2021 ASSESSMENT REPORT
ITC315118 - COMPUTER SCIENCE

SECTION A

Question 1

a) 4 and/or 3

b) 22.9

c)
```
When "calculate" button is pressed
    if value in height TextField != 0
        Set weight to value in 'weight' TextField
        Set height to value in 'height' TextField
        bmi = (double)  weight / (height*height)
        display bmi
    else
       display "division by zero error enter a valid height"
```

d) Add the following lines after display bmi (above)
```
        if bmi < 18.5
                display "Underweight"
        else
                if bmi <= 24.9
                        display "Healthy"
                else
                        display "Overweight"
```

Examiner Comments

- Parts (a) and (b) were done very well by students.
- Part (c), many solutions successfully checked that height was greater than 0 but failed to use an else clause to prevent the calculation of the BMI when it was 0.
- Part (d) was generally well done but many students didn't use two 'else's' to correctly link all the statements together. It can be done with 3 if statements as well.
- The marker required students to write **BMI>=18.5 and BMI<=24.9**. Just using the notation **18.5-24.9** was not sufficient.

Question 2

a) Before Line 10 add

```
set hour to hour + 1
```

b) Modify the newly added line in part (a)

```
set hour to hours+1
If (hours==12)
        if pm=false
        set pm true
        else
         set pm false
if (hours>12)
        (hours =1)
```

Examiner Comments

Question 2 was generally well done with some confusion caused by not reading the question carefully or not understanding converting 24-hour time to 12-hour time. Toggling pm could more easily be done using (*pm = not pm*).

The most common misunderstanding was toggling pm when (hours>12) instead of (hours =12) but students were not penalised for not knowing the subtilties of the 12-hour clock. There was some confusion about the use of pm as a Boolean.

Question 3

```
Initially
roll = 0
last_roll = -1
second_last_roll = -2
score = 0
count = 0

When a number is entered into "roll" TextField
Set roll to value in "roll" TextField
if (roll > 6) or (roll < 1)
        roll = 0
display "Roll is " roll
```

```
When the "Calculate" button is pressed
If(score=90) and (roll=1) score = 91
if (second_last_roll = last_roll)
        if (score + roll * 2) <=91
                score = score + roll * 2
 else
        if (score + roll <= 91)
                score = score + roll
 display "Score is " score
 second_last_roll = last_roll
 last_roll = roll
 count = count + 1
 if (score==91)
        display "Number of rolls to reach 91 was " count

When "reset" button is pressed
 set roll = 0
 set last_roll = -1
 set second_last_roll = -2
 set score = 0
 set count = 0
```

Examiner Comments

This was a challenging algorithm, and it was pleasing to see that most students who attempted this question had some success but there were very few perfect solutions. The main problems were:

1. Not stopping the score passing 91 if the last toss took you past 91. The intention was to reject this toss and continue until a suitable final toss was achieved. This was not made clear in the question although the sample game did do this at roll 17
2. Ignoring the instruction that if the score gets to 90 a throw of 1 will end the game regardless of the two previous throws being the same. This happens in the sample game at roll 18.
3. Many students wanted to double the second toss when "two in a row occurred" rather than the toss that followed "two in a row". As this was a common error, students were not penalised to any significant extent. The method used above requires the storing and updating of the two previous tosses. It can be done in other ways.

SECTION B

Question 4

a) i. 8.0

ii. 0

iii. 9.0

b) i. Final value of **d** is 4. Integer division discards the remainder.

ii. Final value of **e** is 'Y'. temp is used to hold value while the contents of e and f are swapped.

iii. Final value of **g** = 6

g	i
3	2
	3
6	4
	5
	6

iv. Final value of **x** = 0

x
21
7
2
0

c)
```
if (input % 2 == 0)
{
    output = input * 2;
} else {
    output = input * 5;
}
```

Examiner Comments

a) These questions were generally very well answered.

b) i. This question was generally well answered.

b) ii. This question was reasonably well answered. Common errors included converting to int and being confused about the order of assignments when apparently tracing through the code in reverse. Some students obtained partial marks by providing an otherwise correct explanation despite making a mistake.

b) iii. A few students went through the loop one time too many, but otherwise this was generally well answered.

b) iv. Generally well answered, with all students obtaining at least partial marks. Common errors included neglecting integer division and stopping before **x** reached 0.

c) Most students knew what was required, but many made basic syntax errors, including '=' instead of '==' for the comparison, omitting brackets around the condition, and omitting ';'.

