COMPUTER SCIENCE (ITC315113)

EXAMINER COMMENT

Section B and some other parts of the exam were generally answered well with many students attaining ‘A’ ratings. The markers were disappointed, however, with overall student performance on what we thought was a straightforward exam. Although the exam contained few surprises, many students were not able to demonstrate the ‘C’ rating skills required in this course. This was particularly evident in Sections D and E where the same style of question is repeated each year. We would advise teachers and students to spend more time analysing algorithms, tracing Java code and revising for sections D and E.

SOME GENERAL COMMENTS

Read the questions carefully.

When writing algorithms, use indenting conventions.

For any questions that involve calculations, show your working. You should do this even if the question doesn’t explicitly request it. It is especially important to show your working in the ‘A’ standard questions of Sections B, D and E. It is difficult to award part marks for an incorrect answer without some explanation of how you obtained that answer.

SECTION A

QUESTION 1

(a) (i) The lowest cost of a pizza is $11.

(ii) You can have up to 3 topping without increasing the cost of the pizza.

(b) Code to set upper limit of 7 for number of toppings.

(c) Code to implement stuffed crust.

Initially
set toppings = 1
set stuffed = 0
set cost = 0

When the “More Toppings” button is pressed
set toppings = toppings + 1
if toppings equals 8
    set toppings = 7
    display “Number of toppings” toppings

When the “Less Toppings” button is pressed
set toppings = toppings – 1
if toppings equals –1
    set toppings = 0
    display “Number of toppings” toppings
When the “Stuffed Crust” button is pressed
    set stuffed = 3
    display “stuffed crust”

When the “Not Stuffed Crust” button is pressed
    set stuffed = 0
    display “no stuffed crust”

When the “Calculate” button is pressed
    set extra_toppings = 0
    if toppings greater than 3
        set extra_toppings = toppings - 3
        set cost = 11 + extra_toppings * 2 + stuffed
    display “Cost of pizza is $" cost

Examiner comment
Question 1 was done well by most students. The only problem came in part C where many attempted to adjust
the cost within the When the “Stuffed Crust” button is pressed method. It is best to do this within the calculate
section as repeated pressing of the When the “Stuffed Crust” button can lead to unwanted consequences if it is
not structured correctly.

QUESTION 2

(a) The sum of the card values has a sum of 8 for the player to receive the maximum 400 points

(b) Code to have all other possible card values give a score of 0.

(c) Code to allow only one chance to swap a card for a new one.

Initially
    set card_number = 1
    set card_1 to random value selected from 1, 2 or 5
    set card_2 to random value selected from 1, 2 or 5
    set card_3 to random value selected from 1, 2 or 5
    set score = 0
    set swapped = false
    display “card 1:” card1 " card 2: card 2 * card 2 * card 3:” card3

When a number is entered into the “Card Number” Textfield
    if swapped equals false
        set swapped = true
        set card_number to value in “Card Number” Textfield
        if card_number equals 1
            set card_1 to random value selected from 1, 2 or 5
        if card_number equals 2
            set card_2 to random value selected from 1, 2 or 5
        if card_number equals 3
            set card_3 to random value selected from 1, 2 or 5
        display “card 1:” card1 " card 2: card 2 * card 2 * card 3:” card3
    else
        display “Card can only be swapped once.”
When the “Score” button is pressed
set score = 0
set sum = card1 + card2 + card3
if sum equals 8
  set score = 400
if sum equals 3, 6 or 15
  set score = 200
display "card 1:" card1 " card 2:" card2 " card 3:" card3 " and Score is : " score

Examiner comment
Question 2 was not well done although part (a) was obvious for most students.
In part (b) many students just put else set score =0 after set score=200. This doesn’t work unless you put an else before if sum equals 3, 6 or 15. There were many ways to do this with lots of students using sum not equal 3,6,15,8. Those who listed the other possible combinations needed to be careful that they listed all possibilities!
In part (c) many students used a Boolean (or number) to tag when the cards had not been swapped but then failed to change its value once a swap was made (allowing further swaps). Others did change it but then used an if statement to state that a change had already occurred without linking it (with an else) to the original if statement.
The correct use of indentation is always useful in this section, both to students and the marker.

