows the same style as question 1(a) and was preferred, but the other two options could also work with different programming styles.

1 (c) This question was reasonably well answered, however some students seemed to not understand what a 'logic' error was and tried to propose syntax changes.

1 (d) Most students got the lines to be added correct, but in some cases the location was not provided or was wrong. Inserting the lines in the range of lines 18-23 will produce the wrong results, and after line 26 will operate correctly but will not update the display.

QUESTION 2

(a) 9. if count is greater than 12

(b) Between lines 15 and 18, inside the 'if' clause on line 15

(c) 17.1 else if guess greater than correct
17.2 display "Too High"
17.3 else
17.4 display "Too Low"

Examiner Comments

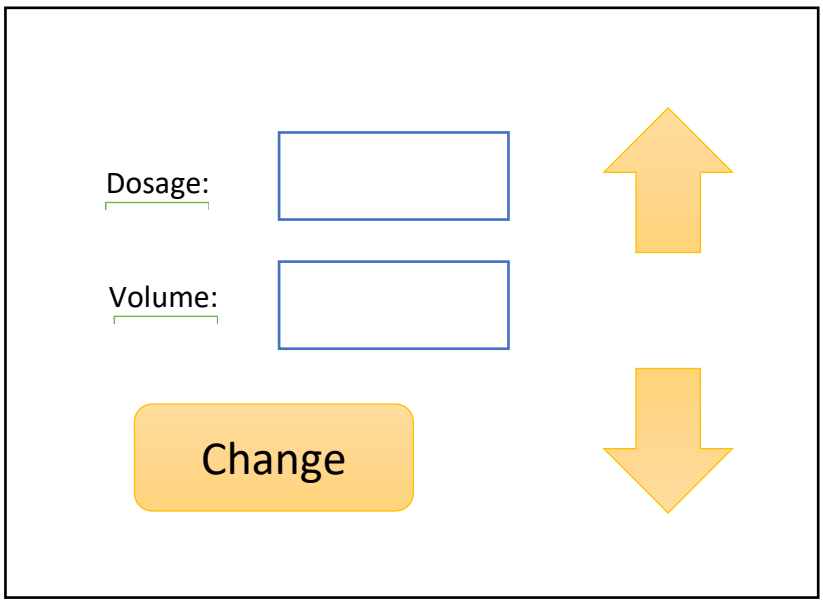
2 (a) This question was well answered.

2 (b) This question was well answered.

3 (b) This question was reasonably well answered in general. Some students perceived line18 as having an indentation error and placed their code after this line which was accepted. Common mistakes were placing the code in the range of lines 23-30 which would effectively give users free guesses, and not structuring their 'if/else' statement in a way that printed one of the messages even if the guess was correct.

Question 3

(a)



(b)

```
1.    Initially:
2.        set dosage to 0
3.        set volume to 1000
4.        set dispensing to false
5.        display dosage
6.
7.    When the "Up" button is pressed:
8.        if dosage > 25 then
9.                set dosage to 30
10.       else
11.               set dosage to dosage + 5
12.       endif
13.       display dosage
14.
15.   When the "Down" button is pressed:
16.       if dosage < 5 then
17.               set dosage to 0
18.       else
19.               set dosage to dosage – 5
20.       endif
21.       display dosage
22.
23.   When the "Change" button is pressed:
24.       if volume < 100 then
25.               enable beeper
26.       endif
27.       if dispensing = false then
28.               set dispensing to true
29.       else
30.               set dispensing to false
31.       endif
```

Examiner Comments

3 (a) Most students who attempted this question had an acceptable answer (note that the volume field wasn't strictly required). Some students added the speaker which isn't appropriate for an applet screen, but were not penalised for this. Some students who otherwise did well in this section appeared to have missed answering this question. Students are advised to check carefully that they have answered every question.

3 (b) There were multiple possible solutions to this question and most students who attempted this question had a reasonable answer, but often incomplete. Common missing elements were: limiting the dosage, updating the dosage display, and handling the toggling of the dispensing state.

SECTION B - CRITERION 2

Question 4

(a)

(i)	1.25	20	1	20.0
(ii)	12.0	13.0	14.0	15.0
(iii)	3	6	9	13

(b) (i) Value of ***f***: 6
Explanation: bodmas 15/5 first = 3, then left to right -4 + 3 -> -1 + 7 = 6 //

(ii) Value of ***j***: remainder after dividing 56 by 5 is 1.

