MARKSCHEME

May 2004

COMPUTER SCIENCE

Standard Level

Paper 2

10 pages

If you do not have a copy of the current Computer Science Guide, please request one from IBCA.

General Marking Instructions

After marking a sufficient number of scripts to become familiar with the markscheme and candidates' responses to all or the majority of questions, Assistant Examiners (AEs) will be contacted by their Team Leader (TL) by telephone. The purpose of this contact is to discuss the standard of marking, the interpretation of the markscheme and any difficulties with particular questions. It may be necessary to review your initial marking after contacting your TL. **DO NOT BEGIN THE FINAL MARKING OF YOUR SCRIPTS IN RED INK UNTIL YOU RECEIVE NOTIFICATION THAT THE MARKSCHEME IS FINALIZED.** *You will be informed by e-mail, fax or post of modifications to the markscheme and should receive these about one week after the date of the examination. If you have not received them within 10 days you should contact your Team Leader by telephone. Make an allowance for any difference in time zone before calling.* **AEs WHO DO NOT COMPLY WITH THESE INSTRUCTIONS MAY NOT BE INVITED TO MARK IN FUTURE SESSIONS.**

You should contact the TL whose name appears on your "Allocation of Schools listing" sheet.

Note:
Please use a personal courier service when sending sample materials to TLs unless postal services can be guaranteed. Record the costs on your examiner claim form.

General Marking Instructions

1. Follow the markscheme provided, do **not** use decimals or fractions and mark only in **RED**.

2. Where a mark is awarded, a tick (✓) should be placed in the text at the **precise point** where it becomes clear that the candidate deserves the mark.

3. Sometimes, careful consideration is required to decide whether or not to award a mark. Indeed, another examiner may have arrived at the opposite decision. In these cases write a brief annotation in the **left hand margin** to explain your decision. You are encouraged to write comments where it helps clarity, especially for moderation and remarking.

4. Unexplained symbols or personal codes/notations on their own are unacceptable.

5. Record subtotals (where applicable) in the right-hand margin against the part of the answer to which they refer. Show a mark for each part question (a), (b), *etc.* Do **not** circle sub-totals. Circle the total mark for the question in the right-hand margin opposite the last line of the answer.

6. Where an answer to a part question is worth no marks, put a zero in the right-hand margin.

7. Record the mark awarded for each of the three questions answered in the Examiner Column on the cover Sheet.
Add up the marks awarded and enter this in the box marked TOTAL in the Examiner Column on the cover sheet.

8. After entering the marks on the cover sheet check your addition of all marks to ensure that you have not made an arithmetical error. Check also that you have transferred the marks correctly to the cover sheet. **We have script checking and a note of all clerical errors may be given in feedback to all examiners.**

9. Every page and every question must have an indication that you have marked it. Do this by **writing your initials** on each page where you have made no other mark.

10. A candidate can be penalized if he/she clearly contradicts him/herself within an answer. Once again make a comment to this effect in the left hand margin.

Subject Details: Computer Science SL Paper 2 Markscheme

Mark Allocation

Candidates are required to answer ALL questions. (***[30 marks]*** for question 1, ***[25 marks]*** for question 2 and ***[15 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 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.
- The order of points does not have to be as written (unless stated otherwise).
- 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) (i) accept only 165; ***[1 mark]***
(ii) accept only 258; ***[1 mark]***

(b) *Candidates are not required to re-write the entire algorithm as long as they have made it clear where the additions belong. Award **[1 mark]** for a correct* `if` *condition and **[1 mark]** for returning –1 correctly e.g.*

```
function MINUTE (val real TIME) result integer
/* TIME represents a time in the form hours.minutes */
/* the total minutes since midnight are returned */
  declare HOURS, MINUTES, TOTAL integer
  HOURS <-- truncate(TIME)
  MINUTES <-- round((TIME - HOURS)*100)
  if  MINUTES < = 59 then
    TOTAL <-- HOURS + MINUTES
    return TOTAL
  else
    return -1
  endif
endfunction MINUTE
```