**QUESTION 3**

Enter minutes: [ ]
Enter hour: [ ]
Enter Day: [ ]
Time taken will be: [ ]

Initially
set minutes = 0
set hour = 7
set day = Monday

When a number is entered into the “Minutes” Textfield
set minutes to value in “Minutes” TextField

When a number is entered into the “Hour” Textfield
set hour to value in “Hour” TextField

When a number is entered into the “Day” Textfield
set day to value in “Day” TextField
When the "Calculate" button is pressed
  if hour is 8 or 9
    set bus_ minutes = 15
  else
    set bus_ minutes = 10
  if day equals Monday or Tuesday
    set bus_ minutes = bus_ minutes + 3
  if minutes equals 0
    set duration = minutes
  else
    if minutes less than or equal to 20
      set duration = 20 - minutes
    else
      if minutes less than or equal to 40
        set duration = 40 - minutes
      else
        set duration = 60 - minutes
  set duration = duration + bus_ minutes
  display "Time taken will be: " duration

Examiner comment
Question 3 was not well done with many students not attempting the question or not reading it carefully enough. Very few students realised that, as well as the time required to travel on the bus, there would also be a waiting time before it arrived depending upon when the student arrived at the stop. This made it important to separate the hour and the minutes into 2 variables.

Successful students tended to use an extended sequence of if statements to calculate the bus_minutes without observing the pattern used in the algorithm above. The marker was pleased to see the use of arrays to solve this problem as it does make the code easier to edit if the data changes.

An alternate way (to the if structure in the solution) of finding the waiting duration was 20-minutes%20 although arriving at exactly 00, 20, 40 does cause problems!

(If minutes%20=0 duration =0 else duration=20-minutes%20)
SECTION B

QUESTION 4

(a) (i) \( a = 9.5 \)
(ii) \( b = 2 \)
(iii) \( c = "25" \)

(b) (i) \( d = 2 \)
(ii) \( e = 19 \)

\[
\begin{array}{c|c}
 i & e \\
\hline
 10 & \\
 2 & 12 \\
 3 & 15 \\
 4 & 19 \\
\end{array}
\]

(iii) \( f = 2 \)

\[
\begin{array}{c}
 f \\
 20 \\
 10 \\
 5 \\
 2 \\
\end{array}
\]

(c) \( h = 115 \)

\[
\begin{array}{c|c|c}
 t & u & h \\
\hline
 3 & 6 & 100 \\
 4 & 104 & \\
 5 & 109 & \\
 6 & 115 & \\
\end{array}
\]

Examiner comment
4a (i) The most common mistake was the incorrect order of operations.
4a (ii) Some students forgot what \( \% \) (modulus) is or does.
4a (iii) Confusion whether to treat as integers (addition) or text (concatenation of strings). Some students tried converting to ASCII values.
4b (i) This was the best-answered question in the entire section, with 86\% of students getting the answer right. Those that didn’t generally failed to trace through the second if statement.
(ii) A considerable number of students began the trace with i=2 and e=10 on the SAME line, resulting in answers that were out of sync with the for loop.

(iii) Many students kept tracing code after while condition no longer valid (perhaps to fill the table, or perhaps a confusion with int/rounding decimals)?

Again, a sizeable number of students erroneously continued the trace using a loop on the 'if' statement to fill all boxes in the table, rather than exiting when the 'while' loop ended.

**QUESTION 6**

(a) (i) **word1**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>'P'</td>
<td>'A'</td>
<td>'Z'</td>
<td>'Z'</td>
<td>'A'</td>
<td></td>
</tr>
</tbody>
</table>

(a) (ii) **word2**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>' '</td>
<td>' '</td>
<td>' '</td>
<td>' '</td>
<td>' '</td>
<td>' '</td>
<td>' '</td>
</tr>
</tbody>
</table>

(ii) **letters**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>'E'</td>
<td>'A'</td>
<td>'I'</td>
<td>'O'</td>
<td>'U'</td>
<td>'S'</td>
<td>'I'</td>
</tr>
</tbody>
</table>

(iv)

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>word2</th>
<th>found</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>A</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(b) (i) The “Find failed” string will be displayed if none of the letters in length6 array are in the word1 array.

(ii) The scope of the variable found is inside the method find.

(iii)

No.

Letters = length6;

would change the formal parameter letters to point to the same array as length6 but would not change the actual parameter which is nul. This means the method selectLength would do nothing.

Examiner comment
6a (i), 6a (ii), 6a (iii) - These questions were generally well answered by those who attempted them (in some instances, even students who were scraping the 'C'-levels answered successfully).

