UNIVERSITY OF CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the May/June 2012 question paper

for the guidance of teachers

9691 COMPUTING

9691/33 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.

1 (a) (i) The table has a repeated group of attributes // each aircraft has a repeated attributes

(ii) AircraftID, Type and YearBought would have to be repeated for all records // FlightCode, Departure and Arrival are the repeated group [1]

(b) (i) The Aircraft table would contain:

AircraftID	Type	YearBought
1	747	1998
2	747–400	2007
3	747–400	2007

[1]

(ii) 10 records [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) AircraftID [1]

(d) (i) *foreign key*
An attribute/field in one table
Which links to the primary key in another table [2]

(ii) AircraftID [1]

(e) - the two non-key attributes // Country & NumberOFRunways
- are not dependant on each other [2]

(f) *data inconsistency ...*
The data value in one table does not match up with what should be the same data value in a second table. [1]

[Total: 13]

2 (a) (i) N [1]

(ii) 4E [1]

(b) (i) Addition and subtraction calculations give the correct result (provided the answ within range)
There is only one representation for zero
All the bits have a place value [MAX

(ii)

–13	1	1	1	1	0	0	1	1	
+59	0	0	1	1	1	0	1	1	+
	0	0	1	0	1	1	1	0	
1	1	1	1			1	1		

1 mark for correct –13 binary
1 mark for correct +59 binary
1 mark for the correct binary addition **showing carry evidence** [3]

(c) (i) –88
mark as follows:
Exponent: +7 // move pattern 7 places
Mantissa: –11/16 // 1.0101
Answer: $-11/16 \times 2^7$ // or equivalent [3]

(ii) The mantissa/the binary pattern starts with 10 // the first two bits of the mantissa/the binary pattern are different [1]

(iii) Mantissa: 1000 0000
Exponent: 0111
Denary: –128 // -2^7 // $-1 * 2^7$ [3]

[Total: 14]

3 (a)

HeadPointer = 5

	Country
1	SWEDEN
2	DENMARK
3	INDIA
4	COLUMBIA
5	BANGLADESH
6	NEPAL
7	MAURITIUS

	Pointer
1	0
2	3
3	7
4	2
5	4
6	1
7	6

Mark as follows:
HeadPointer = 5 [1]
COLUMBIA – 2 and DENMARK – 3 [1]
All others correct [1]
SWEDEN has a 'null pointer' [1]

(b) IF HeadPointer = NULL/0/-1
NoMoreValues ← FALSE
Current ← Pointer[Current]

(c) Input the country

(d) If headpointer = 0 then list empty
Move to the head position
REPEAT
    IF this country is > the value input / first value found
    ELSE Move to the next value
UNTIL value found
REPEAT
    OUTPUT all values after this one
UNTIL null pointer found

Mark points:
- Special case test for empty list
- Input country
- Move to headpointer position
- Comparison
- Repeat until value found
- Loop to output all values [MAX 4]

(e) - Search the linked list until delete value is found
- some change takes place to the Pointer array // the links are changed
- Pointer[Previous] ... // Previous' pointer changes to …
- ← Pointer[Current] // … the value of Current's pointer
- The space for position Current can be returned to the pool of 'free space' [MAX 4]

[Total: 16]

4 **(a)** 15 [1]