(c) **Trace:**

x	i
1	0
1	1
1	2
3	3
	4

Final value of x is: 3

Examiner Comments

Question 4 (a)

Part (a) was generally well answered with most students getting full marks. One of the common mistakes was in question (a)(i) where some students circled 20 instead of the correct answer which was 20.0 when the code clearly showed ''double y''

Question 4 (b) (i)

Part (b)(i) was very well done with most students remembering order of operations and getting full marks for this.

Question 4 (b) (ii)

Part (b)(ii) was well answered with most students getting full marks. Those who got this wrong seemed to forget that % means the remainder.

Question 4 (b) (iii)

This was another well answered question with most students able to trace the code and find the value of the variable x. A few students forgot to end the for loop when the x value was greater than 3.

Question 5

(a) (i) Value of f: 24.0
Explanation: f (9.0) is greater than 8.0, true branch leads to f = f - 1.0 (8.0).
f (8.0) is less than or equal to 8.0 true branch leads to f = f * 3.0 so f becomes 24.0

(b)

place	mult	i	data[i]	data[place]
5	0	1	2	4
		2	4	
	1	3	2	
		4	4	
	2	5	4	
	3	6		

(c)

m	0	1	2	3
0	0	4	5	6
1	3	2	5	6
2	3	4	4	6

Examiner Comments

Question 5 (a)

This question was generally well answered with the only major fault being that students gave 24 as the answer rather than 24.0. The variable f is clearly a double.

Question 5 (b)

This was also well done. A common error was to have data[place]=5 rather than the 5th value in the array. The condition data[i] = data[4] is clearly true on 3 occasions so mult is incremented 3 times.

Question 5 (c)

This was also well done by those who attempted it. There was no pattern to the errors students made. You just need to be careful as you trace code.

Question 6

(a) Before: 20 12 23 17 7 8 10 2 1 0
After: 20 12 0 2 10 8 7 17 1 23

(b) Line 7 – delete result[ ] - not required (it isn't essential to remove this line)
Line 9 replace int [ ] with void
Line 27 delete "return arr" - not required
Line 40 remove "result =" - should just be separate_odd_even(nums)
Line 41 display (result) is changed to display(nums)

(c) int arr[ ] is an object and objects are passed by reference. Hence any changes made to the object within the method will be permanent once the method call has been completed.

Examiner Comments

This question was only attempted by a limited number of students with mixed success.

Question 6 (a)

Part (a) was answered correctly by half the students who attempted the question. Students received some credit for identifying that the even numbers went to the left and odds to the right. Putting them in order was a good guess but not correct.

Question 6 (b)

Part (b) was poorly done with few students realising what "making the method Void" meant. Clearly, returning the array is not required as objects are passed by reference and any changes made inside the method are reflected in the array passed to the method.

Question 6 (c)

Part (c) was well answered by the students who worked out what the question was asking.

Question 7

(a) (i) Answer is 103
(ii) x

(b) (i) TextA = pear
TextB = grape

(ii) TextA = KIWI FRUIT
TextB = kiwi fruit

(c) (i) "sat on "
(ii) "How much longer=15"
(iii) first value of string3 ""
second value of string3 "ter"
third value of string3 "terjohn"

Examiner Comments

Question 7(a)

(i) Many candidates were unable to produce the correct answer for this question; many mistakenly evaluated 10+3 to yield "Answer is 13".
(ii) Majority of candidates chose the correct response.

Question 7(b)

(i) Most candidates were able to choose the correct response.
(ii) Most candidates were able to choose the correct response.

Question 7(c)

(i) Many candidates completed this question successfully. Many demonstrated a lack of understanding of one (or both) of the two String methods.
(ii) Generally well done by those candidates who attempted this question.
(iii) Similar to part (i).

Question 8

(i)
```
Car car1 = new Car("Holden", "Commodore", 2007, "Red",  3000.00);
Car car2 = new Car("Ford", "Falcon", 2006, "Blue", 2500.00);
```

(ii)
```
car1.setDiscount(10);
car2.setDiscount(10);
```

(iii)
```
g.drawString(car1.getCarDetails( ), 20, 20);
g.drawString(car2.getCarDetails( ), 20, 40);
```

(iv)
```
car1.setColour("Black");
car1.setSalePrice(car1.getSalePrice() + 1000.00);
```

Examiner Comments

Question 8(a)

(i) Generally, well done by most candidates. Some did not construct statements with correct syntax and were awarded partial marks.
(ii) Those candidates who attempted this question generally provided correct responses for both.
(iii) Generally, well answered by the candidates who attempted this question with a wide variety of techniques used to display the required information.
(iv) Similar to part (ii).