6a (iv) Proved far more challenging for most students. Many students attempting this question were able to partially trace the code, but often missed the second 'A', or added another letter found/true.

6b (i) and 6b (ii) generally well answered when attempted.

6b (iii) proved significantly more difficult to answer, with a wide variety of responses and explanations.

SECTION C

QUESTION 7

(a) (i)

openA: Y _______
openB: N _______
openC: N _______

(ii)

openA: N _______
openB: N_______
openC: N _______

(iii) Press the buttons in the order of BoxA then BoxC then BoxB.

(b) (i) “oneness”

(ii) Counting on it=14

(iii) SHOUT OUT!
Examiner comment
Part (a) was answered poorly by students, mostly because they struggled to navigate the nested if statements. Many were not able to trace the algorithm correctly, and thus not able to demonstrate basic understanding of how text in a textbox may be changed.

Part (b) was generally answered poorly. Students did not know the technical details of how the substring method, the length method, or the replace method worked, even though they are stated clearly in the Information booklet. The concept of a method acting on a method (as in toUpperCase().replace(...) caused confusion in what order the methods are called and what is getting made upper case.

QUESTION 8

(a) (i)
Student student1 = new Student("Mike");
Student student2 = new Student("Sophie");

(ii) The values of the properties test1 and test2 are -1 in the initialised variables.

(iii)
student1.setTest1(52);
student1.setTest2(65);
student2.setTest1(56);
student2.setTest2(68);

(b)
g.drawString(student1.getName()+ " Average: "+student1.average()+"%", 10, 10);
g.drawString(student2.getName()+ " Average: "+student2.average()+"%", 10, 30);

(c) int highestScore = student2.highAverage(student1.average());

Examiner comment
Students were either able to give a good answer to all parts of this question, or struggled to get many correct at all. If they understood the use of objects then they were able to correctly use the constructor (part (a)i), interpret the constructor code (part (a)ii) and use the setTest1 and setTest2 methods correctly (part (a)iii). Aside from completely nonsensical declarations, errors usually involved declaring two variables with the same identifier (both called “studentname” for example) and calling the method incorrectly in part (a)iii.

Part (b) was done well by those who understood how to call a method, but marks were lost for not displaying the result. Many who didn’t understand the question wrote code to calculate the average by adding the two test marks and dividing by two.

Part (c) was done poorly, with many students not understanding how to pass the result of a method into another method, or not understanding that it is what was required.
public class Subject {
    public String name;
    public int[] criterion = new int[5];

    public Subject(String newName) {
        name = newName;
        for(int i = 1; i <= 4; i++)
            criterion[i] = 0;
    }

    public String getName() {
        return name;
    }

    public int setResult(int criterionNumber, int result) {
        // return criterion[criterionNumber] = result;
    }

    public char award() { // missing method definition
        int sum = 0;
        for(int i = 1; i <= 4; i++)
            sum = sum + criterion[i];
        if (sum < 5)
            return 'N';
        else
            if (sum < 12)
                return 'P';
            else
                return 'C';
    }
}

Examiner comment
This question was generally answered well by those who attempted it. Many students chose not to use an array, which is acceptable (though not as efficient), but rather have an int variable for each criterion. If this approach was used, a switch statement or a series of nested IFs was required when defining a setRating method. A common error was to have four separate setRating methods, one for each criterion, which is bad code AND against the requirements of the question.

There was also much confusion about how the getAward method would function. Some students set an “award” variable to N, P or C without returning anything. Others stated the function return type was “void” and then proceeded to return a variable. Generally, in this question there was a lot of inconsistency between defined return types and the actual appearance of “return Something” in the method.

Another common error was to have the getAward method take four ints as parameters, and calculate the award from those four, rather than referring to the four ratings that are already stored as variables.
SECTION D

QUESTION 10

(a) (i) \[ E \equiv A \land (B \lor (\neg A \land \neg B)) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>\neg A</th>
<th>\neg B</th>
<th>\neg A \land \neg B</th>
<th>B \lor (\neg A \land \neg B)</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(ii) \[ E \equiv A \land B \]

Examiner comment
Generally answered well.

(b) (i)

(ii) \[ G \equiv \neg((P \land \neg Q) \lor (Q \land R)) \]

Examiner comment
A common mistake in part (i) was to put one or more of the NOT gates in the wrong position. Another source of error came from attempting to rearrange the expression before drawing the logic circuit. When drawing the logic circuit for an expression, students should not rearrange the formula unless asked to do so.

