CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the May/June 2013 series

9691 COMPUTING

9691/32 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 should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2013 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

1 **(a)** **(i)** Many-to-many [1]

(ii) E-R diagram

[1]

(iii)

Link table drawn [1]

2 × one-to-many relationships [1]

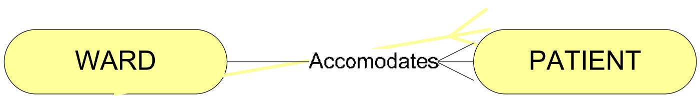
primary key in DOCTOR links to foreign key in link table [1]

primary key in PATIENT links to foreign key in link table [1]

No mention of foreign key scores max 1 for final two points …

(b) **(i)** One to many [1]

(ii) E-R diagram



[1]

(c) The primary key of table WARD - WardName [1]

Matches to WardName in the PATIENT table [1]

(d) Displays a 'list' of the ward**s** (names) [1]
R. Number of wards
Which has unoccupied beds available [1]
R. the condition explained using the attribute identifiers

[Total: 12]

2 **(a)** Meta language
Rules // Grammar (which describe a high level programming language / protocol specification)
The <u>syntax</u> or <u>structure</u> of all program statements [2]

(b) **(i)** A <u>rule</u> which is defined in terms of itself
NB Not 'procedure' ... [1]

(ii) Rule 3 [1]

(iii)

Expression	Valid / Invalid	Rules used		
0	Invalid	1 , 4	4, 2	[1 + 1]
"1"	Valid	4 then combination of 1,2 and 3	combination of 1,2 and 3, end with 4	[1 + 1]
"001"	Valid	4 then combination of 1,2 and 3 AND rule 3 used more than once	combination of 1,2 and 3 with rule 3 used more than once, ends with 4	[1 + 1 + 1]

(c)
```
<Dollar> ::= $

<BinaryString> ::= <Paren theses><Binary><Parentheses>

                   |<Parentheses><Dollar><Binary><Parentheses>
```

Note: credit alternative answers which use an intermediate expression [2]

[Total: 13]

3 **(a)** Direct addressing / LDD [1]

(b) Indexed addressing / LDX [1]

(c) Annotation to show 203 used as a forwarding address [1]

Accumulator contains 38 [1]

(d)

	Memory location		
ACC	109	110	Output
19 **(must be the first column entry)**		0	
20	20		
37			
38			
58		**58 /ft**	**58 /ft**

1 mark for each of the emboldened numbers in the correct column and sequence [MAX 5]

(e) Labels added to a (symbol) table // creates a list of addresses [1]

Labels are later looked up to determine the actual address / Assembler must allocate addresses to labels [1]

Mnemonic looked up to give binary code/machine code [1]

Macro instructions are expanded into a group of instructions [1]

The software makes two passes through the source program [1]
[MAX 3]

[Total: 12]

4 (a) (i) Dynamic data structure changes size ... [1]

At execution time [1]

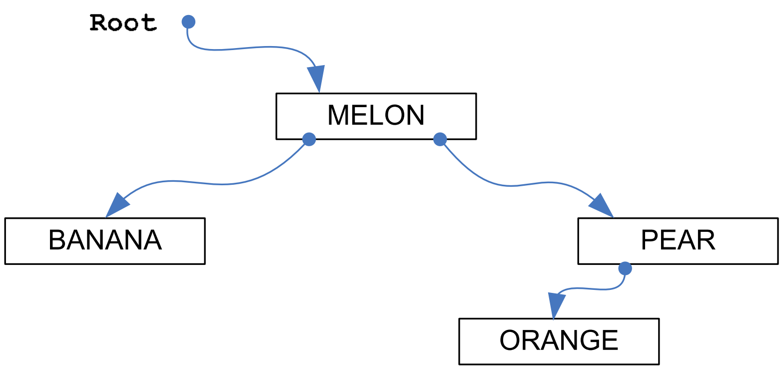
// A static data structure has a fixed size [1]
[MAX 2]

(ii) Dynamic data structure matches size to data requirements [1]

Takes memory from heap as required //
returns memory as required (following node deletion) [1]

There is no wasted memory space / makes efficient use of memory [1]
[MAX 1]

(b)



Root is MELON1 [1]
Correct left subtree [1]
Correct right subtree [1]

(c) (i) InOrder(LeftP[Root]) // InOrder(RightP[Root]) [1]