Question 5

a) Final value of q = 13. As there are no break statements, the switch will continue to evaluate all remaining cases after its first match, case 3.

b) boolean isRightAngle = part5b(3.0, 4.0, 5.0);

c) i. Final value of **m** is 16

		j		
		0	1	2
i	0	1	2	3
	1	5	6	7
	2	9	10	11
	3	13	14	15

ii. This would result in an ArrayIndexOutOfBounds exception error

Examiner Comments

a) The majority of students obtained partial marks by correctly following case 3 but neglecting the absence of the break.

b) Most students attempted this question, but only about a third obtained full marks. Students were expected to write code using correct syntax demonstrating how the method would be called and how to do something with the returned value.

c) i. Relatively few students obtained full marks for this question. A common mistake was to neglect the increment at start of each row. Similarly, the final increment after the grid was filled, was often overlooked.

ii. This was reasonably well answered by the students who attempted it. Even if the name of the error wasn't provided, partial or full marks were given based on the explanation of the problem.

Question 6

a) **inStr** = "FROG"

batch after **getBatch()**

0	1	2	3
5	17	14	6

code after **encode()**

0	1	2	3
155	111	108	76

since
code[0] = 3×5 + 10×14 = 155
code[1] = 3×17 + 10×6 = 111
code[2] = 2×5 + 7×14 = 108
code[3] = 2×17 + 7×6 = 76

code after **adjustCode()**

0	1	2	3
20	3	0	22

since
code[0] = 155 % 27 = 20
code[1] = 111 % 27 = 3
code[2] = 108 % 27 = 0
code[3] = 76 % 27 = 22

outStr = "UDRAW"

b) The purpose of variable **p** is to keep track of the current position of the input string to ensure that it is processed in groups of 4.

c) Variable **p** is a global variable. Changes made to it within **getBatch()** need to be visible outside the method for the program to work correctly. If it were passed as a parameter and not returned, then changes made within the method would not be visible outside the method.

By contrast, variable **input** is only used within **getBatch**() and is not modified. The **batch** array is modified, but because it is a reference type, the changes made within the method will be visible outside the method.

Examiner Comments

a) Few students attempted this question. However, of those that did, many were able to obtain full or close to full marks.

b) These parts were generally well answered by those who attempted them.

c) These parts were generally well answered by those who attempted them.

SECTION C

Question 7

a) i. Answer 2: Bicycle b = new Bicycle(27.5, 3);

ii. Answer 2: 1

iii. Answer 2: double

iv. Answer 3: b1.changeGear(1);

b) s = 60*3.14*29.0/60*5 = 455.3 (if it is a double)

s = 455 (if it is an int)

c)
```
public void setNumberOfGears(int numberOfGears) {
        this.numberOfGears = numberOfGears;
    }
```

d) After the line "else if (direction == 1) {"

Insert the code: if (this.year < this.numberOfGears)

this.gear++;

Examiner Comments

This question was very well answered by most students. There was a departure from previous years in that students were asked to write code and there was less focus on working with inbuilt Java classes. This seemed to be welcomed by students.

a) Most students answered all 4 of these multiple-choice questions correctly which was pleasing to see. Some students selected more than one answer because they thought the question suggested it.

i. Almost all students recognised that there needed to be a double followed by an integer in the parameters.

ii. A number of students circled 2 (counting void as a parameter).

iii. Most students correctly circled 'double', though some circled 'int'

iv. Most students chose the correct answer.

b) Some students pointed out that because the getSpeed method returned a double and s was defined as an integer this code would not compile. Because of this error both the answers 455 and 455.3 were accepted. Because the calculation was not simple, marks were deducted for not showing working out.

c) Many students correctly answered this question. Some students did not obtain full marks because of a missing semi-colon.

d) This question was very well done. Most students coped well with the fact that the code did not have line numbers by giving instructions for where their code should be added or rewriting the whole method.

Question 8

a) i. Leg leg1 = new Leg('A', 3.2, 33.0);
Leg leg2 = new Leg('B', 5.3, 71.0);

ii. leg1.setDistance(4.1);

iii. double speed2 = leg2.getDistance() / leg2.getTime();

iv. double bearingChange = leg1.bearingChange(leg2);
(because of the way the legs are specified in the question)

OR,

double bearingChange = leg2.bearingChange(leg1);

b) It is the class constructor and these do not need to specify a return type as they never have one.