In part (ii), the most common mistake was to omit the outermost set of brackets.

(c) Multiplication is done using repeated addition: i.e. \[ 3 \times 5 = 5 + 5 + 5 \].

Examiner comment
Generally answered well, even by students who struggled with the rest of Q10. A number of students explained how this would work in TOY code.
QUESTION 11

(a) (i)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

(ii) \[ F \equiv (\neg A \land \neg B) \lor (\neg C \land \neg D) \lor (\neg A \land D) \lor (B \land \neg C) \]

Examiner comment
Part (i) was answered very well.

In part (ii), there was a wide variety of answers, including many correct, equivalent expressions. Relatively few students seemed to notice that the K-map could be broken up into four overlapping clusters of four elements.

Having given a correct answer, some students introduced errors attempting to unnecessarily simplify their expressions. For example, by treating \( \neg (A \land B) \) as equal to \( \neg A \land \neg B \).

(b) In the branch zero instruction if \( R[d] == 0 \) pc \( \leftarrow \) addr, “pc” is the program counter register. It holds the address of the instruction to be executed. It is used by the Control Unit to locate the instruction to be loaded in the program instruction register from the program stored in the Memory Unit.

Examiner comment
Relatively few students were able to answer this correctly. Despite not knowing what “pc” stood for, however, some students were able to demonstrate understanding of role of the program counter.

(c) The values stored in registers \( s \) and \( t \) are transferred by the Bus to the ALU.

The ALU then adds the values together.

The sum is then sent by the Bus from the ALU to be stored in the register \( d \).

Examiner comment
This part was either not answered or poorly answered by most students.
QUESTION 12

(a)  

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Contents</th>
<th>Pseudocode</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0000</td>
<td>data</td>
<td>Used for variable i</td>
</tr>
<tr>
<td>02</td>
<td>0000</td>
<td>data</td>
<td>Used for variable j</td>
</tr>
<tr>
<td>10</td>
<td>7303</td>
<td>R[3] ← 03</td>
<td>Set register 3 to value 03</td>
</tr>
<tr>
<td>11</td>
<td>7505</td>
<td>R[5] ← 05</td>
<td>Set register 5 to the value 05</td>
</tr>
<tr>
<td>12</td>
<td>7909</td>
<td>R[9] ← 09</td>
<td>Set register 9 to the value 09</td>
</tr>
<tr>
<td>15</td>
<td>D213</td>
<td>if (R[A] &gt; 0) goto 15</td>
<td>If register A greater than 0 goto 15</td>
</tr>
<tr>
<td>16</td>
<td>9301</td>
<td>mem[01] ← R[3]</td>
<td>Store register 3 into location 01</td>
</tr>
<tr>
<td>17</td>
<td>9502</td>
<td>mem[02] ← R[5]</td>
<td>Store register 5 into location 02</td>
</tr>
<tr>
<td>1A</td>
<td>0000</td>
<td>halt</td>
<td></td>
</tr>
</tbody>
</table>

(b)  

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td></td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Each fetch/decode/execute cycle involves reading from memory.

Given that program execution starts at line 10, and the loop from lines 13 to 15 runs twice, there are 3 + (3 × 2) + 3 = 12 fetch/decode/execute cycles in total.

Additionally, two instructions involve writing to memory. (These are indicated by the two circles in the program trace).

Because read/write times for the CPU registers can be considered 0ns, any instructions that don’t involve read/write to main memory can be ignored.

Therefore, total time for read/write is 12 × 100 ns = 1200 ns.

Examiner comment
A significant proportion of students didn’t attempt any part of Question 12. Of the students who wrote more than a few lines of code for 12a, most were able to demonstrate their understanding of TOY programming and the elements required to answer the question. The trace of their program in 12b was invaluable for understanding how their code was intended to work.

12c was particularly challenging for all students. Of those who attempted it, many simply wrote a number. To obtain even partial marks on this question, students were expected to provide some explanation of their thinking.
SECTION E

QUESTION 13

(a)

\[
\begin{array}{cccccc}
1 & 1 & 0 & 0 & 1 \\
+ & 1 & 0 & 0 & 0 \\
\hline
1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

Examiner comment
Despite this style of question being regularly asked, some students still had difficulty with this question.

