GCE 2005
January Series

ASSESSMENT and QUALIFICATIONS ALLIANCE

Mark Scheme

Computing Specification

CPT4 Processing and Programming Techniques

Mark schemes are prepared by the Principal Examiner and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation meeting attended by all examiners and is the scheme which was used by them in this examination. The standardisation meeting ensures that the mark scheme covers the candidates' responses to questions and that every examiner understands and applies it in the same correct way. As preparation for the standardisation meeting each examiner analyses a number of candidates' scripts: alternative answers not already covered by the mark scheme are discussed at the meeting and legislated for. If, after this meeting, examiners encounter unusual answers which have not been discussed at the meeting they are required to refer these to the Principal Examiner.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of candidates' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this Mark Scheme are available to download from the AQA Website: www.aqa.org.uk

Set and published by the Assessment and Qualifications Alliance.

The Assessment and Qualifications Alliance (AQA) is a company limited by guarantee registered in England and Wales 3644723 and a registered charity number 1073334. Registered address AQA, Devas Street, Manchester. M15 6EX. *Dr Michael Cresswell Director General*

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.**

I *punctuation i.e. accept without . (fullstop)*
Penalise once in each of (a), (b), (c), (d) for wrong case
(must be lower case for predicates and atoms, and variables must be capitalised)

1. (a) (i) contains(raitha, yoghurt).; **1**

(ii) dairy product(yoghurt).; **1**

(b) nuts;eggs; **I** *any punctuation or 'and'* **I** *order* **2**

(c) not_suitable(X,eggs).;; *one mark for not_suitable(X,), one mark for* 'eggs' *in correct place*
A contains (X,eggs);;*marking as above*
A not_suitable(X,eggs) IF contains (X, eggs); *give one mark only, as not a goal but a rule*
T.O. *if anything else added to correct answer* **I** *?* **2**

(d)

Not_for_vegetarians(X) IF meat(X) OR contains(X,Y) AND meat(Y).;;;

1 mark 1 mark 1 mark

If AND then T.O. of one mark

A *above written as 2 clauses*

A not_suitable *instead of* contains **3**

I *case for* AND. AND *can be represented by a comma*

R contains (X,meat) **R** contains (Meat,X)

accept any variable for Y but must be the same variable in both cases.
If not do not give mark for meat(Y).

Total 9

2. (a) system resources exist in separate nodes of a network (with transparent access by users);
Tasks can be shared between different nodes; *must imply shared processing*
R *idea of mainframe and terminals* **1**

(b) network with File server / print server / webserver; **A** client-server; **1**
A *an application or specific example e.g. search engines, Internet*
R *brand names*

(c) able to share resources; **A** *data, hardware for resources*
share processing of task among several processors;
local processing done locally;
maximise use of resources // faster execution of complex calculation **2**

Total 4

3. (a) (i) *Multi-programming:*
concurrent/apparent simultaneous; execution of two (or more) programs;
A two (or more) programs seem to be running at the same time;; **2**
A *job, process, thread, sequence of instructions instead of program*

(ii) *Process:* a program currently executing//waiting to be executed
an instance of a program: // a program in a phase of execution;
R *task/application instead of program* **1**

(b) (i) *allow addresses in the Pointer column.*

Position	Name	Running Time	Address	Pointer
1	Process6	7	01400	4 (02300)
3	Process7	17	01700	5 (04100)
4	Process2	17	02300	3 (01700);
5	Process9	45	04100	-1; **A** 0;
6	Process5	2	01200	8 (01900)
8	Process19	5	01900	1 (01400);

1 mark for 4,5,3 correct

1 mark for null pointer correct **A** sensible const. Name representing null pointer

1 mark for 8,1 correct

3

(ii) array; of records; *OR* linked list; of records; *OR* 4 1-D arrays; one for each column; *OR* one 1-D array for process name; one 2-D arrays for numerical data; **2**

(iii) *Marks to be allocated as follows:*

1 for initialisation ListPointer ← HeadPointer;
1 for while not at end of list While ListPointer <>-1 Do;
1 for printing Print ListArray[ListPointer].Name;
1 for getting next pointer ListPointer ← ListArray[ListPointer].Pointer;
P1 *if headpointer is reassigned*

Any name acceptable for ListPointer and ListArray

Note: a sorting method gets a maximum of 3 marks (inefficient) **4**

Alternative solution:
REPEAT UNTIL next=-1 *OR* IF listpointer <>-1 then REPEAT….