Question 9

```
public class Property {

  String streetAddress;
  String suburb;
  int numberOfBedrooms;
  double rentalCost;
  boolean isRented;

  public Property(String streetAddress, String suburb,
                  int numberOfBedrooms, double rentalCost,
                  boolean isRented) {
    this.streetAddress = streetAddress;
    this.suburb = suburb;
    if ((rentalCost >= 0.0) && (rentalCost <= 1000.0)) {
      this.rentalCost = rentalCost;
    } else {
      System.out.println("Rent value out of range");
    }
    this.isRented = isRented;
    if ((numberOfBedrooms >= 1) && (numberOfBedrooms <= 5)) {
      this.numberOfBedrooms = numberOfBedrooms;
    } else {
      System.out.println("Room value out of range");
    }
  }

  public Property(String streetAddress, String suburb) {
    this.streetAddress = streetAddress;
    this.suburb = suburb;
  }

  public String getPropertyInfo() {
    String str = "Address = " + streetAddress + ", Suburb = " + suburb
               + ", Number of Bedrooms = " + numberOfBedrooms
               + ", Rental Cost = " + rentalCost
               + ", Available for rent = " + isRented;
    return str
  }

  public setRentalCost(double rentalCost) {
    if ((rentalCost < 0.0) || (rentalCost > 1000.0)) {
      System.out.println("Rent value out of range");
      return;
    }
    this.rentalCost = rentalCost;
  }

  public setIsRented(boolean isRented) {
    this.isRented = isRented;
  }
}
```

Examiner Comments

Question 9

Over 60% of candidates attempted this question; an increase of 10% over last year. Many were able to produce the code required to define the Class and its associated variables, an appropriate constructor method and the three required methods. Some students seemed unfamiliar with the term "overloaded" constructor. Only a small number of candidates included any form of validation to ensure that the numberOfBedrooms and the rentalCost were within the required range. Some candidates mistakenly defined methods to return the rentalCost and/or value of isRented rather than being able to set these values.

SECTION D - CRITERION 4

Question 10

(a) (i)

A	B	~ A ∧ B
F	F	
F	T	
T	F	
T	T	

~ A ∧ B
F
T
F
T

~ A ∧ B
F
T
F
F

~ A ∧ B
T
F
F
F

~ A ∧ B
F
F
F
T

(ii)

~A ∨ ~B	T	~A ∧ ~B	~A ∨ B

(iii)

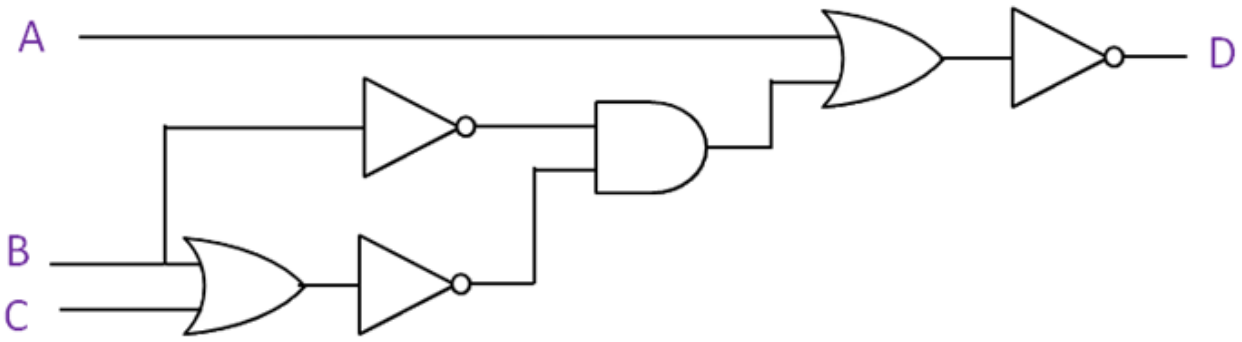
((~A ∧ B) ∨ (B ∨ C))	~ ((~A ∧ B) ∨ (B ∨ ~C))
~ ((~A ∧ B) ∧ (B ∨ ~C))	~ ((A ∧ ~B) ∨ (B ∨ ~C))