(b) (i) The calculation is done as 103 + (−91). The value of −91 is given by the Twos Complement of 91 = 10100101. Then they are added ...

\[
\begin{array}{cccccc}
0 & 1 & 1 & 0 & 0 & 1 \\
+ & 1 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

The highest bit in the calculation is lost because it is an 8 bit word thus the answer is 00001100 which is equal to 12.

Examiner comment
Many students described the process without working through with the numbers provided.

(ii) The conversion of 2A3D\text{16} to binary is 0010 1010 0011 1101\text{2}.

Examiner comment
Generally, well answered although some tried to convert to decimal.

(iii) The largest value in an 8 bit Twos Complement representation is 01111111. This is because this representation requires the most significant bit to be 0 to identify it as a positive value. The other bits add up as: \(2^7+2^6+2^5+2^4+2^3+2^2+2^1 = 2^7-1 = 127\)

Examiner comment
Well answered.
(c) (i) Characters such as the letter ‘A’ are represented in the computer using a standard code number. An example of a standard code is Unicode in which ‘A’ has the code 65. In this case ‘A’ would be represented by the binary number 10000012 in the computer.

Examiner comment
The key part of this answer was the use of a code system. A few answers suggested Hexadecimal rather than ASCII or Unicode.

(ii) The minimum number of bits required to store a boolean value is 1 bit.

Examiner comment
Generally, well answered although a number suggested 8 bits was the minimum

QUESTION 14

(a) The two digit number will have values from 00 to 99 which is 100 values and that is less than 127 = 27 – 1 so will require 7 bits. Each letter will require a value between 0 and 25 and so is 26 values and that is less than 31 = 25 – 1 and so will need 5 bits. So in total the minimum number of bits would be 7 + 3 x 5 = 22 bits per registration plate.

Examiner comment
Answers ranged from 5 bits up to 2000 bits. Most answers used five bits for a letter and four bits for a digit, giving 23 bits. This was marked as correct.

(b) 0.9 x 2 = 1.8
0.8 x 2 = 1.6
0.6 x 2 = 1.2
0.2 x 2 = 0.4
0.4 x 2 = 0.8
0.8 x 2 = 1.6
0.6 x 2 = 1.2
0.2 x 2 = 0.4

In a floating point number that can represent 15 bits the mantissa for 0.9 would be: most significant bit → [1] 1001 1000 1001 1

Examiner comment
Some students attempted to show a full representation using a sign, exponent and mantissa. They incorrectly pointed to the sign bit as being the most significant bit.

(c) (i) With a signed integer representation half the values represented are negative. So if you only want to represent positive values then the unsigned representation would give you twice as many values for the same number of bits.

Examiner comment
Many students recognised an increased number of positive values but didn’t realise it doubled.
(ii) When colours are represented they use the RGB system which is three bytes each storing unsigned integers from 0 to 255.

Examiner comment
Most students gave an example of an unsigned integer, rather than a representation that uses unsigned integers. The students that used an example like $1101 = 13$ unsigned or $1101 = -3$ using 2's complement, were given full marks.

QUESTION 15

(a)
Data Block is $384 \times 16 = 6144$ bits
Sampling is 16 bits at 32 000 times a second (32 kHz)

1 minute = $16 \times 32000 \times 60 = 30720000$ bits

Number of data blocks = $30720000 / 6144 = 5000$

Total number of bits = $5000 \times (6144 + 28) = 5000 \times 6172 = 30860000$

Examiner comment
Again, there were a range of answers given, from few bits to some GB. Often there was just a multiplication of the numbers given with no explanation of what was being calculated or why. Only a small number of candidates were able to arrive at the correct answer.

(b) (i) When 7.2 is converted to binary is has a recurring binary fraction this means it is not stored exactly. This means that $7.2 \times 50$ does not come to exactly 360.0 and so the loop condition is never met and so it is an infinite loop.

When 11.25 is used this is exactly represented in binary and so $11.25 \times 32 = 360.0$ this means the loop will stop when $c = 360.0$

Examiner comment
Many students mentioned that doubles were not stored accurately and could lead to errors. Good answers recognised that 11.25 had an exact representation where 7.2 didn’t.

(ii) To fix this problem the loop condition should change to ($c < 360$).

Examiner comment
Some answers specified a range within which $c$ could approach 360. Some of the more novel solutions included suggesting that different numbers could be used, or change the types to float or int. One even suggested that the code be replaced with $c=360$;