ASSESSMENT and QUALIFICATIONS ALLIANCE

Mark scheme
January 2004

GCE

Computing

Unit CPT4

Computing: Unit CPT4

The following notation is used in the mark scheme

- **; - means a single mark;**
- **/ - means alternative response;**
- **A – means acceptable creditworthy answer;**
- **R – means reject answer as not creditworthy;**
- **I – means ignore.**

1. (a) B76; **R** lower case B **1**

(b) $183^3/_8$;; *one mark for correct integer part,*
183.375;; *one mark for correct fractional part* **2**

(c) (i) -36.5;;;
partial marks for workings if result incorrect:
1 mark for x2^6; accept showing that binary point moves 6 places right;
1 mark for negative number; **3**

(ii) a significant bit is stored after the binary point;
bit after point different to bit before point;
negative number starts with 10… positive number starts with 01….;
to max **1**

(iii) to maximise accuracy / number stored with maximum precision;
A more accurate;
a given number can only be expressed in one way in a given number of bits; **1**

Total **8**

2. (a) male(peter).; *Penalise once if capitals used instead of lower case* **1**

(b) father(X,Y) IF male(X) AND parent (X,Y)
// father(X,Y) IF parent(X,Y) AND NOT female(X); **1**

(c) brother (X,Y) IF;
parent(Z,X) AND parent(Z,Y); *accept father/mother instead of parent*
AND male(X) AND male(Y);

OR

brother(X,Y) IF;
mother(A,X) AND mother(A,Y) AND/OR father(B,X) AND father (B,Y);
AND male(X) AND male(Y); **3**

Total **5**

3.	(a)	(i)	56,576 / 13 x 16^3 + 13 x 16^2 (*as final answer*);	**1**
		(ii)	16;	**1**
		(iii)	16;	**1**
		(iv)	$512_{(10)}$; 2^9; **A** 200_{16}; **A** &200;	**1**
	(b)	(i)	a signal/command/message; **R** request from a device/source seeking the attention of the processor; *or* from a device/source to the processor;	**2**
		(ii)	control bus; interrupt request line;	**1**
		(iii)	when buffer is full; when transfer is complete; **R** when buffer empty to max	**1**
			finish the current fetch-execute cycle; identify interrupt // type of interrupt is found; disable lower priority interrupts; save contents of registers // save volatile environment // save current status; jump to interrupt service routine // interrupt handler loaded // interrupt serviced; estore contents of registers // restore volatile environment; enable interrupts; **R** goes back to where it left off to max	**3**
			Total	**11**

4. (a) need to access/address machine registers / exact memory addresses / hardware directly;
fast speed of operation required;
code needs to take up little memory;
A minimise the size of the program/code;
no compiler/interpreter exists yet for machine // no other translator exists;
R manipulate bits
to max **2**

(b) *the assembly language instruction MOV or MOVE may be used in place of LOAD, LD, STORE, ST.*
Immediate addressing must be used for constants but could be indicated by annotation e.g. load register with 0. Indicate any omissions
A pseudocode statement gets no mark. Order of statements is important.
Ensure that loop continues while x <> 999.
If subtract is used in place of CMP check value of x is not destroyed.
If symbolic addressing not used, and statements are given addresses used in JMP instructions give 2 marks. Accept relative addressing for jumps.
No marks for jumps not altering flow of control.

```
        LOAD register,#0 / initialisation      1 mark
label1:                                        / label 1 mark
        CMP register, # 999                    / compare 1 mark
        BEQ label2                             / correct branch to label 1 mark
        INC register or ADD register,#1        / increment 1 mark
        JMP label1                             / unconditional branch 1 mark
label2:                                        / correct label 1 mark
        STORE register, memorylocation         / 1 mark
```

to max **7**

Total **9**

5. (a) it calls itself / is defined in terms of itself / is re-entrant / contains within its body a reference to itself;
Ensure 'it' refers to procedure, if meaning program or object no mark **1**

(b) the current state of the machine must be saved/preserved so can return correctly to previous invocation of B;

or

return address / procedure parameter / status register / other register values / local variables must be saved/preserved so can return correctly to previous invocation of B**)**; **1**

(c)

