# MARKSCHEME

## May 2007

## COMPUTER SCIENCE

## Higher Level

## Paper 2

15 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 part of a question.

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

- Each marking 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 mark scheme 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. Effective communication is more important than grammatical niceties.
- 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) 7; ***[1 mark]***

(b) *Award **[1 mark]** for any of the following up to **[2 marks max]**.*

no check for end of array;
in the while loop;
possible (array index) out of bounds error;
comparing double numbers/-1.0 may not work;
because they are/it is not stored precisely/accurately/correctly; ***[2 marks max]***

(c) A possible solution is:

```
public int findSmallest(double[] n)
{
     double smallest = n[0];
     int smallPos = 0;
     for(int x = 1; n[x] > 0.0; x++)
     {
          if (n[x] < smallest)
          {
            smallest = n[x];
            smallPos = x;
          }
     }
     return smallPos;
}
```

*Award marks as follows up to **[6 marks max]**.*
algorithm declared to return an int value;
array as a parameter;
correct initialisation of smallest and smallPos (or equivalent);
correct data types for smallest and smallPos (or equivalent);
correct loop (start, end AND increment);
test for smaller than current smallest;
update of smallest and smallPos;
return of smallPos; ***[6 marks max]***

(if smallPos has been omitted altogether, ie the algorithm attempts to return the smallest value, 1 mark may be awarded for the correct initialisation, correct type AND return of smallest or its equivalent). This solution should not be awarded more than 4 marks maximum, however.

Note that small could equally well be set to a negative value and all array elements (including zero) tested. No penalty for this slightly less efficient solution.

(d) *Award marks as follows up to* ***[2 marks max]***.

| element: | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|---|
| contents | -1.0 | | | | | | - | - |

the -1.0 value at element 0 (contents of other elements are not relevant);
a pointer/arrow to element zero, clearly labelled TOP, (or stacktop or something equally clear); ***[2 marks max]***

(e) A possible solution is:

```
public double pop(double[] n)
{
     double topValue = n[0];
     for(int x = 1; n[x-1] > 0.0; x++)
     {
          n[x-1] = n[x];
     }
          return topValue;
}
```

*Award marks as follows up to* ***[4 marks max]***.
correct start point for loop;
correct end condition;
correct shuffle;
test for empty stack;
correct return value; ***[4 marks max]***

Note: Other solutions are possible but should be able to be marked as above, e.g.:

```
public double pop2(double[] n)
{
     if (n[0] == -1.0)                // test empty stack
     {
          return -1.0;                // or n[0] correct return part 1
     }
     else
     {
          int x = 1;                  // correct start point
          double temp = n[0];         // correct return part 2
          while (n[x] > 0.0)          // correct end point
          {
               n[x-1] = n[x];         // shuffle 1
               x++;                   // shuffle 2
          }
     }
     n[x-1] = -1.0;                   // correct shuffle 3
     return temp;                     // correct return part 3
}
```

*Award* ***[1 mark]*** *for part 3's if 1 and 2 also correct.*
Be careful to award marks for correct but terse syntax such as:

$n[x-1] = n[x++];$

which combines the shuffle and increment.

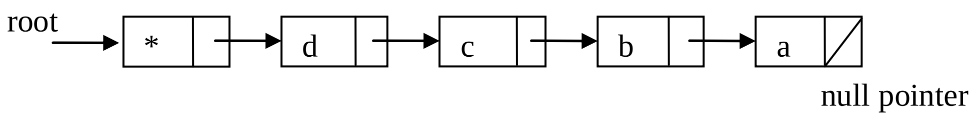
(f) $O(n)$; ***[1 mark]***

(g) *Award* ***[1 mark]*** *for any of the following up to* ***[4 marks max]***.
the stack pointer/top could be moved;
so that it points to the last data item;
eliminating the need for a -1.0 value/end of data marker;
1 more item could be stored in the array;
the push/pop operation(s) would then be O(1)/more efficient; ***[4 marks max]***

