M11/5/COMSC/HP2/ENG/TZ0/XX/M

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

MARKSCHEME

May 2011

COMPUTER SCIENCE

Higher Level

Paper 2

13 pages

General Marking Instructions

Subject Details: Computer Science HL Paper 2 Markscheme

Mark Allocation

Candidates are required to answer ALL questions ***[20 marks]*** for question 1, ***[20 marks]*** for question 2, ***[20 marks]*** for question 3 and ***[40 marks]*** for question 4. Maximum total = ***[100 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 penalising them for what they have not achieved or what they have got wrong.
- 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.

1. (a) *Award up to **[3 marks max]**.*
Encapsulation;
Data hiding;
This data cannot be (directly) changed / accessed outside of this class (as the data is "private");
As data can only be changed via the transformer methods / viewed using the accessor methods;
Thus preventing accidental alteration of data;
Creates abstraction of the object;
Enhances security / protects the integrity of the data; ***[3 marks]***

(b) *Award marks as follows up to **[3 marks max]**.*
*Award **[1 mark]** for signature line with 2 parameters;*
*Award **[1 mark]** for of type String;*
*Award **[1 mark]** for correct assignments (with or without `this`) – allow use of transformer methods);*

Example answer:

```
public Seat(String x, String y)
{
  this.seatName = x;
  this.passengerName = y;
}
```
[3 marks]

(c) Two-dimensional array;
Of `Seat` objects;
With 4 rows and 10 columns / 4×10 / 10×4 ;

*Also, award **[3 marks]** for code that sufficiently shows the above, such as* `Seat[][] seatArray = new Seat[4][10];`. ***[3 marks]***

(d) *Award marks as follows up to **[5 marks max]**.*
*Award **[1 mark]** for initializing and incrementing the counter*;
*Award **[1 mark]** for correct nested loops*;
*Award **[2 marks]** for correct comparison, award only **[1 mark]** if accessor is not used (**do not allow** `if (seatArray[x][y] == null);`*
*Award **[1 mark]** for returning the counter*;

Example answer:

```
public int seatsRemaining()
{
  int count = 0;
  for (int x = 0; x < 10; x = x + 1)
  {
    for (int y = 0; y < 4; y = y + 1)
    {
      if (seatArray[x][y].getPassengerName() == null)
      {
        count = count + 1;
      }
    }
  }
  return count;
}
```
Indices can be reversed in parts (d) and (e). ***[5 marks]***

continued ...

Question 1 continued

(e) *Award marks as follows up to **[4 marks max]**.*
*Award **[1 mark]** for initializing the reply / returning "no seats" message;*
*Award **[1 mark]** for correct loop;*
*Award **[2 marks]** for correct comparison for both window seats, award only **[1 mark]** if accessor is not used;*
*Award **[1 mark]** for returning the seat name;*

***Note:** For parts (d) and (e) only penalise once if the accessor methods are omitted.*

Example answer:

```
public String freeWindowSeat()
{
  String reply = "No Seats Remaining";
  for (int x = 0; x < 10; x = x + 1)
  {
    if (seatArray[x][0].getPassengerName() == null)
    {
      return seatArray[x][0].getSeatName();
    }
    if (seatArray[x][3].getPassengerName() == null)
    {
      return seatArray[x][3].getSeatName();
    }
  }
  return reply;
}
```

Full marks can also be gained using a nested loop. ***[4 marks]***

(f) *Award up to **[2 marks max]**.*
A three-dimensional array could be used;
Assuming both levels held the same number of seats;
With the third dimension representing the level;
Accept code such as `Seat[][][] seatArray = new Seat[4][10][2];` *for **[2 marks]**.*

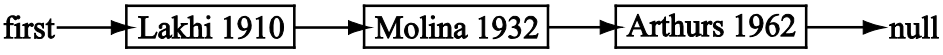
Making the array larger *e.g.* the number of columns could be increased;
To allow for the extra seats in the upper deck;

A second array could be used;
Especially for the upper deck;

*For any theoretically possible but improbable structure award **[1 mark]**.* ***[2 marks]***

Total: [20 marks]

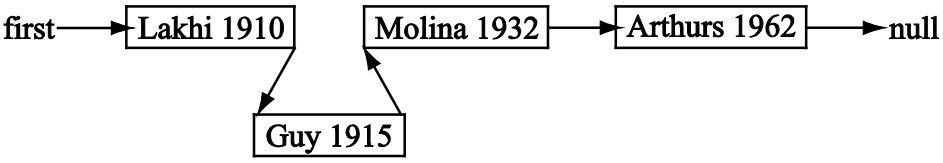
2. (a) *Award marks as follows up to **[2 marks max]**.*
*Award **[1 mark]** for correct position of node connected in some ways;*
*Award **[1 mark]** for pointers between nodes (don't accept plain lines);*
*Award **[1 mark]** for correct position of first (or equivalent) and null (or equivalent);*