(b) **(i)** c 5 + b c – /1

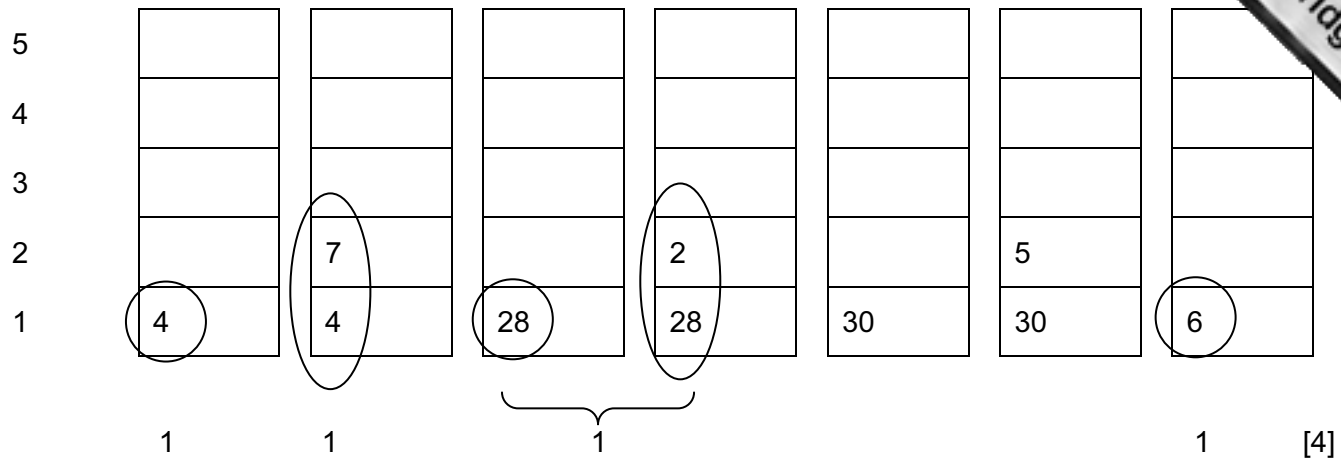
(ii) 3 9 * 6 2 / – [2]

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

(d) **(i)** last item added to the stack will be the first item to leave (N.E LIFO) [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)** a <u>model/program</u> of the <u>real-world</u> system is produced
to <u>predict</u> the likely behaviour of a <u>real-world</u> 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) Rate at which cars arrive on new road
Rate at which cars arrive on existing road
Timing intervals of the lights on new road / existing road
Day of the week / time of day
Number of lanes
Is there a pedestrian time interval?
Anything plausible ... [MAX 3]

(d) - Increase the rate on arrival of cars ...
- … will increase the average queue length
Or any plausible input and resulting output... [2]

[Total: 9]

6 (a)

LDD 66

Accumulator

1010 1000

Main memory

60	0110 0000
61	0100 0000
62	1111 1110
63	1111 0000
64	0101 1101
65	0001 0001
66	1010 1000
67	1100 0001
100	1001 1111

Mark as follows:
- Sensible annotation which makes clear 66 used
- Final value in Acc [2]

(b)

LDI 61

Accumulator

0101 1101

Main memory

60	0110 0000
61	0100 0000
62	1111 1110
63	1111 0000
64	0101 1101
65	0001 0001
66	1010 1000
67	1100 0001
200	1001 1111

Mark as follows ...
- Go to address 61 // shows arrow to 61
- Pick up the forwarding address 64 // shows arrow to 64

Correct final contents copied to Acc // shows arrow from contents of 64 to Acc [3]

(c)

Accumulator	Memory Address 207	208
	16	150
150		
151		151
16		
17		
	17	

Mark as follows ...
- 150 to Acc
- Incremented to 151 and copied to 208
- 16 copied to Acc and
- incremented to 17 copied to address 207

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

Total: 10

7 **(a)** *An interrupt*
a signal/message from some device
to indicate that some event has occurred //the device is seeking the attention of the processor [2]

(b) Identify the source of the interrupt
Disable all interrupts of a lower priority
Save the contents of the PC
Save the contents of the other registers ...
Onto the stack
Load and run the appropriate ISR code
Restore the registers
From the stack (stack mentioned 1 mark only ...)
Enable all interrupts
Continue execution of the interrupted process [MAX 6]

(c) - Partitioning
- Memory is divided into partitions
- One or more programs loaded into each partition
- Different partitions used for different types of job
- Partitions can be of fixed size or dynamic
- Programs are scheduled when partition has space for whole program

OR ...

- Paging / Virtual memory
- The program is divided into a number of pages // The main memory is divided into a number of page frames (of the same size)
- Not all pages of the program need to be initially loaded
- Pages swapped in/out of memory as required
- use of page table

OR

- segmentation
- Programs are divided into segments by the programmer
- Not all segments are initially loaded // segments are loaded as and when required during execution
- segments can be of varying size

(d) Estimated run time
A run priority // based on time to completion / time to deadline
Estimated memory requirements
Resources required
User priority [MAX 3]

[Total: 17]