(b) (i)

A	B	C	~ A	~ B	~ A ∧ ~B	B ∧ C	F
0	0	0	1	1	1	0	1
0	0	1	1	1	1	0	1
0	1	0	1	0	0	0	0
0	1	1	1	0	0	1	1
1	0	0	0	1	0	0	0
1	0	1	0	1	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	1	1

(ii) F = (~A ∧ ~B) ∨ (~A ∧ B) ∨ (A ∧ ~B)
OR: F= ~(A ∧ B)
OR: F= ~A ∨ ~B
were all acceptable answers

(c) (i)



(ii) E ≡ ~C ∧ ~ (~A ∨ B)

(d) Multiplication is achieved by using repeated addition, which can be done with a loop that has a variable to count how many times the addition has occurred. ie: 7 + 7 + 7 + 7 + 7

Examiner Comments

Question 10
Generally well done across the board, with not many common errors made by the students.

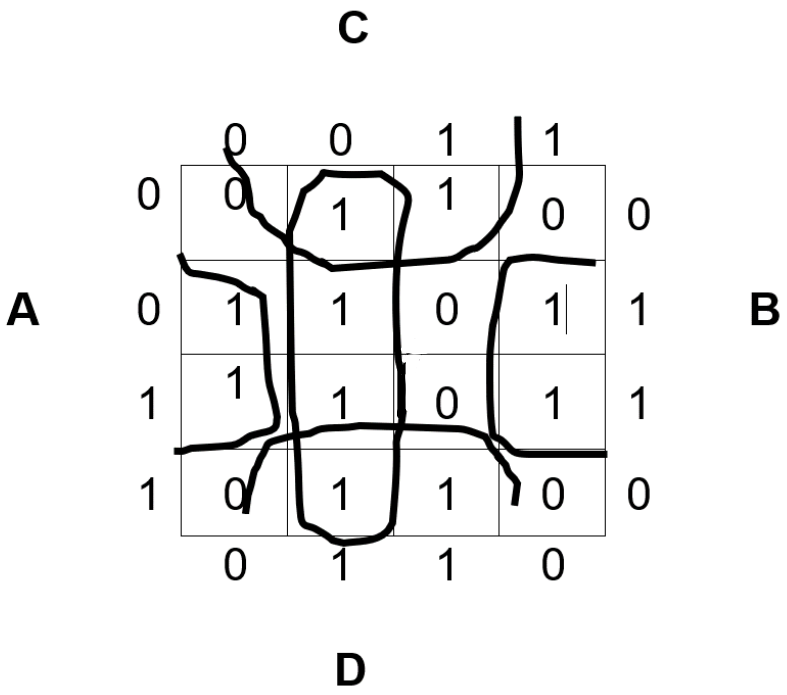
Question 10 (c) (i)
Frequently had students missing a NOT gate.

Question 10 (c) (ii)
Many students did not include the ~ (NOT) symbol outside the brackets of ~(~A ∨ B)

Question 10 (d)
The question stated "Explain" so a simple answer of "repeated addition – 7 + 7 + 7 + 7 + 7" only gained partial credit. To gain full credit, a mention of a loop or an extra variable to track how many times it has happened was required.

Question 11

(a) (i)



H ≡ B ∧ ~D ∨ ~B ∧ D ∨ ~C ∧ D

Or depending on groups
H ≡ B ∧ ~D ∨ ~B ∧ D ∨ B ∨ B ∧ ~C

(ii) ~ (~A ∧ ~ C) ∨ ~C

≡ ~ ~A ∨ ~~ C ∨ ~C	L 11 De Morgan's Law
≡ A ∨ C ∨ ~C	L 6 Law of negation
≡ A ∨ T	L 14 Law of excluded middle
≡ T	L 25 Law of Constants

(b) (i) The data moves from two of the registers into the ALU along a bus. The ALU performs the subtraction operation and the result is sent back to the destination register along a bus.

(ii) This architecture allowed for programs to be stored within the memory of the computer. This enabled computers to become general purpose devices which can easily switch from one program to another.

Examiner Comments

Question 11(a) (i)

This was generally well done.

Question 11(a) (ii)

Students frequently did one or two lines of simplification but did not go all the way to simplest form. Some students listed the laws they would use without showing any simplification process.