[2 marks]

(b) *Award marks as follows if the student's meaning is clear either through the description or through the diagrams, up to **[2 marks max]**.*
*Award **[1 mark]** if the correct position is found;*
*Award **[1 mark]** if the previous node's next pointer is changed to point to the new node;*
*Award **[1 mark]** if the new node's next pointer points to the next node;*

***Note:** The pointers must be explicitly referred to in the description or clearly shown on the diagram.*



[2 marks]

(c) *Award marks either for an explanation or by showing the code.*
The constructor is called;
The data "Xiao", 1920, null is passed to the constructor;

```
Node newNode = new Node(Xiao, 1920, null);
```

[2 marks]

(d) *Award marks as follows up to **[2 marks max]**.*
*Award **[1 mark]** for testing for an empty list;*
*Award **[1 mark]** for removing the first node;*

```
public void removeLeast()
{
  if (first != null)
  {
    first = first.next;
  }
}
```

[2 marks]

continued ...

Question 2 continued

(e) *Award marks as follows up to **[8 marks max]**.*
*Award **[1 mark]** for use of `temp` variable (or equivalent);*
*Award **[1 mark]** for use of `previous` variable **(or equivalent)**;*
*Award **[2 marks]** for each of the two elements in the loop (as shown below);*
*Award **[1 mark]** for changing the value of `temp` to allow for traversal;*
*Award **[1 mark]** for creation of a new node (with any pointer value);*
*Award **[1 mark]** for identifying that there is more than one scenario (empty list / not empty list / first node etc.) and **attempting** to address them;*
*Award **[2 mark]** for correctly placing new node for all scenarios and **[1 mark]** for successfully placing new node for just 1 of the scenarios;*

Example answer 1:

```
public void addNode(String name, int year)
{
  Node temp = first;
  Node previous = null;

  while (temp != null && temp.yearOfBirth < year) // or <=
  {
    previous = temp;
    temp = temp.next;
  }

  Node newNode = new Node(name, year, temp); //create new node

  if (previous == null)  // if newNode becomes the first node
                         // or empty list
  {
    first = newNode;
  }
  else  // otherwise
  {
    previous.next = newNode;
  }
}
```

Notes:

- *Remember that we are testing the logic and not specifically the syntax.*
- *The year comparison can be correctly placed within the loop.*
- *There are alternatives to the `previous` variable (e.g. using `temp.next != null` where `temp` is equivalent to `previous`).*

continued ...

Question 2 continued

Example answer2:

```
public void addNode(String name, int year)
{
if (first == null)  // empty list
{
  first = new Node(name, year, null);
}
else
{
  Node temp = first;
  if (first.yearOfBirth > year) // 1st node in non-empty list
  {
    first = new Node(name, year, first);
  }
  else  // otherwise
  {
    while (temp.next != null)
    {
      if (temp.next.yearOfBirth > year)
      {
        break;
      }
      else
      {
        temp = temp.next;
      }
    }
    temp.next = new Node(name, year, temp.next);
  }
}
```

[8 marks]

(f) (i) BigO for removal is O(1);
Because the correct node is always pointed to by `first` / at the start of the list / the method is independent of the number of nodes / has 1 assignment only; ***[2 marks]***

(ii) BigO for addition is O(n);
Because the list might have to be traversed to the end / have to pass each of the nodes currently in the list / number of comparisons is proportional to the number of nodes; ***[2 marks]***

Total: [20 marks]

3. (a) (i) *Award up to **[2 marks max]**.*

*Award **[1 mark]*** for the idea of going through the file in order / each record in turn until the (correct) record is found;

*Award **[1 mark]** for any further point from the list below:*
Starting with the first record on the file;
The search item / ID number is searched for / compared;
Continues until the end of file (if record not found);
Or the end file is reached; ***[2 marks]***

(ii) *Award up to **[2 marks max]**.*
Because the file is very large;
The search process could be slow;
With O(n) efficiency;
As records have to be inspected in the order they are stored;
So might have to inspect many records; ***[2 marks]***

(b) (i) *Award up to **[4 marks max]**.*
A calculation is performed / hash key produced;
On the key field / ID number;
Which gives/leads to/identifies the address of the record on the disk;
Access is direct / O(1);
Example of hash algorithm;
Describes use of hash table;
Possible collisions have to be taken care of;
Records would have to be of equal length; ***[4 marks]***

(ii) Must generate all of the record addresses in the file;
Should be fast (to calculate);
Should minimise collisions / accept **no** collisions; ***[3 marks]***

(c) (i) *Award up to **[2 marks max]**.*
If you could hash the surname this would generate different addresses/hashes;
Than those generated by the ID number;
Two hash tables would be needed;
Cannot hash for two different fields; ***[2 marks]***

(ii) *Award up to **[3 marks max]**.*
Two indexes would be created;
One for the ID and one for the surname;
(User) can choose which index to use;
Each index would contain an entry for every search item/full index;
A search of the (appropriate) index would provide the memory location of the required record;
Note:** Do **not** accept "1 index for 2 fields".* ***[3 marks]

continued ...