Examiner Comments

This section was more typical than question 7. Overall, it was well answered by students. This question was similar to a previous year's exam and many students obtained full marks for the entire question. Only a couple of students noticed that the variable for timeTaken was referred to elsewhere in the question as time so the code would not compile.

a) i. The legs needed to be labelled as 'A' and 'B' in the parameters for full marks.

ii. Very well done.

iii. A number of students wrote their own calculateSpeed method for this question. This would have been correct if they then had provided code to calculate the speed of leg2. Almost all students were aware that Speed = Distance / Time.

iv. Many students accessed the getDistance method twice to calculate the bearing change, rather than using the bearingChange method. Whether this was because they did not read the code carefully enough or because they realised the bearingChange method contained an error it is hard to ascertain. Students were not penalised for this as they were still using a method from the code.

b) Almost all students provided a successful answer here. Only one student pointed out that because the code contained the word "Public" it would not compile.

Question 9

```
public class Scoreboard {
    public int minutes, seconds, scoreA, scoreB, teamFoulsA, teamFoulsB, period;
    public Scoreboard() {
        minutes = 12;
        seconds = 0;
        scoreA = 0;
        scoreB = 0;
        teamFoulsA = 0;
        teamFoulsB = 0;
        period = 1;
    }
    public increaseScore(char team, int pointsIncrease) {
        if (team=='A') {
            if (pointsIncrease >= 1 && pointsIncrease <= 3)
                scoreA += pointsIncrease;
            if (scoreA>99)
                scoreA = 99;
        }
        if (team=='B') {
            if (pointsIncrease >= 1 && pointsIncrease <= 3)
                scoreB += pointsIncrease;
            if (scoreB>99)
                scoreB= 99;


        }
    public decreaseScore(char team, int pointsDecrease) {
        if (team=='A') {
            if (pointsDecrease >= 1 && pointsDecrease <= 3)
                scoreA -= pointsDecrease;
            if (scoreA<0)
                scoreA = 0;
        }
        if (team=='B') {
```

```
                    if (pointsDecrease >= 1 && pointsDecrease <= 3)
                        scoreB -= pointsDecrease;
                    if (scoreB<0)
                        scoreB= 0;


            }
        public increaseFouls(char team) {
            if (team=='A')
                teamFoulsA++;
            if (team=='B')
                teamFoulsB++;
        public decreaseFouls(char team) {
            if (team=='A')
                teamFoulsA--;
            if (team=='B')
                teamFoulsB--;
        }
        public incrementPeriod() {
            if (period<4)
                period++;
            minutes = 12;
            seconds = 0;
            teamFoulsA = 0;
            teamFoulsB = 0;
        }
}
```

Examiner Comments

Many students did not attempt this question which was disappointing as the question was more accessible than previous years. Those that attempted the section did quite well. Some students wrote their answer more in the style of Criterion 1 with Initially/When – these students received part marks if their logic was correct. Other students avoided repetition with some of the methods by describing code that would be duplicated and what would be changed – this approach was given full marks.

In marking this section, it was assumed that:

- scoreA and scoreB would be reset to 0 if they went below 0
- scoreA and scoreB would be reset to 99 if they went above 99
- the increaseScore and decreaseScore methods needed to check that a value of 1, 2 or 3 was passed to them
- teamFoulsA and teamFoulsB did not need to be checked to see if their values were below 0 or above 9.
- that the incrementPeriod method needed to ensure the period was not increasing above 4.

SECTION D

Question 10

a) i. $\sim A \wedge \sim (B \vee A) \wedge C$

ii.

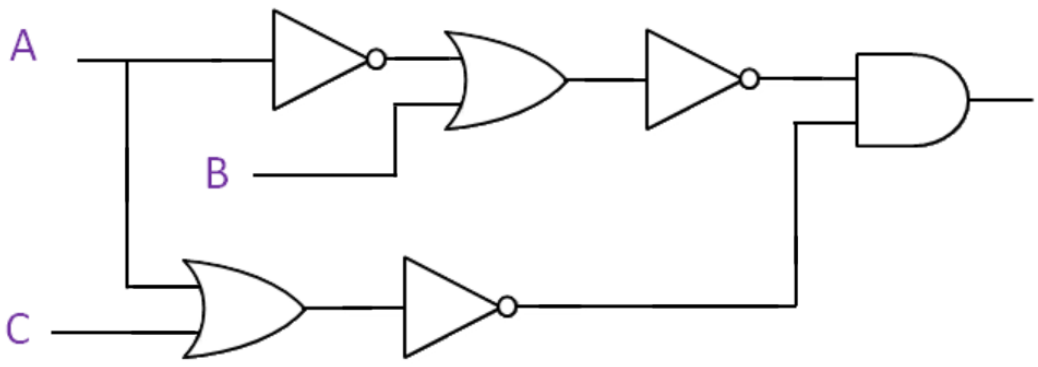
~ A V B
T
T
F
T

iii. F

b) i.

