Computer Science

Course Code: ITC315113

Generally speaking, the paper was reasonably well answered. I would advise both students and teachers to spend more time analysing and tracing code prior to the exam. Some poor results in both section B and C resulted from an inability to read and understand code. A basic lack of understanding of "objects" was also of concern.

Section A

Question 1

(a)

Ingredients for 5 pancakes: 170 *g of flour and* 1 *egg and* 60 *ml of milk*

(b) & (c)

Initially

```
Set pancakes = 5
Set flour = 170
Set eggs = 1
Set milk = 60
```

When a number is entered into the "Number of pancakes?" Textfield

set pancakes to value in ***"Number of pancakes?"*** TextField

```
if pancakes less than 5
        Set pancakes = 5
if pancakes greater than 17
        Set pancakes = 17
Display pancakes
```

When the "Calculate" button is pressed

```
Set flour = pancakes * 34
Set milk = pancakes * 12
if pancakes less than 8
        Set eggs = 1
else
        if pancakes less than 13
                Set eggs = 2
        else
                Set eggs = 3
Display "Ingredients:" flour "g of flour and " eggs "eggs and " milk "ml of milk"
```

Question 1 Comments

Part a was answered very well by almost all students.

Part b was also answered well by the majority of students.

Part c - There was some difficulty here with the way students added the extra part of the "if statement" (they really didn't understand a nested if). If they did the easy option, that is just added another if statement for >12, then it would have been fine. Those that answered this section well generally did well in Question 2.

Question 2

(a) The text value would be 0.

Initially

```
Set shake1 = 0
Set shake2 = 0
Set score = 0
Set cleared = false
Set shake1_done = false
```

When a number is entered into the "Number to drop on first shake?" Textfield

```
if  shake1_done equals false
Set shake1 to value in "Number to drop on first shake?" TextField
        Set shake1_done = true
        if shake1 greater than 7
                Set shake1 = 7
        Display shake1
```

When a number is entered into the "Number to drop on second shake?" Textfield

```
if shake1_done equals true
Set shake2 to value in "Number to drop on second shake?" TextField
        if (shake1 + shake2) greater than 7
                Set shake2 = 7 – shake1
        Display shake2
```

When the "Score" button is pressed

```
if cleared equals true
        Set score = score + (2 x shake1) + shake2
        Set cleared = false
else
        Set score = score + (shake1 + shake2)

if (shake1 + shake2) equals 7
                Set cleared = true

Set shake1_done = false
Set shake1 = 0
Set shake2 = 0
Display score
```

Part a, had an interpretation issue with many students considering the value of score and not the value in the textfields. Most students just gave an answer of 0(zero) and this was accepted.

Part b, the first part was answered reasonably well although the test used by many was "if the second shake >7", which it will never be, rather than "if shake1 + shake2 >=7". They generally got the next line after the "if" correct for the assignment of "shake2 value".

The second part of this generally determined if the student had the ability to get a B and many didn't get the fact that the bonus was added on the round after "shake1 + shake2 =7". Many also interpreted incorrectly the bonus only occurred "if shake1 = 7". They also had trouble sequencing the bonus test in this.

Question 3

Initially

```
Set distance = 0
Set raining = "N"
Set windy = "N"
Set freezing = "N"
Set time_bike = 0.0
Set time_car = 0.0
Set time_walk = 0.0
Set travel = ""
```

When a number is entered into the "Enter distance to travel?" Textfield

Set distance to value in ***"Enter distance to travel?"*** TextField

```
    if distance > 20
Set distance = 20
```