Question 11(b) (i)

Many students described data movement in the entire Fetch-Decode-Execute cycle here, rather than just the execute step. Since this all happens in the registers and the ALU, the memory is not used at all. Many students lost marks for mentioning that the data comes from memory or is sent back to memory after the operation.

Question 11(b) (ii)

This question was quite well answered most of the time. The biggest complaint was simply stating "The computer didn't have to be rewired" which is stated in the question, and the students were expected to expand on the benefit of that in a modern computing device.

Question 12

(a) (i)

14	2EAD	R[E] <- R[A] - R[D]	R[E] = i -3
15	CE1A	if (R[E] == 0) pc <- 1A	if (i – 3 = 0) goto 1A
16	DE1A	if (R[E] >= 0) pc <- 1A	if (i – 3 > 0) goto 1A
17	6BBC	R[B] <- R[B] >> R[C]	j = j / 2;
18	1AAC	R[A] <- R[A] + R[C]	i = i + 1
19	C014	if (R[0] == 0) pc <- 14	goto 14
1A	9A01	mem[01] <- R[A]	Store final value of i
1B	9B02	mem[02] <- R[B]	Store final value of j
1C	0000	HALT	

(ii)

Mem	Contents	i	j	R[A]	R[B]	R[C]	R[D]	R[E]
10	8A01	0	100	0				
11	8B02				100			
12	8C03					1		
13	8D04						3	
14	2EAD							-3
15	CE1A							
16	DE1A							
17	6BBC				50			
18	1AAC			1				
19	C014							
14	2EAD							-2
15	CE1A							
16	DE1A							
17	6BBC				25			
18	1AAC			2				
19	C014							
14	2EAD							-1
15	CE1A							
16	DE1A							
17	6BBC				12			
18	1AAC			3				
19	C014							
14	2EAD							0
15	CE1A							
1A	9A01	3						
1B	9B02		12					
1C	0000							

(b) Since loading data from memory is the slowest part of the Fetch-Decode-Execute cycle, having an instruction access the memory up to 4 times would obviously slow down each cycle dramatically. The 'Execute' phase of the cycle would require multiple memory accesses and become very slow.

A benefit might be that programs may become simpler and fewer lines required to do the same process, but each line would be much slower to execute and the overall program less efficient, due to the von Neumann bottleneck

Examiners Comments

Question 12 (a) (i)

It was good to see lots of students give this question a genuine attempt. It was quite difficult to have something that modelled the FOR loop exactly, and some shortcuts were taken by most students. Most got close to a solution, and even those that had slight errors were still given credit if the concept was sound. It was important for students to track the incrementing 'i' value (from 0 to 3) for full credit, although any who just had a loop that executed 3 times received partial credit.

Question 12 (a) (ii)

A few students didn't understand this trace table, either by writing lines of code in sequence (ignoring when jumps happen) or trying to show all changes to all registers in the first 4 lines of the table. Students were also penalised here for having "12.5" in the table as a result of doing 25 / 2. The correct answer is 12.

Question 12 (b)

There were few students who really understood how to answer this question. Many made vague reference to the Fetch-Decode-Execute cycle without stating anything clearly, and others who stated it would be more efficient due to the need for fewer ops. Some students made a vague reference about the von Neumann bottleneck, which was given partial credit.

SECTION E - CRITERION 5

Question 13

(a)

(i)	011_2	00111100_2	11000011_2	10001110_2
(ii)	3	11	21	25
(iii)	6	69	97	102

(b)

	1	0	1	1	1
+	1	0	1	0	1
1	0	1	1	0	0

(c) (i) 11110011_2

(ii) 1 x 0.5 + 0 x 0.25 + 1 x 0.125 = 0.625

(d) (i) Each additional bit doubles the range of possible values.
6 bits gives 64 values, or a range of 0 to 63 as an unsigned integer.

(ii) 0.101010_2 (truncated to 6 places) or 0.101011_2 (rounded to 6 places)

Examiner Comments

Question13(a)

These questions were generally well answered.

Question 13 (b)

For a straight-forward question, this proved harder than expected, with only about two thirds of candidates providing the correct answer. Students are advised to practice their binary addition in preparation for the exam.

Question 13 (c) (i)

This was relatively well answered. Common mistakes included starting with the wrong binary representation of 13, taking the 2's complement incorrectly, or not working with an 8-bit word length as required. A few students appeared to provide solutions in sign and magnitude notation.