A	B	C	(B∧C)	~(B∧C)	~B V ~(B∧C)	A ∧ (~B V ~(B∧C))	~(A ∧ (~B V ~(B∧C)))
0	0	0	0	1	1	0	1
0	0	1	0	1	1	0	1
0	1	0	0	1	1	0	1
0	1	1	1	0	0	0	1
1	0	0	0	1	1	1	0
1	0	1	0	1	1	1	0
1	1	0	0	1	1	1	0
1	1	1	1	0	0	0	1

ii.



iii. $H \equiv A \wedge B \wedge \sim(B \vee C) \vee \sim (B \vee C) \equiv \sim(B \vee C)$

c)

Memory Address	Contents	Pseudocode	Explanation
1	0009	data	mem[01] contains value 9
2	0002	data	mem[02] contains value 2
10	8A01	R[A] ← mem[01]	Load 9 into register A
11	8B02	R[B] ← mem[02]	Load 2 into register B
12	6CAB	R[C] ← R[A] >> R[B]	R[C] = R[C] / 4
13	0000	Halt	

On program termination R[C] = 2

Examiner Comments

a) These questions were generally well answered, although some students assumed that iii couldn't be simplified.

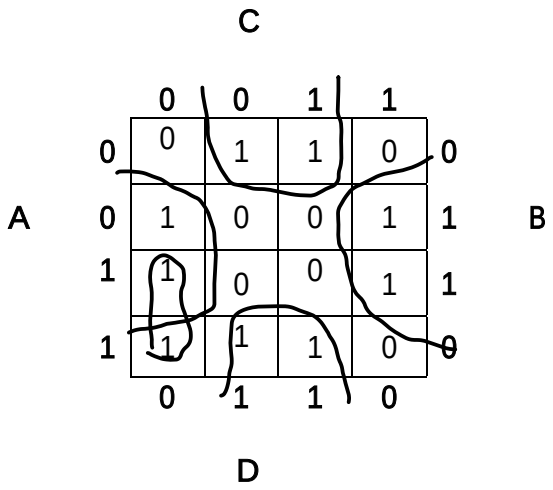
b) i. A number of students did not realise that they could build each column from the value in a preceding one. Many students mistakenly assumed that ~(B∧C) was the same as ~B∧~C.

b) Parts i. and ii. were generally well answered.

c) Many students were unfamiliar with how the shift operator works.

Question 11

a) i.



B ∧ ~D V ~ B ∧ D V A ∧ ~C ∧~D

b) i. program counter

ii. The pc is an extra register that keeps track of the next instruction to be executed. It can be modified with branch and jump instructions to create ifs and loops.

Examiner Comments

a) i. Most students got most of the expression correct, but either forgot or were unable to provide an expression for the 1 in the lower left corner.

a) ii. Many students knew the answer, but only about half were able to provide the specific logic laws used to get there. Some students changed the parentheses to allow them to use laws which didn't apply.

b) Most students did not know the specific term "Program Counter", but a reasonable number were able to give a partial answer related to branching or loops.

Question 12

a) i.

Memory Address	Contents	Pseudocode	Explanation
01	0004	data	Used for variable x
02	0000	data	Used for variable y
03	0003	data	Constant decimal 3
10	8A01	R[A] <- mem[01]	R[A] = 4
11	8B02	R[B] <- mem[02]	R[B] = 3
12	DA16	if (R[A] > 0) pc ← 16	If (x>0) branch to line 16
13	1ABB	R[A] ← R[B] + R[B]	R[A] = 3 + 3 = 6
14	9A02	mem[02] ← R[A]	y = 6
15	0000	halt	
16	9A02	mem[02] ← R[A]	y = 3
17	0000	halt	

a) ii.

Mem Address	Contents	R[A]	R[B]	R[C]	mem[01]
01	0000				
10	8B01		0		
11	8AFF	5			
12	CA17				
13	DA17				
17	2CAB			5	
18	DC20				
20	9A01				5
21	C010				
10	8B01		5		
11	8AFF	3			
12	CA17				
13	DA17				
17	2CAB			-2	
18	DC20				
19	C010				
10	8B01		5		
11	8AFF	-1			
12	CA17				
13	DA17				
14	8C01			5	
15	9CFF				
16	0000				

The purpose of this program is to read in consecutive integer values from the keyboard and to store the largest valued encountered and then display this value on termination. Program is terminated when a negative value is inputted.