When text is entered into "Is it raining?" TextField

Set raining to value in ***"Is it raining?"*** TextField

When text is entered into "Is it windy?" TextField

Set windy to value in ***"Is it windy?"*** TextField

When text is entered into "Is it freezing?" TextField

Set freezing to value in ***"Is it freezing?"*** TextField

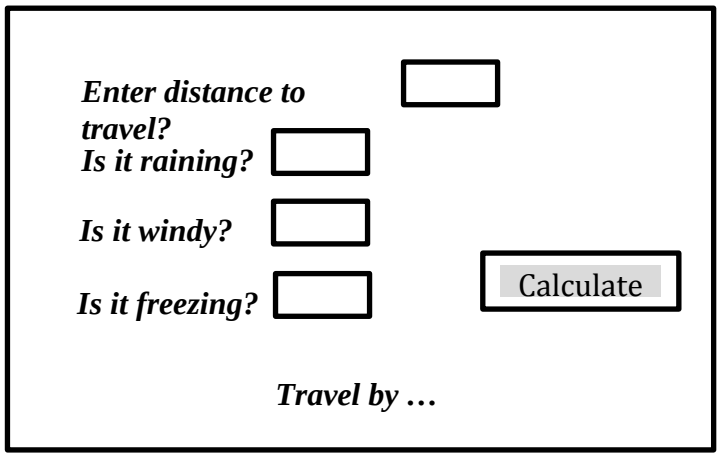
When the "Calculate" button is pressed

```
Set time_walk = 60 x distance / 5
Set time_bike = 60 x distance / 15
Set time_car = 60 x distance / 50
    if windy equals "Y"
        Set time_walk = 60 x distance / 4
        Set time_bike = 100

    if raining equals "Y"
        Set time_walk = 100
        Set time_bike = 100
        Set time_car = 60 x distance / 40

    if (freezing equals "Y" and  raining equals "Y"
        Set time_walk = 60 x distance / 3
        Set time_bike = 100
        Set time_car = 100

    if time_bike <= 60
        Set travel = "Bike"
    else
        if time_walk <= 60
            Set travel = "Walk"
        else
            if time_car <= 60
                Set travel = "Car"
            else
                Set travel = "No option available"
    Display "Travel by" travel
```



The answer above is just one of the possible solutions and the markers looked carefully at any solution provide to see if it met the requirements.

Students who had a go generally did enough to support their performance in the other 2 section. Very few actually took into account the travel time and many lost track of what they were doing.

About 5-8% of the students gave a complete solution that was appropriate and met the brief.

Section B

Question 4

Many students struggled with operator precedence. Many students missed out on a C rating because they had not satisfactorily answered question 4 and did not attempt question 5 or 6.

(a) (i) a = 18 *Explain:* 20 – 12 / 6 = 20 – 2 = 18
(ii) b = 32.0 *Explain:* 8 * 4 = 32
(iii) c = false Explain: Boolean values can contain only true or false ∴ !true = false

(b) (i) d = 10 *Explain:* d = 5, (d>0) true, d = 10, (d>10) false, d=10
Some students stated that 10 was greater than 10 so they either entered the nested or looped until d =20
(ii) e = 20

i	e
	0
1	2
2	6
3	12
4	20

Adding 2 to e before multiplying was a common error.

(c) f = 2

t	*u*	*f*
5	5	0
4	4	0
3	6	1
2	6	2
1		

Many students ignored the integer division (6 / t) and many ignored the while (t > 1)

Question 5

(a) h = 4
Some students ignored while (x[h] != '~') and found h = 5

x

0	1	2	3	4	5
'1'	'C'	'2'	'B'	'~'	'A'

h
0
1
2
3
4

(b) Answered well by many of the students who attempted this question but some students had difficulty with k[i] = k[i-1] + 3.

i	*k*				
	0	1	2	3	4
0	2				
1		5			
2			8		
3				11	
4					14

Final k

0	1	2	3	4
2	5	8	11	14

(c) Many students did not attempt this question and partial attempts were common. Common errors were to ignore integer arithmetic and not understanding % (modulus). Some successful students showed their workings.

m

0	1	2	3	4
19	37	44	54	77

Final n

0	1	2	3	4
44	54	77	19	37

i	*y*	*z*	*n*				
			0	1	2	3	4
			0	0	0	0	0
0	8	3				19	
1	4	4					37
2	0	0	44				
3	1	1		54			
4	0	0					
4		1					
4		2			77		

Question 6

Many students did not attempt question 6.