**2.** (a) *Award marks as follows up to **[3 marks max]**.*
*Award **[1 mark max]** if no reference to the example code is given.*

a constructor is a (special) method;
used to construct/instantiate/create an object/instance;
there are two Constructors in the example;
they have the same name as the Class (ListNode);
a Constructor does not have a return type; ***[3 marks max]***

(b) The sketch should look like this:



*Award as follows up to up to **[5 marks max]**.*
*Award **[2 marks max]** if the sketch is not clearly labelled.*
*Award **[4 marks max]** if the items are in reverse order.*

node structure;
with at least one char value inside;
with a clear pointer field;
with nodes connected by pointers;
root pointer at start of list;
null pointer at end of list;
correct sequence of node characters;
inclusion of '*' as a node at the start; ***[5 marks max]***

(c) *Award **[1 mark]** for*
output of `*` /output of the sequence `a b c d` only.
*Award **[2 marks]** for*
output of the correct sequence: `a b c d` *. ***[2 marks]***

(d) The main alternatives are:

```
public String theWord(ListNode r, String word)
{
    if (r == null)            // check for end of list
    {
        return word;            // return parameter
    }
    // check for star char, don't add to the word
    else
    {
        if (r.getCh() != '*')
        {
             return theWord(r.getNext(), r.getCh() + word);
        }
        return theWord(r.getNext(), word);
    }
}
```

*Award marks as follows up to* ***[6 marks max]****.*

correct test for end of list;
returning word or equivalent on end;
check for star character;
moving to next node;
adding character to sequence;
correct recursive return call;
avoiding addition of star char to string; ***[6 marks max]***

**OR**

```
public String theWord2(ListNode r)
{
     String word = "";
     while (r != null)
     {
          if (r.getCh() != '*')
          {
               word = r.getCh()+ word;
          }
          r = r.getNext();
     }
     return word;
}
```

*Award marks as follows up to **[6 marks max]**.*

declaration/initialization of word or equivalent;
correct test for end of list;
check for star character;
adding character to sequence;
avoiding addition of star char to string;
moving to next node;
returning word or equivalent on end; ***[6 marks max]***

Since the star character is always the first data item in the list (given the way the question is constructed), the if statement could be replaced by an initial statement before the loop:

```
r = r.getNext();
```

*Award **[2 marks]** for this statement if correctly positioned.*

(e) *Award marks as follows up to **[4 marks max]**.*

another pointer field/identifier would be required;
which points to the previous node;
further set and get methods would be required;
a tail pointer/identifier would be required; ***[4 marks max]***

*Accept a clearly labelled diagram illustrating any of the above points.*
*Accept clear modifications to the class code which correspond to the marking points above.*

**3**. (a) partially-indexed (file organization); ***[1 mark]***

(b) *Award **[1 mark]** for each of the following up to **[4 marks max]**.*

the index is searched;
until the entry danh is found;
the record number of chan / previous entry is retrieved;
and used to start searching the main file;
which is searched;
sequentially;
until chun is found;
or the next entry/end of file is encountered; ***[4 marks max]***

(c) (i) *Award **[1 mark]** for each of the following up to **[3 marks max]**.*

this record is added at the start of the file;
added in order;
this requires moving following records;
to create necessary space;
this is an inefficient operation; ***[3 marks max]***

(ii) *Award **[1 mark]** for each of the following up to **[3 marks max]**.*

the index values (after the insertion) have to be changed;
the index entry anh is changed;
to adz; ***[3 marks max]***

(d) *Award **[1 mark]** for any of the following up to **[2 marks max]**.*

each index entry/letter/group of records could be allocated a block;
which includes some empty space;
where new records could be added;
(partial) index still points to first in block;

the data file could be unsorted;
and records added at the end;
this would require a full index; ***[2 marks max]***

(e) *Award **[1 mark]** for each of the following up to **[5 marks max]**.*
a binary search (could be used);
in which the mid-point of the array/file is identified;
and the record there compared;
to the wanted record;
if the wanted record is higher/lower;
then the lower/upper half of the file/array;
need not be searched;
this has an efficiency of O(log n); ***[5 marks max]***