As TOY uses 16 bits 2's complement -1 would be entered as FFFF

b) i. In one iteration of the loop the code will need to:

- access the memory location of i three times - to test for the end condition and to increment and store the value of i.
- access the memory location of b once – to test for the end condition
- access the memory location of c twice – to add on the value of d and store the new value
- access the memory location of d once – to add it to c.

This gives a total number of data access during an iteration of the loop of:
3 + 1 + 2 + 1 = 7
Other solutions with justifications were acceptable.

b) ii. If code is rewritten to maximise the use of internal registers then the values of i, b, c, d could be stored in registers. This would mean that no memory accesses are necessary during the iterations of the loop it would simply require four memory accesses at the beginning of the loop to retrieve the values of i, b, c, d and then one at the end to store the value of c.

Examiner Comments

a) i. About half of the students attempted this question. Those that did generally had a reasonably good solution, some were even quite innovative. The most common error that cost marks was failing to write the final result back into the memory location used for variable y, and also allowing one branch of the if/else to fall into the other, overwriting the calculated value.

a) ii. About one third of students attempted this question. The explanations provided in the question for lines 10 and 14 were incorrect (although the specific instruction contents were correct) and many students were misled by this. Students who answered assuming the explanations were correct were still awarded most of the marks for the question. Many students were not familiar with how to enter values into the TOY machine, and for the question regarding -1 discussed how this value might be calculated instead. Many students were not aware that the "Mem Address" column in the table was supposed to track which line of code was being executed.

b) iii. Almost half of the students attempted this question. There were a lot of solutions that could be correct depending on the assumptions made, and so most answers with reasonable justifications were awarded points. A reasonable number of students seemed not to realise that the "one iteration of the loop" was not supposed to include the initialisation of the variables.

b) i. About one quarter of students attempted this question. The reference to the von Neumann bottleneck was misleading in this question and so credible answers that referenced this issue were also accepted as correct.

SECTION E

Question 13

a) i. 01011010_2

ii. 53

iii. 65

b)

	1	1	0	1	1	
+	1^1	1^1	1^1	1	0	
1	1	1	0	0	1	

c) i. 112 - 01110000_2

ii. 97 is 01100001_2
-97 reverse all bits and add one which yields 10011111
Now perform 112 + (-97)

$$\begin{array}{rr} & 01110000_2 \\ + & 1\ 10011111_2 \\ \hline & 00001111_2 \end{array}$$

answer is 15

d) 8 bit means $2^8 = 256$ colours

Examiner Comment

a) This question was completed correctly by nearly every candidate and very few omitted an answer for (i).

b) This question was completed correctly by nearly every candidate.

c) (i) and (ii) were completed correctly by nearly every candidate but it was very common to omit subscript to indicate base 2 in part (i)

c) (iii) This was completed correctly by most students, some variations included 255 (forgot to include 0) 128, 8^2

Question 14

a) The largest positive number that can be represented in a four-bit 2s complement system is 0111 which is 7. If you add 4 + 5 the result 9 will force a 1 into the most significant bit, this is known as an overflow error and the resulting number will be a negative.

b) 0.10112 = ½ + 1/8 + 1/16 = 11/16 = 0.6875

c) 1024 would be $0.1 \times 2^{+11}$ 0 01011 1000000000
0.25 would be 0.1×2^{-1} 0 10000 1000000000
When following the algorithm for addition:
Step 1 **Adjust** the numbers so that **exponents** are the **same**.
.25 becomes 0.**0000000000**001 $\times 2^{+11}$ the problem is that there are only 10 bits in the mantissa so the 1 is lost in this process leaving 0.25 to be represented as 0.

Examiner Comment

a) This question was generally completed correctly by most students, however a small number failed to illustrate the answer (as asked by the question).

b) This was generally completed correctly by most students.

c) Most students had difficulty in explaining the result. Only a few students provided a detailed description of why this occurred.