Well answered by many of the students who attempted it. Some students wrote 10 in the index column, rather than the place column, and some were not sure of where to position '^'

(a)
board[ ]

0	1	2	3	4	5	6	7	8	9
~	1	~	2	~	3	3	A	A	~

place	*move*	*index*
1	3	
		0
		3
		5
		8
		7
		6
		9
10		

Board: ~ 1 ~ 2 ~ 3 3 A A ~
Place: ^
Board: ~ 1 ~ 2 ~ 3 3 A A ~
Place: ^
Win!

Students who had some success in part (a) tended to have some success in part (b).

(b) (i) This creates an infinite loop between index 3 '*B*' and index 1 '*2*'. The board must be set up so that two locations do not send the piece backwards and forwards between them.

(ii) This creates an *Array Index Out Of Bounds Exception* because ***index*** goes beyond the end of the array. Either the board must be set up so that this will not happen or the program must identify this condition before the error occurs and change the value of ***index***.

Very few students understood the meaning of "the scope of a variable". Many students tried to give a value or a range of values. Some of these were able to answer (ii) quite well.

(c) (i) The variable ***index*** is declared as a parameter to ***nextPlace*** and has a scope of the method ***nextPlace***. The scope is local to nextPlace.

(ii) The code will give a variable undefined error *[index cannot be resolved to a variable]* as the scope of ***index*** is within the call of the method ***nextPlace***.

Section C

Section C was particularly poorly answered. Many candidates had no idea how to declare and instantiate an object and knowledge of the string functions (and the ability to interpret them from the information booklet) was very poor.

Q7 (a) generally, well done but many candidates failed to fill in the contents of the text fields and decided "O" was the winner without analysing the code!

Question 7

(a) (i)

aField: X_____
bField: X _____
cField: - _____
dField: -_____
playField: X win! _____

(ii)

aField: X_____
bField: X _____
cField: O _____
dField: O _____
playField: X win!_____

(b) Given that these string questions have been relatively common on past papers, this was poorly answered. I would suggest that students (teachers also need to address the problem) familiarise themselves with these methods and the information booklet.

String2: wasted two worms
String2 wasted two terms

Question 8

Many students had no idea how to declare and instantiate an object which is disturbing. A common error was to not enclose the ID (a String) in quotes when instantiating. Some students wanted to use 1001 as the Object name and this is invalid (but didn't incur much penalty when marking). Students (and teachers) should concentrate on analysing class definitions and how to use them.

(a)

```
Item  item1 = new Item("1001","Meat Pie", 2.30, 10);
Item  item2 = new Item("2012","Can of Lemonade", 0.70, 23);
```

(b) (i)

```
item1.setPrice(2.40);
```

(ii)

```
g.drawString("$"+item1.totalSaleValue(), 100, 200);
```

(c)

```
item2.setNumber(item2.getNumber() - 1);
```

Question 9

Generally the students who could answer Q8 were successful in this question. A low percentage of students made any credible attempt at this question. Most used the examiners prompting and used an array to store the four opponent pieces but it was allowable (but inefficient) to use 4 strings. There was a lot of "poor programming" technique in this section but that did not incur penalty. It was good to see some students using a "ListArray" object correctly, when this was not necessary, but quite neat.

```
public class Piece
{
      public     String     name, position;
      public     int        number;
      public     String[]   take = new String[5];

      public Piece(String newName, String newPosition)
      {
            name = newName;
            position = newPosition;
            number = 0;
      }

      public void resetTake()
      {
            number = 0;
      }

      public void addTake(String takePiece)
      {
            number = number + 1;
            take[number] = takePiece;
      }

      public boolean findK()
      {
            boolean found = false;
            for (int i=1; i<=number; i++)
            {
                  if (take[i].equals("K"))
                        found = true;
            }
            return found;
      }
```