[2 marks]

(c) *Award **[1 mark]** for an hour out of range (e.g. 24 up or negative). Accept a minute that is more than two decimal places long, e.g. 10.59192.* ***[1 mark]***

(d)

```
function TIMETAKEN (val real TIME1,TIME2,
                    val real TIME2) result integar
  declare MINUTES1, MINUTES2, DIFFERENCE integer
  MINUTES1 <-- MINUTE (TIME1)
  MINUTES2 <-- MINUTE (TIME2)
if MINUTES1 = -1 or MINUTES2 = -1 then
  return -1
else
  DIFFERENCE = MINUTES1 - MINUTES2
  if DIFFERENCE < 0 then
    DIFFERENCE <-- DIFFERENCE * -1
  endif
  return DIFFERENCE
endfunction TIMETAKEN
```

*Award **[1 mark]** for each of the following.*
a valid call to `MINUTE`;
correct test of return value of `MINUTE` in both cases;
correctly returning –1 on error, syntax unimportant but it must be clear that this is a function returning an integer;
checking if `DIFFERENCE` is negative;
multiplying by –1 if so; ***[4 marks max]***

*Award full marks for any valid solution, e.g. using abs function would be worth **[2 marks].***

(e)
```
procedure RETURN (val BIKEID, ref ID string array, ref OUT real array string)
  declare POS integer
  declare CURRENT real
  POS = 1
  while ID[POS] # "ZZZ" and BIKEID # ID[POS] do
       POS <-- POS + 1
  endwhile
  if ID[POS] = "ZZZ" then
     output "Error, no such BIKEID"
  else
     output "Input current time"
     input CURRENT
     output "Time taken ", TIMETAKEN (CURRENT,OUT[POS])
     while ID[POS] # "ZZZ" do
           ID[POS] <-- ID[POS + 1]
           OUT[POS] <-- OUT[POS + 1]
      endwhile
  endif
endprocedure RETURN
```

*Award **[1 mark]** for each statement up to a maximum of **[11 marks]***

`BIKEID` declared as a parameter;
correct declaration of **all** local variables used;

loop through/linear search section *up to **[3 marks]***;
  correct test of `ID[POS]`;
  correct test of `BIKEID` ;
  correct initialization AND increment of `POS` or equivalent;

correct "not found" test;
any error message in correct place;
entry of current;

output statement *up to **[2 marks max]***;
  *Award **[1 mark]** for attempt at output of* `TIMETAKEN`*;*
  *Award **[1 mark]** for any attempt to use* `TIMETAKEN`*;*
  *Award **[1 mark]** for use of* `TIMETAKEN` *consistent with answer to part (d);*

*Award up to **[3 marks max]** for final section – loop to delete entries in both arrays.*
  continuing to loop from `POS`;
  any reasonable attempt at shuffle in `ID` AND `OUT` arrays *(forgive "off by one" errors);*
  *Award **[2 marks]** for a completely correct delete by shuffle in **both** arrays;*

[11 marks max]

(f)
```
procedure ADD (val BIKEID string, val TIME real)
  declare POS integer
  POS <- 1
  while ID[POS] # "ZZZ" do
       POS <- POS + 1
  endwhile
  ID[POS] <- BIKEID
  OUT[POS] <- TIME
  ID[POS + 1] = "ZZZ"
endprocedure ADD
```

*Award **[1 mark]** for each of the following unless otherwise stated.*
correct parameters;
any local variables used correctly declared;
correct loop through to find `ZZZ` entry ***[2 marks]***;
assigning both values for `ID` and `OUT`;
assigning "`ZZZ`" entry to new last position; ***[6 marks]***

(g) the arrays could be organized such that the `ID`'s were in order;
this would make a binary search possible;
a binary search would be faster;
especially if there is a large number of entries;
adding new items would require shuffling the array;
deleting items would take (more or less) the same amount of time; ***[4 marks max]***