Question 3 continued

(d) (i) *Award up to **[2 marks max]**.*
If the number of entries changes;
A dynamic structure expands/contracts to the required size;
Which prevents wasted memory / too little space;
Note:** Do **not** accept references to modification of the data structure.* ***[2 marks]

(ii) A binary tree provides a faster search algorithm / searches faster / binary search;
O(log n) as opposed to O(n) for a linked list; ***[2 marks]***

Total: [20 marks]

4. (a) *Award up to **[2 marks max]**.*
If a plane's status is changed;
The FIDS display system could be updated automatically / in real time;
As the flight movements are controlled by ATC; ***[2 marks]***

(b) (i) *Award up to **[1 mark max]**.*
Fibre optic / Wi-Fi / Ethernet cable (**not** just cable); ***[1 mark]***

(ii) *Award up to **[2 marks max]** for suggesting one reason **[1 mark]** and expansion **[1 mark]**, or for suggesting two reasons **[2 mark]**.*
(Fibre optic) would allow fast communication around the network;
Which is important for coordination between the different systems / which is essential for critical systems;
High bandwidth;
So can accommodate many systems at the same time;
Less "noise" than traditional media / high integrity of data;

(Wi-Fi) could allow airport / security personnel;
To access the network;
From portable devices;

(Ethernet) allows fast transfer of data;
Reliable / well-established medium; ***[2 marks]***

(c) Retina scans (any biometrics will do);
Are unique to each individual;
Turns features into mathematical expressions;
Will compare a person's characteristics with a central database; ***[2 marks]***

(d) *Award **[1 mark]** for each feature and **[1 mark]** for a description that clearly shows its benefit, up to **[4 marks max]**.*
***Note:** Only credit features that are present inside of the airport.*

Modern display;
Give up-to-date information on plane movement;

Improved sound systems;
Provide clear audio information;

Self-service check-in;
Speeds up check-in / avoids queuing:

Wi-Fi systems;
Allow passengers to connect to the Internet; ***[4 marks]***

continued ...

Question 4 continued

(e) *Award marks as follows up to **[5 marks max]**.*
*Award **[2 marks]** for a clear description of the advantage;*
*Award **[2 marks]** for a clear description of the disadvantage;*
*Award **[1 mark]** for a conclusion / summary / opinion;*

For example, the baggage system – there will be others (but only credit those systems that operate within an airport).
Computerized systems move bags around the airport very quickly;
Allows passengers to make connections / avoids loss of bags;
However, if system breaks down there will be chaos;
As personnel to help will be limited;
Plus a substantiated overall opinion;

Note:** Award **[3 marks max]** for non-airport system such as PNR.* ***[5 marks]

(f) (i) Security would be able to see illegal access;
As tags will identify location of each passenger; ***[2 marks]***

(ii) Passengers who have not boarded;
Can be quickly located thereby avoiding delays; ***[2 marks]***

(g) (i) *Award up to **[4 marks max]**.*
Would allow them to test different systems;
Without having to actually build them / to find the best;

Saves considerable time (in the development cycle) / money;
As (supposedly) less chance of final system going wrong;

By simulating a system they could detect problems;
And correct them at the design stage / less chance of failure / improve the system; ***[4 marks]***

(ii) *Award up to **[3 marks max]**.*
Requirements change over time;
Better software/hardware becomes available;
There may have been unforeseen problems with the original design / bugs;
Maintenance allows for beneficial changes to be made; ***[3 marks]***

(h) Wi-Fi transmissions can be easily intercepted;
As they travel through the air / by anyone in the same area;
Encryption prevents the understanding of any intercepted messages; ***[3 marks]***

continued ...

Question 4 continued

(i) *Award up to **[4 marks max]**.*
The tree would be initially populated with the passengers' names;
(In such as way) so that an in-order traversal;
Will produce an ordered list;
Each node will also contain a link to the main file;
So that all details can be accessed / where the details are stored; ***[4 marks]***

(j) (i) A linked list;
Of 11 nodes / items;
Where each node contains details of departing flight / "Flight" object;

An array;
Of size 11;
Where each index stores details of departing flight / "Flight" object;

Note: *No credit for queue, but "Follow Through" credit can be given in part (ii).* ***[3 marks]***

(ii) *Award up to **[3 marks max]**.*
Correct flight (found and) deleted;
By changing pointers;
Next flight due to depart added;
To end of **list**;

Find the departed flight in the array;
Shift all later flight down 1 index position;
Next flight due to depart added;
At end of **array** (index 10); ***[3 marks]***

Total: [40 marks]