Section D comments

Question 10

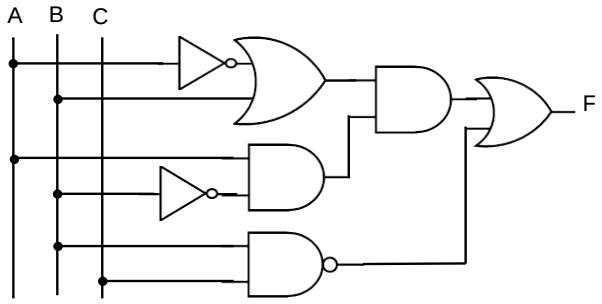
(a) (i)

A	***B***	***~B***	***A ∧ ~B***	***A ∧ B***	***E***
0	0	1	0	0	0
0	1	0	0	0	0
1	0	1	1	0	1
1	1	0	0	1	1

(ii) $E \equiv A$

Generally answered well, although a few students simply repeated the original expression as response to part ii.

(b) $F \equiv ((\sim A \vee B) \wedge (A \wedge \sim B)) \vee \sim(B \wedge C)$



Generally answered well. The most comment mistake was made with the last part of the expression, with many students drawing ~B ^ C rather than ~(B ^ C).

Students were expected to draw a logic circuit that represents the given expression, not a simplified version of the expression.

(c)
```
If (b==0)
c = 1;
else
c = a / b;
```

Answered reasonably well, although some students either had c = a / b before the if statement (or omitted c = a / b entirely). A few were over cautious e.g. e.g. only allowing c = a / b for b > 0.

Given the focus of this section of the exam, students were not required to write perfectly formed Java to obtain full marks.

Question 11

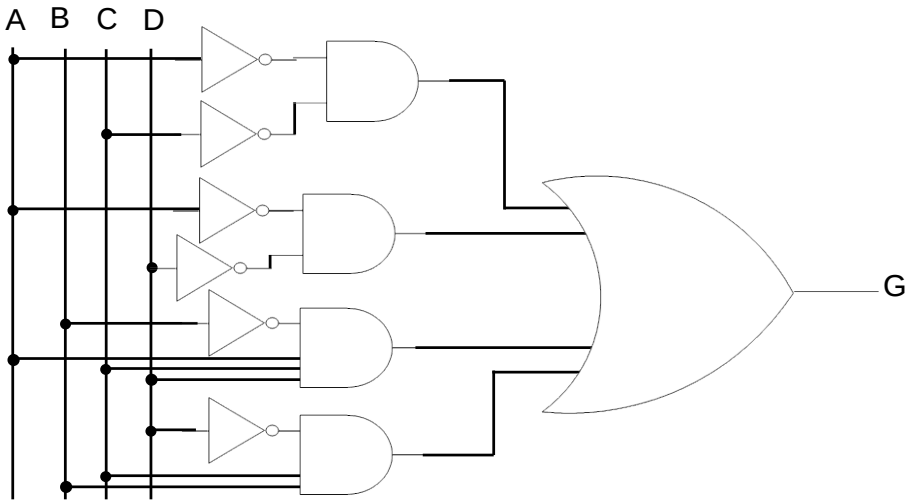
(a)

(i)

A \ CD	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	0	0	1
10	0	0	1	0

(ii) $G \equiv (\sim A \wedge \sim C) \vee (\sim A \wedge \sim D) \vee (A \wedge \sim B \wedge C \wedge D) \vee (B \wedge C \wedge \sim D)$

(iii)



Parts i and iii were generally answered well, but relatively few students obtained full marks for part ii.

A common mistake was to describe the top-left grouping of the K-map as ~(A ∧ C).

(b) (i) Flip-flops can be found in the ALU as part of the registers.

(ii) A set of flip-flops are used to store binary numbers in the register. A flip-flop is used to store the binary digit (0 or 1) for each place in the number.