2. (a) *uniformity*:
FMS allows the customization of manufactured goods;
since the machines can be programmed;
to produce small numbers / individual goods;
still allowing cost savings of large scale production;

size:
conventional systems require large capital outlay;
whereas FMS can be adapted to different products;
by changing the programme; ***[6 marks]***

(b) different operations might be represented by icons;
associated parameters available via lists;
programs can be assembled by drag and drop
forming a visual sequence;
complex operations can be presented in separate windows; ***[3 marks]***

(c) communications in the FMS need to be reliable;
since wrong instructions;
or corrupt data;
could lead to faulty goods;
or accidents;
and the machines are not under direct operator control; ***[4 marks]***

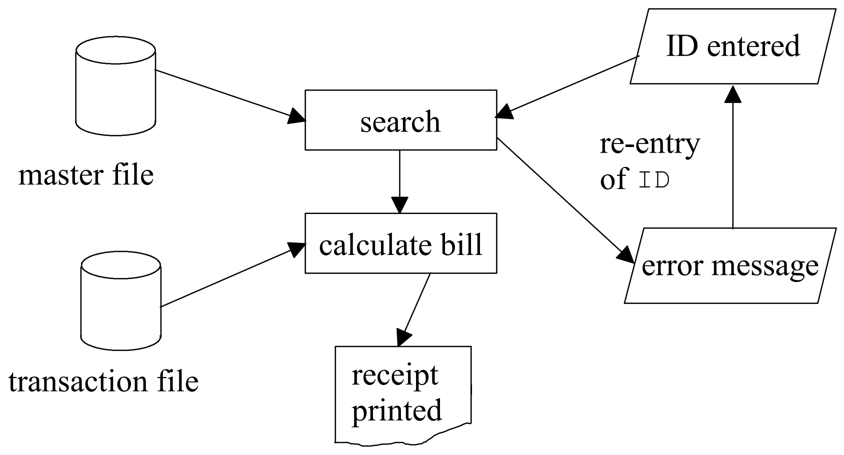
(d) *Award* ***[1 mark]*** *for each reason and* ***[1 mark]*** *for each explanation.*

differing response time requirements: single computer is impractical to handle all;
differing control needs at each level: each can be tailored to suit a particular need;
if computer fails: other operations can be maintained; ***[6 marks max]***

(e) both wire-frame and surface models are ambiguous;
because not all the information about the object is present;
wire-frames only have data about the points and edges of an object;
and surface models do not have data about which parts of the object are solid and which parts are voids;
only solid models have all the data to completely represent an object;
and can therefore model objects in such a way as to prevent intersection; ***[6 marks]***

Allow any other points that are correct and relevant to the question.

3. (a)



*[**2 marks**] for error message box and clear indication of data re-entry (explanatory text not required). [**1 mark**] for each other correct box and its associated arrow pointing in the correct direction.* ***[8 marks]***

(b) *[**1 mark**] for correct identification, [**1 mark**] for an elaboration up to [**2 × 2 marks**].*

a barcode could be used on the card ***[1 mark]*** and a barcode reader used ***[1 mark]***;
a number could be typed on the card ***[1 mark]*** and an OCR scanner used ***[1 mark]***;
ID's could be printed in magnetic ink ***[1 mark]*** and an MICR reader used ***[1 mark]***;
(pretty expensive for a bicycle rental business, I appreciate, but let it go)
the card could have holes punched in it ***[1 mark]*** which could be read by a punched card reader ***[1 mark]***;
a magnetic strip could be used to encode the ID ***[1 mark]*** and this could be read by using a (badge) reader ***[1 mark]***. ***[4 marks max]***

(c) (i) *Award [**1 mark**] for either of the following.*
it is text based;
commands are typed in; ***[1 mark max]***

(ii) *Award [**1 mark**] each for two of the following.*
less processing required;
could use a lower spec machine;
limited range of data to input;
limited range of commands needed; ***[2 marks]***