M10/5/COMSC/HP2/ENG/TZ0/XX/M+

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

MARKSCHEME

May 2010

COMPUTER SCIENCE

Higher Level

Paper 2

12 pages

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.
- 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 up to **[2 marks max]**.*
The constructor is used to create a new object/instance;
Of the class to which it belongs;
Initializes (the data members) of the new object; ***[2 marks]***

(b) (i) Pete; ***[1 mark]***

(ii) 840; ***[1 mark]***

(c) *Award marks as follows up to **[5 marks max]**.*
*Award **[1 mark]** for initializing* `position` *and/or correctly returning* `-1`.
*Award **[1 mark]** for correct loop.*
*Award **[1 mark]** for correct comparison including use of dot notation.*
*Award **[1 mark]** for early exit from loop.*
*Award **[1 mark]** for correct return statement(s).*

Example 1:

```
public int compare(Player latest)
{
  int position = -1;
  for (int x = 0; x < 10; x = x + 1)
  {
    if (latest.score > topTen[x].score)
    {
      position = x;
      break;
    }
  }
  return position;
}
```

Example 2:

```
public int compare(Player latest)
{
  int position = -1;
  int x = 0;
  boolean found = false;
  do
  {
    if (latest.score > topTen[x].score)
    {
      position = x;
      found = true;
    }
    x = x + 1;
  } while ((!found) && (x < 10));
  return position;
}
```

continued ...

Question 1 continued

Example 3:

```
public int compare(Player latest)
{
  int position = 10;
  while (position > 0 && latest.score > topTen[position - 1].score)
    position = position - 1;
  if (position == 10) position = -1;
  return position;
}
```

[5 marks]

(d) *Award marks as follows up to **[8 marks max]**.*
*Award **[1 mark]** for creation of temporary array.*
*Award **[1 mark]** for correct 1st loop.*
*Award **[1 mark]** for assignment of temp in loop 1.*
*Award **[1 mark]** for adding new game data (between loops) in correct position.*
*Award **[2 marks]** for correct 2nd loop (award **[1 mark]** for good but incorrect attempt).*
*Award **[1 mark]** for assignment of temp in loop 2.*
*Award **[1 mark]** for* `topTen = temp`.

Example:

```
public static void updateList(Player latest, int position)
{
  Player[] temp = new Player[10];
  for (int x = 0; x < position; x = x + 1)
  {
    temp[x] = topTen[x];
  }
  temp[position] = latest;
  for (int y = position; y < 9; y = y + 1)
  {
    temp[y + 1] = topTen[y];
  }
  topTen = temp;
}
```

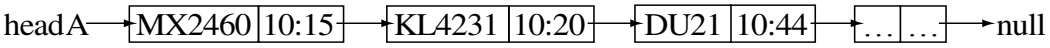
[8 marks]

(e) *Award up to **[3 marks max]**.*
When the insertion position is found;
The remaining values are shuffled down one position (or any equivalent explanation);
Except for the last one, which is omitted;
The new game data is placed in its correct position; ***[3 marks]***

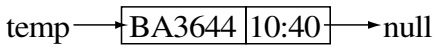
Total: [20 marks]

2. (a) *Award up to **[3 marks max]**.*
Easy to expand the list if necessary;
As a linked list is a dynamic structure;
No wasted memory (for the same reason);
Nodes can be inserted or deleted easily;
Simply by adjusting pointers/references; ***[3 marks]***

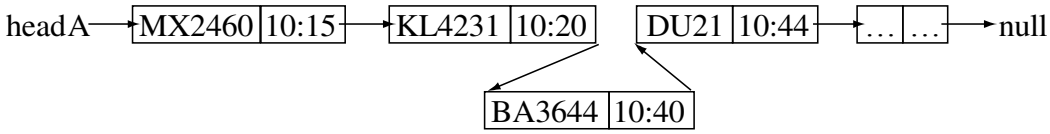
(b) Original linked list.



A new node is created with details of the new flight.



Pointers are adjusted as shown below.



Award marks as follows.
A new node is created with details of the new flight;
The insertion point is found by searching through the list;
Pointer of previous node changed so that it points to new node;
Pointer of new node changed so it points to next node in list;
A reasonably clear diagram;

Marks can be given if any of the above points are clearly shown by diagrams. ***[5 marks]***

(c) *Award marks as follows up to **[4 marks max]**.*
*Award **[1 mark]** for use of temp.*
*Award **[1 mark]** for correct loop.*
*Award **[1 mark]** for displaying flight number.*
*Award **[1 mark]** for moving along loop.*

Example:

```
public static void displayListA()
{
  Node temp = headA;
  while (temp != null)
  {
    output(temp.flightNumber);
    temp = temp.next;
  }
}
```

[4 marks]

continued …

Question 2 continued

(d) *Award marks as follows up to **[8 marks max]**.*
*Award **[1 mark]** for declaration of variables (including* `tempA` *and* `tempB`*).*
*Award **[1 mark]** for check if head = flight number.*
*Award **[1 mark]** for correctly deleting the first node.*

*Award **[1 mark]** for correct 1st loop for List A.*
*Award **[1 mark]** for deletion of node (changing the pointer of "previous" or alternative method).*
*Award **[1 mark]** for movement along the list and comparison.*

*Award **[1 mark]** for correct 2nd loop for List B.*
*Award **[1 mark]** for adding node from List A.*
*Award **[1 mark]** for making it point to null.*

