N13/5/COMSC/SP2/ENG/TZ0/XX/M

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

MARKSCHEME

November 2013

COMPUTER SCIENCE

Standard Level

Paper 2

10 pages

Subject Details: Computer Science SL Paper 2 Markscheme

Mark Allocation

Candidates are required to answer ALL questions ***[20 marks]*** for question 1, ***[20 marks]*** for question 2 and ***[30 marks]*** for question 3. Maximum total = ***[70 marks]***.

General

A markscheme often has more specific points worthy of a mark than the total allows. This is intentional. Do not award more than the maximum marks allowed for that part of a question.

When deciding upon alternative answers by candidates to those given in the markscheme, consider the following points:

- Each statement worth one point has a separate line and the end is signified by means of a semi-colon (;).
- An alternative answer or wording is indicated in the markscheme by a "/"; either wording can be accepted.
- Words in (…) in the markscheme are not necessary to gain the mark.
- If the candidate's answer has the same meaning or can be clearly interpreted as being the same as that in the markscheme then award the mark.
- Mark positively. Give candidates credit for what they have achieved, and for what they have got correct, rather than penalizing them for what they have not achieved or what they have got wrong.
- Remember that many candidates are writing in a second language; be forgiving of minor linguistic slips. In this subject effective communication is more important than grammatical accuracy.
- Occasionally, a part of a question may require a calculation whose answer is required for subsequent parts. If an error is made in the first part then it should be penalized. However, if the incorrect answer is used correctly in subsequent parts then **follow through** marks should be awarded. Indicate this with "**FT**".

1. (a) *Award **[1 mark]** for each correct row.*

outArray	[0]	[1]	[2]	[3]	[4]	[5]
Original	0	0	0	0	0	0
i = 1	0	4/3	0	0	0	0
i = 2	0	4/3	3	0	0	0
i = 3	0	4/3	3	7/3	0	0
i = 4	0	4/3	3	7/3	2/3	0

[4 marks]

(b) *Award up to **[1 mark max]**.*
each element in the output array is the average of the corresponding element and its (immediate) neighbours in the input array;
calculates a 3-point moving average;
(Accept that the values in the original array `data` *are left unchanged;)* ***[1 mark]***

(c) The average for an element is computed using the element before it – this would cause an out-of-bounds condition if you tried to compute it for the first element;
The same problem would occur if you tried to compute an average for the last element – you need the element after it to compute the average; ***[2 marks]***

(d) *Award marks as follows:*
*Award **[1 mark]** for a correct modification to compute the first element;*
*Award **[1 mark]** for a correct modification to compute the last element;*

For example:

Insert (either before or after the loop):

```
outArray[0] = (inArray[0] + inArray[1]) / 2.0;
outArray[n-1] = (inArray[n-2] + inArray[n-1]) / 2.0;
```

Note: can also be carried out within the loop using "if" statements. ***[2 marks]***

(e) *Award marks as follows:*
*Award **[1 mark]** for including the correct elements in the same row;*
*Award **[1 mark]** for including the correct elements in the same column;*

For example:

```
outData = (data2[1][4] + data2[1][3] + data2[1][5] +
data2[0][4] + data2[2][4]) / 5.0;
```

[2 marks]

Question 1 continued

(f) *Award marks as follows:*
*Award **[1 mark]** for calculating the number of rows;*
*Award **[1 mark]** for calculating the number of columns;*
*Award **[1 mark]** for use of nested loops;*
*Award **[1 mark]** for correct start/stop values in each loop;*
*Award **[1 mark]** for calculating the average correctly;*
*Award **[1 mark]** for returning the new array;*

```
public double[][] doIt2(double[][] in2Array)
{
  int numRows = in2Array.length;
  int numCols = in2Array[0].length;
  double[][] out2Array = new double[numRows][numCols];

  for(int i=1; i< numRows-1; i++)
  {
    for(int j=1; j< numCols-1; j++)
    {
      out2Array[i][j] = out2Array[i][j] + (in2Array[i-1][j] +
      in2Array[i+1][j]+ in2Array[i][j-1] + in2Array[i][j+1]) /5;
    }
  }
return out2Array;
}
```

[6 marks]

(g) Noting that the corners have 2 neighbouring elements that could be averaged;
Noting that the edges have 3 neighbouring elements that could be averaged;
A clear discussion that outlines how this could be implemented; ***[3 marks]***

2. (a) (i)
```
FarmField firstfield = new FarmField();
```
[1 mark]

(ii)
```
firstField.fieldSize = 12000;
firstField.fieldName = "Back forty";
```
[2 marks]

(iii)
```
firstField.soilType = 8;
firstField.cropType = 2;
```
[2 marks]