(c) When the instruction is reached in the program it will be transferred from the memory to the Control Unit. The Operation Code (2) is the instruction which tells the Control unit to use the ALU to add the registers. The two source (s and t) and destination (d) registers numbers are sent from the control unit to the ALU which uses this information to add the two source registers s and t and place the result in register d.

10b and 10c were not attempted by most students and poorly answered by many who did.

Question 12

(a) (i)

R[1]	R[3]	R[4]	R[6]
1	3	7	0
	2		7
	1		14
	0		21

The final value of location 06 is 21.

(ii)

```
int a = 3;
int b = 7;
int c = a * b;
```

Generally well answered by all students, although a number who successfully completed the trace were not able to identify the operator.

(b)

Memory Address	Contents	Pseudocode	Explanation
01	0001	data	($0000\ 0000\ 0000\ 0001_2$, 1_{10})
03	0003	data	($0000\ 0000\ 0000\ 0011_2$, 3_{10})
04	0007	data	($0000\ 0000\ 0000\ 0111_2$, 7_{10})
06	0000	data	($0000\ 0000\ 0000\ 0000_2$, 0_{10})
07	0000	data	($0000\ 0000\ 0000\ 0000_2$, 0_{10})
10	8101	R[1] ← mem[01]	Set register 1 to the contents of location 01
11	8303	R[3] ← mem[03]	Set register 3 to the contents of location 03
12	8404	R[4] ← mem[04]	Set register 4 to the contents of location 04
13	8606	R[6] ← mem[06]	Set register 6 to the contents of location 06
14	8704	R[7] ← mem[04]	Set register 7 to the contents of location 04
15	2773	R[7] ← R[7] - R[3]	Set register 7 to register 7 - register 3
16	D719	if (R[7] > 0) goto 19	If register 7 greater than 0 goto 19
17	C719	if (R[7] == 0) goto 19	If register 7 equals 0 goto 19
18	C01B	goto 1B	Goto 1B
19	1661	R[6] ← R[6] + R[1]	Set register 6 to register 6 + register 1
1A	C015	goto 15	Goto 15
1B	9606	mem[06] ← R[6]	Store register 6 into location 06
1C	1773	R7] ← R[7] + R[3]	Set register 7 to register 7 + register 3
1D	9707	mem[07] ← R[7]	Store register 7 into location 07
1E	0000	halt	

R[1]	R[3]	R[4]	R[6]	R[7]
1	3	7	0	7
			1	4
			2	1
				1

Part 12b was particularly challenging for most students, with very few getting it entirely correct. Of the students who wrote more than a few lines, most demonstrated a solid understanding of TOY programming and the elements required to answer this question.

Section E

Question 13

(a)

	1	1	0	[1]	1
+	[0]	1	0	1	1
1	0	0	[1]	1	0

Many students made simple errors here. Partial credit was awarded whenever evidence of working out was shown.

(b) (i) The value $+109_{10} = 01101101_2$. This means -109_{10} is stored as $10010010 + 1 = 10010011_2$.

An **incredibly** common error here was to not include the leading 1 that made the value negative. Most correctly inverted the binary word 11011101_2 to 00100011_2 but did not give the final answer of 10010011_2

(ii) 0.11_2 is made up of $0.1_2 + 0.01_2 = 2^{-1} + 2^{-2} = 0.5 + 0.25 = 0.75_{10.}$

This question was very well answered by most students.

(c) (i) In the RGB system the colour **White** is stored as three numbers each stored in a byte with every bit set to 1. This means the three hexadecimal values will be FF, FF and FF. Giving a full 24bit colour for white as FFFFFFF is also an acceptable answer.

A common error for this question was to set three bytes to 1 and give 111, or to convert each RGB value to a 3-bit value (since the question mentions 3 numbers) and converted 111_2, 111_2, 111_2 to 777. An error that earned partial credit was to converted the RGB **4-bit** values of 1111_2, 1111_2, 1111_2 to FFF.

