UNIVERSITY OF CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the May/June 2012 question paper

for the guidance of teachers

9691 COMPUTING

9691/31 Paper 3 (Written Paper), maximum raw mark 90

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes must be read in conjunction with the question papers and the report on the examination.

- Cambridge will not enter into discussions or correspondence in connection with these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2012 question papers for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level syllabuses and some Ordinary Level syllabuses.

UNIVERSITY of CAMBRIDGE
International Examinations

1 (a) (i) The table/each student has a repeated group of attributes // each student has a number of subjects [1]

(ii) StudentName, TutorGroup and Tutor would need to be repeated for each record [1]

(b)

Table: Student

StudentName	TutorGroup	Tutor
Tom	6	SAN
Joe	7	MEB
Samir	6	SAN

Table: StudentSubjectChoices

Student Name	Subject	Level	Subject Teacher
Tom	Physics	A	SAN
Tom	Chemistry	A	MEB
Tom	Gen Studies	AS	DIL
Joe	Geography	AS	ROG
Joe	French	AS	HEN
Samir	Computing	A	VAR
Samir	Chemistry	A	MEB
Samir	Maths	A	COR
Samir	Gen. Studies	A	DIL

Mark as follows ....
Complete Student table [1]
Repetition of StudentName in StudentSubjectchoices table [1]
Complete columns 2, 3, and 4 [1]

(c) (i) *primary key...*
- an attribute/combination of attributes
- chosen to ensure that the records in a table are unique // used to identify a record/tuple [2]

(ii) StudentName + Subject Correct Answer Only [1]

(iii) - there is a one-to-many relationship // Student is the 'one side' table – StudentSubjectChoices is the 'many side' table.
- The primary key (attribute StudentName) in Student
- Links to StudentName in the StudentSubjectChoices table
- (StudentName in the) StudentSubjectChoices table is the foreign key // StudentName is the foreign key that links the two tables [MAX 2]

(d) - There are non-key attributes ...
- SubjectTeacher ...
- dependent only on part of the primary key (i.e. Subject) // partial dependency [MAX 2]

(e) - There are dependent <u>non-key</u> attributes // there are <u>non-key</u> dependencies
- TutorGroup is dependant on Tutor // Tutor is dependent on TutorGroup [2]

[Total: 14]

2 (a) 83 [1]

(b) 153 [1]

(c) –110 [1]

(d) (i) +13
mark as follows:
Exponent: +4 // move the pattern four places
Mantissa: +13/16 // 0.1101
Answer: $13/16 \times 2^4$ // or equivalent [3]

(ii) There will be a unique representation for a number
The format will ensure the number is represented with the greatest possible/more accuracy/precision
Multiplication is performed more accurately/precisely [MAX 1]

(iii) Mantissa: 0100 0000
Exponent: 1000
Therefore number is $½ * 2^{-8}$ // +1/512 // $+2^{-9}$ // 0.00195 [3]

(e) choices made will effect range and accuracy
More bits used for the mantissa will result in better accuracy
More bits use for the exponent will result in larger range of numbers [Max 2]

[Total: 12]

3 (a) Boolean [1]
Flags whether or not the requested customer name is found [1]

SearchName [1]
Index [1]
Index + 1 [1]
Index = 2001 // Index >= 2001 // Index > 2000 [1]
IsFound = FALSE // NOT IsFound // Index = 2001 // Index > 2000 [1]

(b) - values are considered in <u>sequence</u>
- when an item is not found all items are considered
- Few comparisons are needed if the value is near the start of the list // Many comparisons are needed/it's time consuming if the value is near the end of the list
- The average number of comparisons needed will be N/2 (or 1000 for this data set) [MAX 2]

(c) (i) The values must be in order
<u>Calculate</u> the middle value and compare with the requested value
If Requested value is less/greater discard the top/bottom list
Repeat with a new list // compare with a new middle value
Continue until value is found or list is empty [MAX 4]

(ii) *Compare with ...*
Kiwi
Banana
Cherry [3]

[Total: 16]

4 **(a)** 21 [1]

(b) **(i)** a 5 – b c + / [1]