Call Number	Parameter
1	53
2	26
3	13
4	**6**
5	**3**
6	**1**

;
;
;

Printed Output: 1 1 0 1 0 1;;; *one mark for each correct pair of bits mark from left and stop marking when error encountered ignore punctuation. if more than 6 bits give a max of 2 marks* **6**

(d) conversion (of a denary number) into binary; **1**

Total **9**

6. Compare Newcastle with (middle item of list), Manchester;
Compare Newcastle with (middle item of upper sublist), Sheffield;
Compare Newcastle with Newcastle // compare only item (in lower sublist of this upper sublist) to get a match;
Lose 1 mark if Newcastle not explicit in comparison stop marking from time it goes wrong

or

List[4] = Newcastle? False; **A** [4] = Newcastle **R** 4 = Newcastle
List[6] = Newcastle? False;
List[5] = Newcastle? True;

if formula explicit, follow through on formula **3**

Total **3**

7. (a) (i) sequence of execution of instructions determined by programmer;
R user *instead of* programmer **R** a specific order
Example: Pascal/Fortran/Basic/ C/C++/Cobol/Algol; **R** Visual… **2**

(ii) program is a set of facts and rules
// programmer declares what has to be done but not how to do it;
Example: Prolog/LISP/SQL; **2**

(iii) execution of code dependent on an event such as user clicking mouse;
Example: Visual Basic / Delphi / C#/ Java /Smalltalk/ Visual C++;
Visual ….; **R** VB **2**

(b) (i) joining together of code/procedures/methods
and data/properties/characteristics/record/attributes into objects;
to max **1**

(ii) Member (=) Class // Class (=) Member *1 mark for keyword Class; 1 mark for Member;*
(**A** Object *instead of* Class)

Public *1 mark for keyword Public in context*;

(procedure) AddNewMember(Details);
(procedure) Amend(Member)Details;
(procedure) Display(Member)Details;
} *1 mark for correct procedure definitions;*

Private *1 mark for keyword Private in context;*

Surname: String;
Firstname: String;
TelephoneNumber: String;
Handicap: integer;
} *1 mark for correct data fields & data types; don't allow extra fields*
A protected;
A text *instead of* string

End;

No marks for a diagrammatic answer
to max **4**

Total **11**

8. (a) (i) the concurrent execution // apparent simultaneous execution (over the same time period) of two or more tasks // the concurrent execution of a group of co-operating/a single user's tasks;
A programs/processes **R** jobs / applications
between which communication is possible // to achieve some common goal;
R descriptions which imply multi-user or batch O.S. **2**

(ii) memory on hard disk;
(used when execution of a program/process)
where total virtual address space exceeds / program and data / main memory needed exceed the physical/main memory capacity;
to store pages (of the process) / parts of the process not currently needed;
A program *instead of* process
lets user think there is more main memory/RAM // not enough main memory;
R computer fooled into thinking more main memory
A disk is used as RAM;
to max **2**

(iii) physical memory/RAM is conceptually divided into a number of (fixed size) page frames; **A** pages/segments;
(virtual address space of) program/process is divided into a number of (fixed size) pages;
page table indicates which pages of process are loaded (and where);
A page table keeps track of pages;
pages are swapped between disk and main memory as required
// pages are swapped into, and out of, memory as required;
No marks for a point which references just data when it should reference a program backing store is not acceptable as a substitute for disk
to max **2**

(b) (i) a thread is the processing performed on a single set of data in the system;
a thread is a process that shares most of its environment;
threads may be distinguished only by the value of their program counters and stack pointers;
several threads share one copy of program code;
a thread executes in the address space of its parent process; sharing global variables but with its own local variables;
a thread is a line of execution within a process; it has its own program counter, stack pointer and register values but runs in the same address space as other threads in the process; **2**

(ii) threads have access to the same memory so they can communicate easily;
multi-threading allows threads to access the same data as they can access the same area of memory (RAM), separate processes do not allow this as they are self-contained;
only one copy of the program needs to be loaded;
saves main memory;
threads share more of their environment with each other than do processes;
faster execution than separate processes // faster execution overall;
Allow carry forward/back between (i) and (ii)
to max **1**

Total **9**

Grand Total **65**