(ii) The least significant 4 bits of the ASCII code for the characters '0' to '9' stores the binary value of the corresponding digit. ie The least significant 4 bits of '0' is 0, The least significant 4 bits of '1' is 1 and so on.

Many students understood the question and were able to demonstrate some understanding and were awarded credit for this, but did not explain the concept clearly. For full marks it needed to be explicit that the least significant 4 bits of the ASCII code give the **binary** value for the digits 0-9. Examples of incorrect answers are "The pattern repeats" or "they all go up by one". Examples of unclear answers are "They are the numbers 0-9" or "they are the same as the digits with 48 added". Another common misunderstanding was to compare ASCII Char values 0-9 with the first 10 values in the ASCII table. Eg "0110 = 6 and 0110 = ACK so they are the same"
There was also a quite a number of students who misinterpreted the question, misunderstanding the term "least significant bits"

Question 14

(a) If the colour is stored as index of the colour table it will require 3 bits and to store the value of the card will be 4 bits. So the total bits to store information on a card is 7 bits.

Some students identified that there were 50 cards and thus it would require 6 bits to store an identifier for each one (since $2^5<50<2^6$) however this is incorrect because the question states explicitly that the colour index must be stored. Partial credit was awarded for this. There were many varied and confusing answers given here, with many students clearly not understanding the question. Some referred to ASCII characters

to store the value, some allocated 24 bits to store the colour, others converted each of the numbers 2-12 to binary and added them up.

(b) Arithmetic overflow is where a number is generated that is too large for the word storage space. 66 + 65 = 01000010 + 01000001 = 10000011. Positive numbers stored using two complement representation must have 0 for the most significant bit. Since the result of this addition has a most significant bit of 1 this is an example of integer overflow.

Many students understood this question and explained it clearly or did the binary addition and demonstrated it, which shows a good understanding of the problem. However, those who simply did the addition, got the result and pointed to the leading "1" with an arrow that said "Error" were not awarded full marks. The question clearly states "explain why" and it was expected that some descriptive words be included in the solution, such as explaining that the leading bit is reserved for the sign in two's complement.

(c) When setting up a format for a floating point number the decision needs to be made about how sharing the available bits between the mantissa and the exponent. That is how many bits to allocate to the mantissa and how many bits to allocate to the exponent. If the number of bits allocated to the mantissa is increased the precision of the number is increased. The more bits that are allocated to the exponent the greater the range of values that can be represented.

Partial credit was given for many respondants in this question who demonstrated some understanding of the concept but could not explain it clearly. The words "mantissa" and "exponent" were not required in the solution (although it helped) as some students were able to clearly describe the concept without using the key words. Many incorrect answers simply referred to the fact that floating point binary cannot accurately represent some terminating decimals.

Question 15

(a) (i) One disadvantage of lossy compression are that it takes longer to extract the image from the compressed file because it will need to be decompressed. The other disadvantage is that the lossy compression results in a loss of some detail from the file resulting in an inferior image.

Not many students gave two valid reasons. Many students stated something like loss of detail, loss of colour, loss of fidelity, lower picture quality or similar, and several stated the same thing again using different words. Very few mentioned the extra computing resources required to decompress the file.