(b) *Award marks as follows:*
*Award **[1 mark]** for looping over through the* `allFields` *array;*
*Award **[1 mark]** for getting the loop start/stop values correct;*
*Award **[1 mark]** for correctly accessing the field size;*
*Award **[1 mark]** for correctly finding the largest field;*

```
int findLargest(FarmField[] allFields, int numberFields)
  {
    int largestField = 0;
    int iField;

    for (iField = 0; iField < numberFields; iField++)
    {
      if (allFields[iFields].fieldSize > largestField)
      {
        largestField = allFields[iFields].fieldSize;
      }
    }
    return largestField;
  }
```
[4 marks]

Question 2 continued

(c) *Award marks as follows:*
*Award **[2 marks]** for correctly finding the insertion point;*
***or [1 mark]** for a credible attempt to find the insertion point;*

*Award **[1 mark]** for creating temporary variables to hold field objects during shifting;*

*Award **[3 marks]** for correctly implementing the shifting of the elements;*
***or [2 marks]** for credibly shifting all the elements that need to be shifted;*
***or [1 mark]** for credibly shifting an element down;*

*Award **[1 mark]** for handling the special case of not needing to shift any elements;*

```
public static void insertField (FarmField newField)
{
     int n=0;
     while (allFields[n] != null)   // find length of array
     {
       n++;
     }
     int size = newField.fieldSize;

     int i = 0;
     while (allFields[i] != null && allFields[i].fieldSize > size)
     {
       i++;                         // find insertion position
     }

     for(int x = n+1; x > i; x--)   //move up 1 place
     {
       allFields[x] = allFields[x-1];
     }
     allFields[i] = newField;       //insert new object
}
```
[7 marks]

(d) *Award marks as follows up to **[4 marks max]**.*
*Award **[1 mark]** for identifying an advantage;*
*Award **[1 mark]** for identifying a disadvantage;*
*Award **[1 mark]** each up to **[2 marks max]** for elaborating and describing;*

Advantages:
There is space to insert additional `FarmField` objects;
Avoids creating a new (smaller array) when deleting a `FarmField` object;

Disadvantages:
Cannot use the `length()` method to determine how many `FarmField` objects there are;
Code for insert and delete is complex;
Memory wasted;
[4 marks]

3. (a) *Award marks as follows up to **[4 marks max]**.*
*Award **[1 mark]** for identifying the precaution;*
*Award **[1 mark]** for describing the precaution beyond simply stating it;*
*For **two** precautions.*

For example:

Assign a password to the device;
So it cannot be accessed by an unauthorized user;

Do not store sensitive data on the smartphone;
So that it is not there to be found if the device is lost;

Do not store username/password account logins (or "remember me" cookies) on the device;
So somebody finding the device will not be able to access your accounts; ***[4 marks]***

(b) *Award marks as follows up to **[4 marks max]**.*
*Award **[1 mark]** for identifying the reason;*
*Award **[1 mark]** for an elaboration;*
*For **two** reasons.*

For example:

Uses a smaller battery;
So weighs less;

Can be marketed as green;
Increases sales;

Will charge up when outside;
Less chance it will run out of power; ***[4 marks]***

(c) They change frequencies several times a second;
Limited range;
Addressing (not pairing); ***[3 marks]***

(d) *Award up to **[3 marks max]**.*
Bluetooth is much slower than Ethernet;
Ethernet requires a physical connection, Bluetooth does not;
Ethernet is more secure than Bluetooth;
An Ethernet network requires additional hardware (routers, hubs, *etc*). Bluetooth does not; ***[3 marks]***

Question 3 continued

(e) *Award **[1 mark]** for each advantage up to **[2 marks max]**.*
*Award **[1 mark]** for each disadvantage up to **[2 marks max]**.*
*Award **[2 marks]** for a discussion that does more than state them.*

Advantages:
Emergency services can locate a distress call;
Friends can see where you are and meet you;
Advertisements could be tailored for things that are near you;

Disadvantages:
Illegal surveillance is possible;
Advertisers could profile your physical habits as well as your online ones;
Loss of location information (*eg* in a tunnel) could create erroneous data in systems; ***[6 marks]***

(f) (i) Flash memory, or EEPROM, or hard-drive; ***[1 mark]***

(ii) Cache memory is much faster than secondary memory (*eg* a hard-drive);
Data that will not fit in RAM but will be needed soon is put into cache;
And loaded back into memory when needed; ***[3 marks]***

Question 3 continued

(g)

Marks	Details
1–2	The candidate conveys some understanding of at least one effect of widespread ownership but merely states it/them without elaboration or example.
3–4	The candidate clearly identifies two or more effects and elaborates on one of them to explore at least one consequence of this effect.
5–6	The candidate clearly identifies and elaborates on the consequences of two or more effects.

Answer to be focused on students. Examples include:
Positive:

- Contact teachers with queries
- Research topics
- Replace expensive textbooks
- Make study schedules
- Store PowerPoint presentations

Negative:

- Become a distraction in class
- Reliance on smartphones as knowledge source
- Plagiarism issues
- Cheating in exams

[6 marks]