IB DIPLOMA PROGRAMME
PROGRAMME DU DIPLÔME DU BI
PROGRAMA DEL DIPLOMA DEL BI

N05/5/COMSC/SP2/ENG/TZ0/XX/M

MARKSCHEME

November 2005

COMPUTER SCIENCE

Standard Level

Paper 2

11 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) *Award **[1 mark]** for each of the following*:

the data types, character and Boolean are not the same;
arrays have to hold the same type of data; ***[2 marks]***

(b) For example:

```
function PLAY (ref n integer)
result boolean
   declare try char
   declare answer boolean
   output SUMS[n]
   input try
   if (try = "y" and CORRECT[n]) or (try = "n" and not CORRECT[n])
      then answer = true else answer = false
   endif
return answer
endfunction
```

There are many possibilities award marks as follows:

Function declared as Boolean;
Equation output;
Correct test for correct;
Correct test for incorrect;
Correct Boolean value returned; ***[5 marks max]***

Note that `SUMS` and `CORRECT` can be passed by reference also but this complicates the next part of the question. However, do not penalise.

(c) Candidates will have to use a method of noting which equations have been used. This could be by introducing another Boolean array which flags those used or by simply keeping a list of those used. Award marks for either correctly implemented.

For example:

```
procedure GAME1( ref SUMS string array [1..50],
                 ref CORRECT boolean array [1..50])
   declare  USED array [1..50] of boolean
   declare COUNT, N, total integer
   for COUNT < -- 1 upto 50 do
         USED[COUNT] <-- 0
   endfor
   TOTAL < -- 0
   COUNT <-- 0
   while COUNT<10 do
         repeat
               N < -- round (random * 50 + 0.5)
         until USED[N] = 0
         COUNT <-- COUNT + 1
         if PLAY (N) then TOTAL < -- TOTAL +1
   endwhile
   output TOTAL
endprocedure
```

*Award marks as follows up to **[9 marks max]**.*

Use of extra array or list for used equations set up;
All local variables declared and initialised;
Loop for 10 equations;
Random integers from 1 to 50 generated ***[2 marks]*** (do not penalise if 0 included).
Equation checked for already used;
Function `PLAY` called;
Scored accordingly;
Correct total output; ***[9 marks]***

(d) *There are many possible methods but the following should be taken into account.*

Large number on screen;
Colour/flashing *etc*. on screen;
A click on button for false or true/or touch sensitive screen with part to push; ***[3 marks]***

(e) *Award **[1 mark]** for each correct output and **[1 mark]** for input of 5.*

n	COUNT	output	input
3	1	9	
	2	9-	
	3	9-*	
	4	9-*=	
	5	9-*=5	5
		x	

[5 marks]

(f) read characters from the string until terminator or end of string
read and count digits until symbol
    if one digit the store as number1
    if two digits then number1 = digit1*10 + digit2
store symbol as its arithmetic equivalent
read and count digits until "="
    if one digit the store as number2
    if two digits then number2 = digit1*10 + digit2
ignore "="
read and count digits until end of string
    if one digit the store as number3
    if two digits then number3 = digit1*10 + digit2
if number1 symbol number2 = number3 then correct else incorrect.

Correct parsing of numbers ***[2 marks]***
*Award only **[1 mark]** if one digit numbers only are considered.*
correct storing of three separate numbers ***[1 mark]***
correct read until end of string ***[1 mark]***
test correct ***[2 marks]*** ***[6 marks max]***

2. (a) *Award **[1 mark]** for any of the following up to **[3 marks max]**.*

Spamming is the sending of un-requested mail to many addresses at the same time;
Since this is opened or read it wastes time;
takes up mailbox space;
and is generally unwanted; annoyance factor
risk that virus could be attached; ***[3 marks]***

(b) (i) *Accept any reasonable answer.*

For example:

Normalisation eliminates redundancy and hence saves time and memory.

Avoids changing one and a copy not being changed — data integrity. ***[2 marks]***

(ii) one central database means that there is only one copy
hence it can be checked as correct
everyone has access to the same data and knows that all others have the same

however if it is changed wrongly everyone has the wrong changes
a local copy always available even if transmission problems
if local copy different then could be working off wrong data ***[4 marks]***

(c) (i) 1. Problem
2. Design / Analysis
3. Implement
4. Evaluate
5. Goto 1 with new problem ***[3 marks]***

*Award **[2 marks]** for the four stages and **[1 mark]** for the cycling back to the beginning.*

(ii) prototype has the same cycle back to beginning ***[2 marks]***
but also includes a second inner cycle at the design stage ***[1 mark]***
where user tests the prototype ***[1 mark]***
which generates feedback ***[1 mark]***
and loops until satisfied ***[1 mark]*** ***[5 marks max]***

(d) A prototype is designed to elicit user information;
It is not designed as an end product;
The final design should not be developed from the prototype because its design may not be optimal for a final design;
To start from scratch enables final design criteria to be incorporated; ***[4 marks]***

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

Businesses who hold information about people need to ensure that it is not used for the wrong purpose;
Or hacked by someone else;
When information is held electronically it is easier for it to be used without anyone knowing that it has been tampered with;
Who does the information belong to?;
What about the ease of plagiarism?;
etc. ***[4 marks max]***

3. (a) (i) CPU: used to process large blocks of data quickly. Data fetched from memory and manipulated by CPU before being output to screen or sent back to memory. A fast processor needed, especially if sound is not to be distorted. ***[2 marks]***

(ii) RAM holds program and data being executed at the moment larger RAM means faster running. ***[2 marks]***

(iii) Cache between RAM and the processor. Holds recently used data and instructions that are likely to be needed soon. Speeds up the running if it is large. ***[2 marks]***

(iv) Hard disk: used to store programs and data when not in use. Graphics programs and files are large and disk needs to be large. ***[2 marks]***

(b) *Award **[1 mark]** for an appropriate feature and **[2 marks]** for explaining how this is an enhancement*.

For example:

web browser: may not be latest version and since HTML is continually developing (plugs-ins).
video player: new plug-in possibly needed to display new type of feature. ***[3 marks]***
Do not accept download speed as this is not software.

(c) Data compression is the process of reducing the size of a file by extracting redundant information ***[1 mark]*** in such a way that it can later be reconstructed ***[1 mark]***. In this application needed to reduce storage space ***[1 mark]*** but also to minimise transmission time when uploading ***[1 mark]***. ***[4 marks]***