Question 13 (c) (ii)

This was generally well answered with a wide variety of responses being considered acceptable.

Question 13 (d) (i)

This was relatively well answered. Students should be reminded, however, that questions like this require some, albeit brief, explanation. An answer of simply "63" (or "64") was not considered adequate.

Question 13 (d) (ii)

A common mistake was to write "101010" (as opposed to "0.101010") or to neglect the number of places (e.g. 0.1010101). Students should be encouraged to be explicit about how they are rounding off.

Question 14

(a) The two's complement representation stores every whole number between -32768 and 32767.
Eg: 32768, -32767, -32766, -2,-1, 0, 1, 2, 3, 32765, 32767.

A floating point representation would use the 16 bits to represent a sign bit, mantissa and an exponent and the result is that it does not distribute the numbers evenly across the number line, instead it is able to represent very small numbers using negative exponents and very large numbers positive exponents.

Eg Using a sign bit, 6-bit 2's complement exponent and 9-bit normalised mantissa

0 011111 111111111 = $0.111111111 \times 2^{+31}$ approximately 2×10^{9} (much larger)

0 111111 111111111 = $0.111111111 \times 2^{-32}$ approximately 2.3×10^{-10} (much smaller)

(b) (i) 4 bits to store the value and 2 bits to store the suit = 6 bits

(ii) 52 X 6 = 312 bits

(c)

0	1	1	1	1	(+15)
0	0	0	0	1	(+1)
1	0	0	0	0	(-16)

Adding two positive numbers and obtaining a negative result is an indication of an overflow error

Examiner Comments

Question 14 (a)

Most students attempted this question, but very few were able to provide a convincing explanation. Many students stated that a 16-bit floating-point representation consists of 1 sign bit, 5 bits for the exponent, and 10 bits for the mantissa, implying this is the only configuration and missing the point of the question.

Question 14 (b)

Most students attempted this question, with approximately half answering at least one part correctly, with many students getting confused about what was being counted. Some students unnecessarily included the values of each index in their calculations. If students came up with the wrong number of bits per card in (i), marks were still given in (ii) if it was clear they had multiplied by 52. As always, students should be reminded to include their calculations, not just a final result, to ensure every chance of obtaining partial marks if something has gone wrong.

Question 14 (c)

It was pleasing to see so many students attempt this question. For full marks students were expected to provide two integers that caused an overflow and perform the binary addition to demonstrate this. Many students were able to show that they understood the problem, but missed out on full marks by providing examples through the use of numbers that don't exist in a 5-bit two's complement representation (e.g. 31 + 1). A few students provided examples of additions that don't cause an overflow error (e.g 15 + -2), possibly confusing carry out with overflow.

Question 15

(a) (i) Assume that wind speed is measured to nearest km/h, and temperature to the nearest degree.
Wind speed 0 to 200 => 201 values => 8 bits
Wind direction 8 values => 3 bits
Temp range -20 to +50 => 71 values => 7 bits 18 bits per sample

OR

201 x 8 x 71 = 114,168 values => 17 bits (2^{17} = 131,072)

(ii) 24 x 60 x 18 bits = 25920 bits = 3240 bytes

(b) 0.1 is a recurring fraction in binary and hence cannot be stored precisely using a floating-point representation, which necessarily has a finite number of bits. Instead it is stored as an approximation of 0.1. In the code fragment, the final value of sum is reached by adding this approximation of 0.1 to itself 10 times, which is why the actual result is different from what we would expect to see.

Examiner Comments

Question 15 (a) (i)

Most students attempted this question, with approaches and answers varying widely. Students should be reminded to be very clear about the assumptions made. For example, only one student explicitly stated the assumption that wind speed would be recorded to the nearest km/h and temperature to the nearest degree. Although it typically didn't impact on their results, a number of students neglected zero (e.g. -20 to +50 is 71 values, not 70). Common mistakes that demonstrated a superficial understanding included adding the number of values required for each data type (e.g. 201+8+71 = 280 values => 9 bits) or multiplying the bits (e.g. 8 x 3 x 7 = 168 bits).

Question 15 (a) (ii)

Regardless of their success in the first part of this question, nearly half the candidates obtained full marks for this part by taking their value from (i) and scaling it to 24 hours and converting to bytes successfully.

Question 15 (b)

Relatively few students answered this successfully.