(iv)

List	*Reason*
List of suspended/blocked/halted/ unrunnable processes;	waiting for a resource or complete a requested I/O transfer;
List of inactive/dormant jobs;	Waiting to be admitted to the system;

I currently running processes **I** interrupt **2**

Total 14

4. (a) interrupt: a signal; (from a device) seeking the attention of the processor; **2**

(b) interrupting device supplies;
an offset; **A** index / indexed address **R** index register
added to the base address; **A** base register *instead of* base address

added to the base address; **A** base register *instead of* base address

gives (start) address of interrupt service routine / ISR / interrupt handler; **3**

A location as BOD

OR

The number supplied by interrupting device;

Is an absolute address of a location;

That is the (start) address of the ISR / interrupt service routing / interrupt handler;

A location as BOD

I *anything about priorities*

Note: Question is not about how an interrupt is being serviced

Total 5

5. (a) produces re-useable code because of inheritance/encapsulation; Produces re-useable objects;
data is protected // only accessible in well-defined ways (because of encapsulation);
more efficient to write programs which use pre-defined / inherited objects / classes;

storage structure of data and method code of a class may be altered without affecting programs that make use of the class;

code produced contains fewer errors / more reliable;

solutions are easier to understand (when expressed in terms of objects);

easier to enforce design consistency; easier to debug;

less maintenance effort required by developer since objects can be re-used;

new functions can be added to objects easily (because of inheritance);

R Easier to program **I** *references to GUIs* **2**

(b) *1 mark for correct base class and derived classes incl. containers;*

1 mark for 2 correctly directed arrows; **2**

R E-R diagrams
I methods listed in containers

(c)
```
Member = Class
(Public)
(procedure) AddNewMember(s);
(procedure) AmendMember(s)
(Procedure) ShowMember(s);
Private
MembershipNo : Integer
FirstName: String/text
Surname: String/text
TelephoneNumber: string/text
End (Class)
```

A proc instead of procedure
R function instead of procedure

; *no mark if methods are private*

; *1 mark for all data fields marked as private*

A string/text *as data type* **R** number

;

A ID
A FName
A SName
A Tel

; **R** *number/integer as data type* **4**

Public may come after Private. Each line may be preceded by Public or Private & in no particular order **R** *diagrammatic answer* **I** *case* **I** *white space*

Total 8

6. (a) the set / list of bit patterns / binary codes representing machine operations;
the set / list of bit patterns / binary codes for which machine operations have been defined;
The collection of different operations available; **A** complete set *instead of* the set
Note: must imply all available opcodes
R *anything that sounds like a program* **1**

(b) a storage unit where datum/data item/instruction is temporarily stored;

storage unit that can be accessed rapidly;

a special high-speed memory location;

storage unit internal to the processor;

storage unit that can be symbolically identified; **1**

Must imply individual items, eg during FE cycle
R *piece of memory instead of storage unit*
Answer must distinguish register from buffer

(c) (i) 75 is the number to be copied/stored in the accumulator/load accumulator with value 75; **1**

(ii) The contents of the accumulator is to be copied/stored at address 75;
A location *inst. of* address **1**

(iii) (The contents of location) 75+;(X)/contents of X; (is to be loaded into the accumulator) **2**

(iv) The contents of the accumulator is to be stored in the location whose address is stored; at
address 75;
A *diagrammatic answer* **2**

(d) (i) 195; **1**

(ii) as a shorthand / because it is easier/quicker to read (than lots of 0's and 1's) /less likely to make mistakes; **1**
Saving space **T.O.,** saving processor time **T.O.**

(iii)

Program Counter	**Content of Index Register X**	**Content of Accumulator A**
00A3	-	-
00A4	1;	-
00A5	1	C3 / 195 / 11000011;
00A6	1 ;	C8 / 200 / 11001000;

Accept trace table moved up 1 row **4**

(iv) contents of accumulator / C8 / 200 / 11001000 is stored at address 00A2; **1**

Total 15

7. (a) BE4; *must be capital letters* **1**

(b) 190.25 / 190 $^{1}/_{4}$;; *one mark for correct integer part,*
one mark for correct fractional part
one mark for correct working
(e.g. correct place values) **3**

(c) -1052;; *1 mark for workings if result incorrect*
1 mark for sign, 1 mark for 1052 **2**

(d) (i) -8.25 / -8$^{1}/_{4}$;;; *partial marks for workings if result incorrect*
1 mark for sign, 1 mark for moving binary point 4 places or showing 2^{4} **3**

(ii) starts with 1 0
the first 2 binary digits are different;
a significant bit is stored after the (implied) binary point;
bit after (implied) binary point different from bit before binary point; **1**
A all leading 1's have been removed // there are no leading 1's;
R there are no leading zeros

Total 10

END OF CPT4 MARK SCHEME