Question 15

a) i. 4 bit (in two's complement)

ii. 3.5 x 60 x 22000 x 4 = 18480000 bits

b) i. 3.5 x 60 x 44100 x 16 = 148176000 bits = 1852200 bytes = 17.66MB

ii. 16 bit gave range of +32767 to -32768
using 10 bit floating point with one sign bit, 5 bit exponent (2's complement) and 4 bit mantissa largest number would be 0 01111 1111

0.1111 x 2 +15 which is 30720

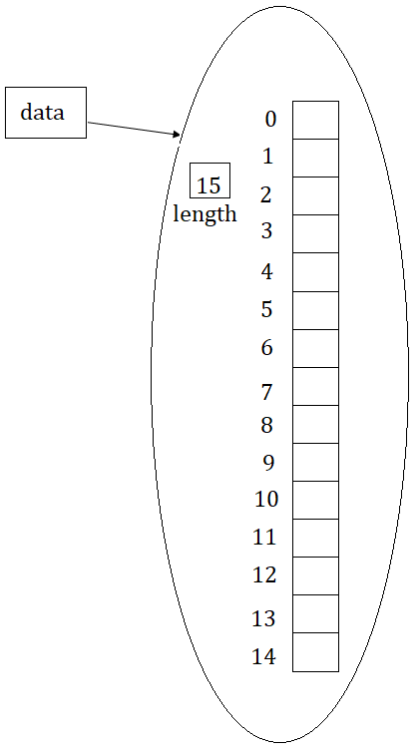
Smallest number would be -30720

iii. Floating point numbers are not distributed evenly across the number line meaning that values close to max or min amplitude would be rounded more than those closer to 0. The second largest representable values would be:

0.1110 x 2 +15 which is 28672 this represents a gap of 2048

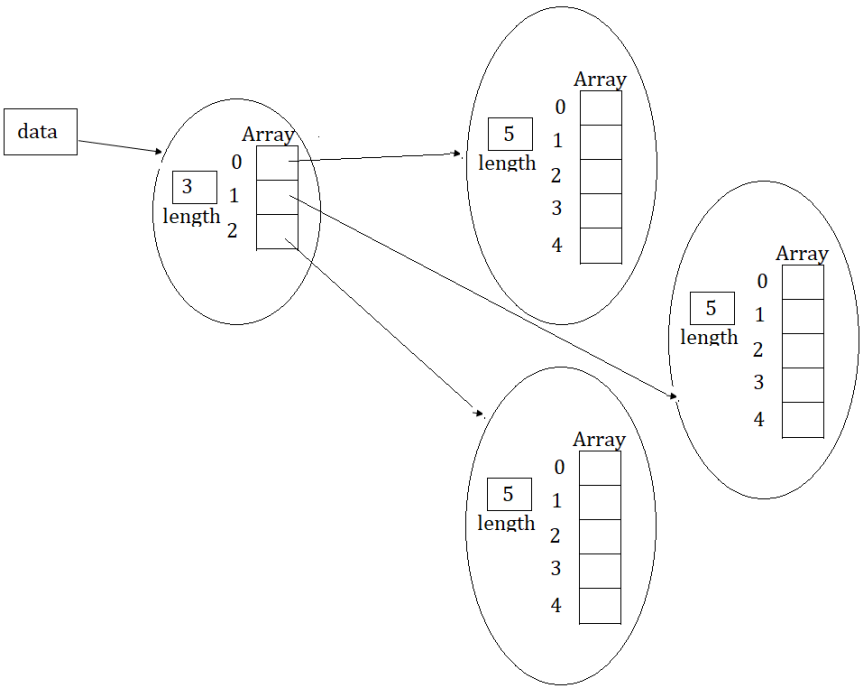
Any samples falling between these two values could be rounded significantly and this may lead to a loss of sound quality.

c) Each box represents a storage location (word). Using a one dimensional array:



The number of words required would be 17 (15 for each of the elements, one for the array variable and one for the array length)

Using a two-dimensional array (3 x 5):



This would require 23 words in total, 6 more than for a single dimension array. Hence their friend was correct.

Examiner Comment

a) i. A little over half the students who attempted this question, identified the required number of bits to represent the range of -8 to +7 was 4 bits

ii. Most students who attempted the question used the correct formula and no marks were deducted if their answer to (i) was incorrect as long as they showed their working out.

b) iii. This was correctly answered by the majority of those who attempted this question.

iv. Most students struggled to answer this question completely. A small number of students identified a representation but did not justify their choice by illustrating the range of numbers available and how that compared with the original 16 bit integer representation.

v. Most students who answered this question indicated that there would be a loss of sound quality but very few attempted to explain why.

Question 16

Most students who attempted this question asserted that their friend was correct, and that less memory is required for the one-dimensional array. The methods used to illustrate and support this conclusion varied widely. The better answers illustrated the two variations with diagrams and then added up the required memory locations of each to state their conclusion.