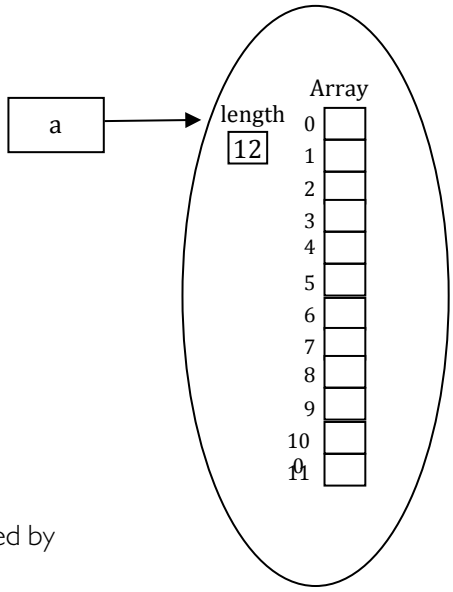
(ii) Storage requirement of original image is 200 x 200 x 3 x 8 = 960 000 bits.
Storage of compressed image is 200 x 200 x 8 + 256 x 3 x 8 = 320000 + 6144 = 326144 bits
Each pixel is represented by an 8 bit index to the lookup table. The lookup table is made up of 256 RGB values so is 256 x 3 x 8.

Most students were able to successfully calculate the size of the uncompressed image (although quite a few divided by 8 instead of multiplying by 8 for the bits per channel). Most calculated 320,000 bits for the compressed image although very few added on the size of the lookup table in the image.

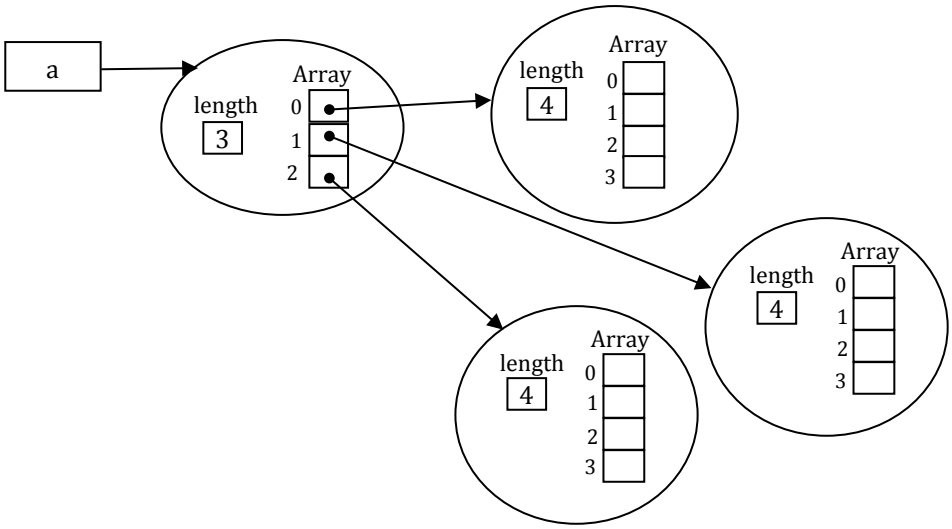
(iii) The compression method will only be lossy if the original image contained more than 256 colours. If this was the case the number of colours in the image would be reduced to 256 resulting in a loss of image detail.

To demonstrate full understanding here, the response needed to include a mention of the fact that if the original only had 256 colours, no fidelity would be lost. Most students here just reiterated their answer from 15(a)(i) by saying "loss of colour depth" or "Lower quality" or "less fidelity" or "poorer image quality". Any specific reference to colour got partial credit here.

(b) The array int[12] can be represented by



The array int[3][4] can be represented by



To get the number of words required, count up how many little boxes are in each diagram.

The number of words required for the int[12] is 12 + 2 = 14.

The number of words required for the int[3][4] is 3 x 5 + 4 + 1 = 20.

The storage requirements are greater at 20 words compared with 14 but it not double the amount.

Very few students got full credit here, but many made admirable attempts. It was relatively easy to get partial credit here because the question asks students explicity to draw an array diagram which can be copied directly out of the information booklet with only simple adjustments. To draw the diagram accurately was worth half the marks of the question, but clearly students are not as familiar with the information booklet as they should be, and many did not draw an acceptable diagram.

Once the diagram was drawn, many students did not count up the data values correctly and did not come to correct conclusions. A very common error here was to count up the bits or the bytes required for the arrays. If done correctly, full marks were awarded but most students were hindered by this complex and irrelevant process.