(ii)

```
InOrder(PEPPER)
LeftP[PEPPER] <> 0 is TRUE
    InOrder(BEAN)
    LeftP[BEAN] <> 0 is FALSE
    OUTPUT BEAN
    RightP[BEAN] <> 0 is TRUE
        InOrder(PEA)
        LeftP[PEA] <> 0 is FALSE
        OUTPUT PEA
        RightP[PEA] <> 0 IS FALSE
        ENDPROCEDURE
    ENDPROCEDURE
OUTPUT PEPPER
RightP[PEPPER] <> 0 is TRUE
    InOrder(PUMPKIN)
    LeftP[PUMPKIN] <> 0 is FALSE
    OUTPUT PUMPKIN
    RightP[PUMPKIN] <> 0 is FALSE
    ENDPROCEDURE
ENDPROCEDURE
```

[4]

(iii) The procedure has to backtrack/unwind from the current call [1]

To return to the calling procedure // return to the addresses from which called [1]
[MAX 1]

[Total: 12]

5 (a) (i) *The keyword table contains:*

all the language keywords/reserved words + <u>with a matching token</u> [1]

The symbol table stores:

each identifier/variable found (and its data type) [1]

the values of all constants [1]

the upper and lower bounds of arrays [1]

[Mark as: 1 + 1]

[MAX 2]

(ii) Keywords are looked up in the keyword table [1]

Keywords are converted to tokens [1]

Identifiers/Variables are looked up in the symbol table [1]

Identifiers/variables are converted to actual addresses [1]

[MAX 2]

(iii) The white space // redundant characters are removed [1]

Illegal identifier names are identified [1]

[MAX 1]

(b) (i) *Optimising*

Code will <u>execute/run/process</u> faster [1]
Code requires less memory
Reduce the amount of code [1]

R. 'more efficient' <u>// removes redundant code</u>

(ii) Any example where the code could be changed [1]
E.g. input of a list of number to compute the total (There would be no need to store the numbers first)

[Total: 8]

6 (a) (i) *Batch processing*
All input/processing/output is performed as a batch [1]

There may be a time delay before processing [1]

All the (data) is processed together/at the same time [1]

There is no user involvement [1]

Processing will not start until all the data is available/collected [1]
[MAX 3]

(ii) *Interactive processing*
The user is constantly interacting directly with the processor [1]

(b) (i) PROG23 [1]

(ii) Any two from PROG17, PROG44 and 45 [1]

(iii) Jobs do not have to occupy a continuous block of memory [1]

Move all jobs still loaded in the partition so that when a job completes there is only ever one 'hole' remaining [1]

Make the partitions of variable size [1]

Allow only part of a program to be initially loaded // paging //segmentation [1]
[MAX 2]

(c) Operating system // specific modules e.g. interrupt handler/scheduler, etc [1]

device drivers [1]

<u>examples</u> of system software or utilities [1]

R. "System software" and "Utilities" [MAX 2]

(d) Runnable // Ready [1]

the program is capable of being run and is awaiting its turn for the use of the processor [1]
R. explanation of (only) 'ready to use the processor'

Suspended // Blocked [1]

the program is unable use the processor/ or by example, the job is currently using an I/O device [1]
Note: the explanation marks are not dependant on the correct name

[Total: 14]

7 **(a)** **(i)** 2 [1]

(ii) COMPILE ERROR // reporting an error [1]

(iii) 0 [1]

(iv) COMPILE ERROR // reporting an error [1]

(b) **(i)** `FUNCTION StringFound(ThisArray : STRING , UBound : INTEGER, ThisValue : STRING) RETURNS BOOLEAN`

Mark as follows:
`FUNCTION StringFound` [1]

`'Array variable' : STRING data type` [1]

`ThisValue : STRING // 'UBound' : INTEGER` [1]

(ii) Numbered 1 – Parameter identifiers labelled [1]

Numbered 2 – `(RETURNS) BOOLEAN` [1]

(iii) `CityWasFound = StringFound(CapitalCities, 300, "LISBON")`

Mark as follows:
`CityWasFound = StringFound( ...` [1]

"LISBON" is the correct position (f/t from 'their' function header) [1]

[Total: 11]

Question 8

(a) **(i)** Example [1]

(ii) *two of the points explained ...*

moveable …
mechanical device ...
sense its surroundings ... ... clear example // temperature, etc.
controlled by a program ... [MAX 2]

(b) Robotic arm [1]

Explained in the context of 'their' robot [1]

Sensor [1]

Capture data [1]

Actuator // Motor [1]

To drive various motors to perform the robot's movement [1]

Microprocessor [1]

To process the various inputs and execute the control program [1]

Camera [1]

To capture images [1]

Memory [1]

To temporarily store input data [1]

Speaker [1]

To provide audio output [1]
[MAX 4]

(c) real-time [1]

[Total: 8]