(f) *Award **[1 mark]** for each of the following up to **[2 marks max]**.*
direct access is required (to the main file);
tape does not support direct access;
but only sequential access; ***[2 marks max]***

**4.** (a) text is scanned into a (image) file;
OCR software converts into a readable format;
speech synthesis/screen magnification enables student to hear/read notes; ***[3 marks]***

(b) (i) Video Magnifiers;
cameras relay text to screen (where it can be magnified); ***[2 marks]***

(ii) Electronic Braille displays;
enables the students to read the contents of the computer screen; ***[2 marks]***

(c) a working model of the interface is created;
a user is asked to trial it;
changes are suggested (by the user);
this cycle is continued as required; ***[3 marks max]***

(d) *Award* ***[1 mark]*** *for an entry method and* ***[1 mark]*** *for explaining why it is reliable.*

For example.

(i) Entry methods:
a Braille keyboard;
a set of buttons with audible response;
etc.

Reliable because
blind persons have acute sense of touch;
Braille system is well established;
input text could be read back to person;
etc. ***[2 marks]***

(ii) Entry methods:
a head-mounted (pointing) device;
a touch screen – clearly identify part of body to be used in place of hands, eg toes;
mouth tube (either physically operated or by suck/blow);
speech input;
etc.

Reliable because
person can see text entered via a monitor;
input text could be read back to person;
etc. ***[2 marks]***

(e) For each method ***[1 mark]*** for a suitable method and ***[1 mark]*** for explaining why it is suitable up to ***[4 marks max]***; for the comparison ***[1 mark]*** for each valid comparison made up to ***[2 marks max]***.

Methods:
observation;
can see what the person is doing;

questionnaire;
person can still write;
can collect a lot of information; (not from many people though – not relevant to question)
could supplement with further written questions to elaborate;

interview (not by conversation though!);
conducted via a word processor;
able to gain richer information;
by asking questions in immediate response to answers;

Comparisons:
interview gives more data than questionnaire/observation alone;
questionnaire can be filled in without analyst/student being present;
observation is independent of the user (less risk of bias); ***[6 marks max]***
etc.

(f) *Award **[1 mark]** for any of the following up to **[2 marks max]**.*

the beep needs to be replaced/augmented;
(probably) with a light signal/bulb/LED;
or alternate method – such as a "thumper"; ***[2 marks max]***

(g) For each of the following, ***[1 mark]*** should be awarded for correctly identifying a person for whom the system is suited and ***[1 mark]*** for a justification/elaboration.

(i) could be especially suited to:
hearing impaired, physically disabled;
they are able to see/read/understand how the system will work;
no interaction is required (by physically disabled); ***[2 marks]***

(ii) hearing impaired; wheelchair bound (but able to use hands);
they are still able to interact with the system;
and can view the responses on screen; ***[2 marks]***

(iii) physically disabled with limited hand movement/coordination/motor control, persons with limited or no vision;
they are able to use the switches;
braille markings could be added;
the outputs would need to be audible (for visually impaired); ***[2 marks]***

(h) *Award* ***[1 mark]*** *for a way,* ***[1 mark]*** *for an elaboration* ***[2 marks]*** *per example × 3 up to* ***[6 marks max]****.*

For example:

A wheelchair bound person;
could give commands via voice;

A physically-disabled/blind person;
could operate light switches/domestic appliances/their environment;

A physically-disabled/blind person;
could use it to enter text (into a word processor)/write a letter; ***[6 marks max]***

(i) *Award* ***[1 mark]*** *for an environment;* ***[1 mark]*** *for an elaboration* ***[2 marks]*** *per example × 3 up to* ***[6 marks max]****.*

For example:

A noisy/street environment;
to see if the person's voice can be heard/distinguished;

A situation where the person's voice may change;
eg if they have a cold;

A potentially unsafe environment (such as a busy street, edge of a lake);
To see that the chair reacts quickly (or using an able bodied person or crash dummy);

accept answers referring to specific situations, such as a shopping mall, airport, camping trip if it is related to safety issues. ***[6 marks max]***