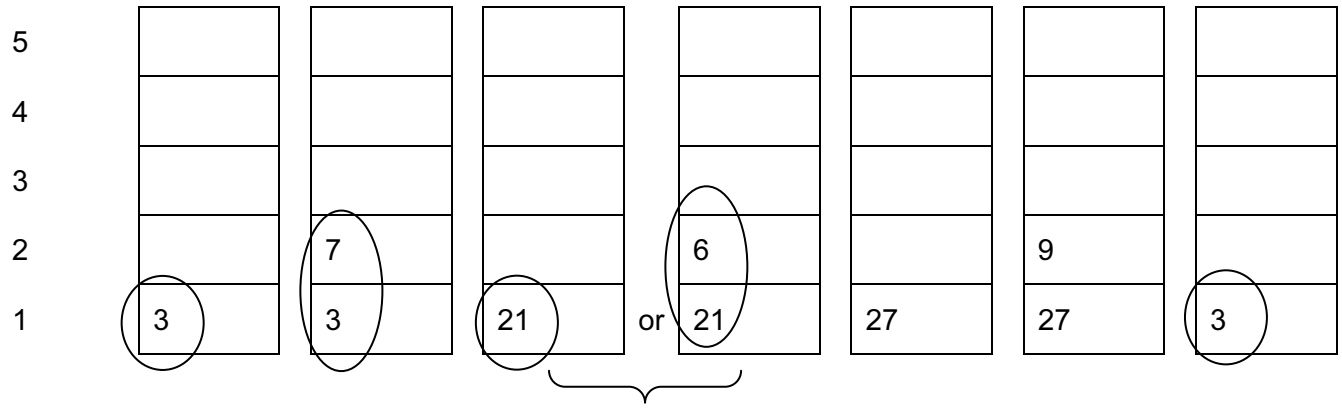
(ii) 2 3 * 6 2 / + [2]

(c) Expressions can be evaluated without the use of brackets
Operators are in execution order / No need to apply a precedence of operators [1]

(d) **(i)** Last item added to the stack will be the first item to leave [1]

(ii) **Static** structure
The size of the array will be fixed / size will be defined before the array is used [2]

(iii)



1 1 1 1 [4]

[Total: 12]

5 **(a)**

LDD 105

Accumulator

0001 0001

Main memory

100	0100 0000
101	0110 1011
102	1111 1110
103	1111 1010
104	0101 1101
105	0001 0001
106	1010 1000
107	1100 0001
200	1001 1111

Mark as follows:
- Sensible annotation which makes clear 105 is the address used
- Final value in Acc [2]

(b)

LDX 101

Accumulator

0101 1101

Index Register

00000011

Main memory

100	0100 0000
101	0110 1011
102	1111 1110
103	1111 1010
104	0101 1101
105	0001 0001
106	1010 1000
107	1100 0001
200	1001 1111

Mark as follows:
- IR contents converted to 3
- Computed address of 101 + 3 = 104
// explanation: add contents of IR to address part of instruction
- Then, 'direct addressing' to 104
- Final value in Acc [MAX 4]

(c)

Accumulator	Memory Address 507	508	509	510
	22	170	0	0
22				
23				
			23	
170				
171				
				171

Mark as follows ...
- 22 to Accumulator
- Incremented to 23
- 23 copied to address 509
- 170 copied to Accumulator and incremented to 171
- 171 in address 510 [5]

(d) Every assembly language instruction is translated into exactly one machine code instruction / there is a 1-to-1 relationship between them [1]

[Total: 11]

6 **(a)** Decide which process ...
Gets next use of the processor (low level scheduler)
// is next loaded into memory (high level scheduler)
maximise system resources [2]

(b) **(i)** Running
The process currently has the use of the processor

Runnable/Ready
The process would like to use the processor but the processor is currently in use by another process

Suspended/Blocked
The process is not capable of using the processor / the process is currently occupied doing I/O [6]

(ii) Maintain a separate 'data structure' for the processes in each state
one field of the Process Control Block will store the current state [1]

(c) **(i)** *Processor bound ...*
The process does very little I/O // the process requires the processor most of the time
3D-graphics calculation // any plausible application

I/O bound ...
The process does lots of I/O // the process requires little processor time // any plausible application [4]

(ii) Priority to I/O bound processes
Otherwise they will not get a look in // processor bound jobs would monopolise the processor [2]

[Total: 15]

7 **(a)** a model/program of the real-world system is produced
to predict the likely behaviour of a real-world system [2]

(b) *Computer system suitable as ...*
A computer program/system can be written/created which model the problem/application
The problem can control the values of all the variables/parameters
The computer can produce results very quickly // e.g. models what actually takes several days into 5 minutes processing
The simulation removes any element of hazard/danger
Some real-world problems are impossible to create
It will be cost-effective to model the problem first [MAX 2]

(c) Time taken to serve a customer
Number of items in the customer basket
Acceptable wait time in the queue
Number of checkouts
Time of day/day of the week
Number of customers arriving
Speed of the checkout operators
Anything plausible ... [MAX 3]

(d) - Increase the average time taken to serve a customer
- … will increase the average queue length
Or anything plausible ... [2]

[Total: 9]