Example:

```
public static void landed(String flightLanded)
{
  boolean found = false;
  Node tempA, tempB, previous;
  tempA = headA;

  if (headA.flightNumber == flightLanded)  // check head of List A
  {
    found = true;
    headA = headA.next;  // or headA = tempA.next;
  }
  while (!found)  // work through rest of List A
  {
    previous = tempA;
    tempA = tempA.next;
    if (tempA.flightNumber == flightLanded)
    {
      previous.next = tempA.next;  // eliminate node from List A
      found = true;
    }
  }
  tempB = headB;
  while (tempB.next != null)  // add to end of List B
  {
    tempB = tempB.next;
  }
  tempB.next = tempA;
  (tempB.next).next = null;  // or tempA.next = null
}
```

[8 marks]

Total: [20 marks]

3. (a) A key is decided upon;
Records sorted in order of key field;
Partial index decided upon (*e.g.* sub-sections);
Record number / memory location of the first member of each sub-index found and added to create part of index; ***[4 marks]***

(b) *Award **[1 mark]** for a suitable structure, and **[2 marks max]** for the reasons.*

EITHER
A *linked list* could be used;
This would allow the index to become longer if necessary (do not allow "because it is dynamic" on its own);
Without wasting memory space;
Would contain nodes which (hold data) that point to the record address;

OR
A *binary (search) tree* could be used;
Which would allow fast searching for the index;
As each comparison eliminates half of the remaining tree / has search efficiency of log n / or something similar to this;
This would allow the index to become longer if necessary (do not allow "because it is dynamic" on its own);
Without wasting memory space;
Would contain nodes which (hold data) that point to the record address; ***[3 marks]***

(c) *Tree (linked list)*
The tree (linked list) is traversed (the method can be given);
Using the key field (can name it);
To find (the node with) the required group;
This (node) contains (a reference to) the file address;
Of the correct group of records;
Which are then searched sequentially; ***[6 marks]***

(d) (i) The index will take up more space;
The index has to be modified whenever a new record is added;
Allow "can be slower" if a suitable explanation is given. ***[2 marks]***

(ii) Access is faster;
As the index will allow direct access to the required record/without need to search through others/the group;
Reduces overheads;
As no need to have the records sorted; ***[2 marks]***

(e) *Award up to **[3 marks max]**.*
Hash table / Direct access / Random Access File could be used;
In which a calculation;
Based on the key field;
Provides the memory location of / direct access to the record; ***[3 marks]***

Total: [20 marks]

4. (a) *Award up to **[2 marks max]**.*
This allows for the new system to be evaluated;
Without any consequences if the system fails;
As the old system is still functioning; ***[2 marks]***

(b) In order to make the testing as realistic as possible;
To ensure that there would be no problems when the terminal was opened; ***[2 marks]***

(c) (i) *Accept any of the systems that would need real-time.*
e.g. the control tower system; ***[1 mark]***

(ii) *e.g.* for the control tower
Input data must be processed immediately;
In order to have an up-to-date situation;
To prevent any potentially dangerous situations; ***[3 marks]***

(d) (i) Several people are likely to be in the control tower;
Whose voices could be mistakenly received by the voice recognition system;
Leading to wrong information being input / delays in processing genuine commands / potentially dangerous situations;

Allow answers that focus on the inaccuracy of voice recognition. ***[3 marks]***

(ii) *Award up to **[3 marks max]**.*
A touch screen might be suitable;
Commands could be given by touching various menus/boxes;
Different airplanes could be represented graphically on the screen;
Whose position could be changed by touch; ***[3 marks]***

(iii) *Award **[1 mark]** for a realistic suggestion and **[1 mark]** for outlining its use.*
Example:
Airplane graphic changes size/colour/makes noise;
To show relative proximity to the airport; ***[2 marks]***

(iv) *Award up to **[2 marks max]**.*
The previous system (paper strip) could be kept functioning;
With all data entering both systems;
So that it could take over if needed;

The system could be mirrored;
Using the old control tower / separate server;
With all data entering both systems;

Divert all airplanes;
To another airport;

Do not award any marks for a strategy that would either lose data or would involve a significant interruption in processing. ***[2 marks]***

continued…

Question 4 continued

(e) *There are many possible answers, including ones based on personal experiences. For each side of the argument, award **[1 mark]** for an indication of the side, **[1 mark]** for stating why and **[1 mark]** for an explanation, up to **[3 marks max] for each side**.*

Examples:
Governments are ultimately responsible for the safety of its citizens;
Therefore have the right to take any measures necessary;
To identify threats;

This information should not be made widely available;
As it is private/confidential;
And medical details *etc.* could be used against the interests of the passenger;

Governments' proven inability to keep data secure could also lead to a correct answer. ***[6 marks]***

(f) *Award up to **[2 marks max]**.*
Each section can be separately protected;
So if one part of the system is compromised in some way;
The other sections still have their own protection; ***[2 marks]***

(g) (i) Because there are only 2 possible values / already "digital";
Beam broken or not broken; ***[2 marks]***

(ii) *Award up to **[4 marks max]**. To gain **[4 marks]**, both interrupts and polling should be considered.*
Interrupts would happen immediately;
Display screens would be permanently up to date;
This might be safer; (don't just accept "this will be 'better'")
Polling would poll many sensors unnecessarily;
Might lead to a delay in reporting changes in position;
Because polling inspects all inputs;
A complete / up-to-date picture can be maintained; ***[4 marks]***

(h) *Award up to **[3 marks max]**.*
They can inherit several common data members;
And common methods;
Or they can override methods of the abstract/`Flight` class;
Whilst defining their own particular members/methods; ***[3 marks]***

(i) (i) The check-in terminals would be the clients;
The computer holding the flight database would be the server; ***[2 marks]***

(ii) The kiosks/desks send request/receive information to/from the server/queries;
The server processes requests and communicates back;
By accessing its database; ***[3 marks]***

Total: [40 marks]