CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the May/June 2013 series

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 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.

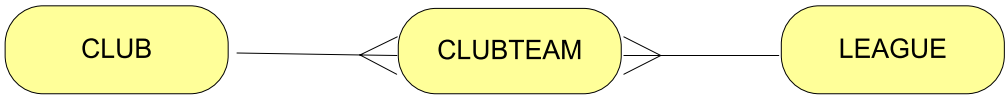
Cambridge is publishing the mark schemes for the May/June 2013 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

CAMBRIDGE
International Examinations

1 (a) (i) Many CLUBs play in many LEAGUEs. [1]

(ii) E-R diagram [1]

(iii)



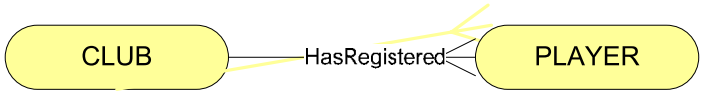
– Link table drawn [1]
– 2 x one-to-many relationships [1]

– primary key in CLUB links to foreign key in link table [1]
– primary key in LEAGUE links to foreign key in link table [1]

No mention of foreign keys scores max 1 for final mark points

(b) (i) One CLUB has many PLAYERs [1]

(ii) E-R diagram [1]



(c) The primary key of table CLUB – `ClubName` [1]
Matches to `ClubName` in the PLAYER table [1]

(d) Displays a 'list' of the player names and registration numbers [1]
Who are female defenders [1]

[Total: 12]

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

(b) (i) A rule which is defined in terms of itself
NB Not 'procedure' ….. [1]

(ii) Rule 6 [1]

(iii)

Expression	Valid/Invalid	Rules used		
[g]	Valid	Uses all the rules except 3		1 + 1
[dc]	Invalid	– Starts with 7 – 3 not used	– 3 not used – ends with 7	1 + 1
[w,a]	Valid	– Starts with 7 – all rules used, incl, rule 6 twice	– all rules used, incl, rule 6 twice – ends with 7	1 + 2

(c) `<DoubleCharacter> ::= <Character> < Character>`
`<ListItem> ::= <Character> | <DoubleCharacter>` [3]

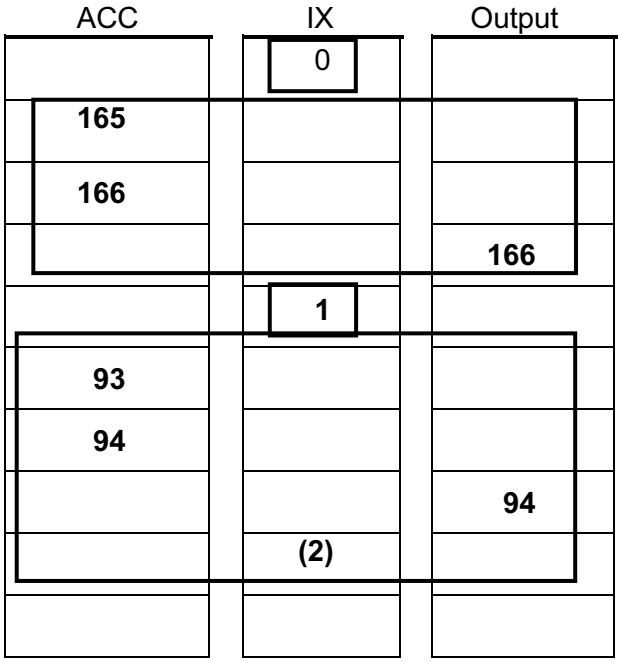
[Total: 14]

3 **(a)** Indexed addressing // LDX [1]

(b) Indirect addressing
Annotation to explain that address 203 is used as a forwarding address [2]

(c) 48 [1]

(d)



Mark as follows:
Index register contain 0 [1]
Sequence of first box (or subsequent sequence for the same instructions) [1]
Index register contains 1 [1]
Sequence for final box [1]

(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

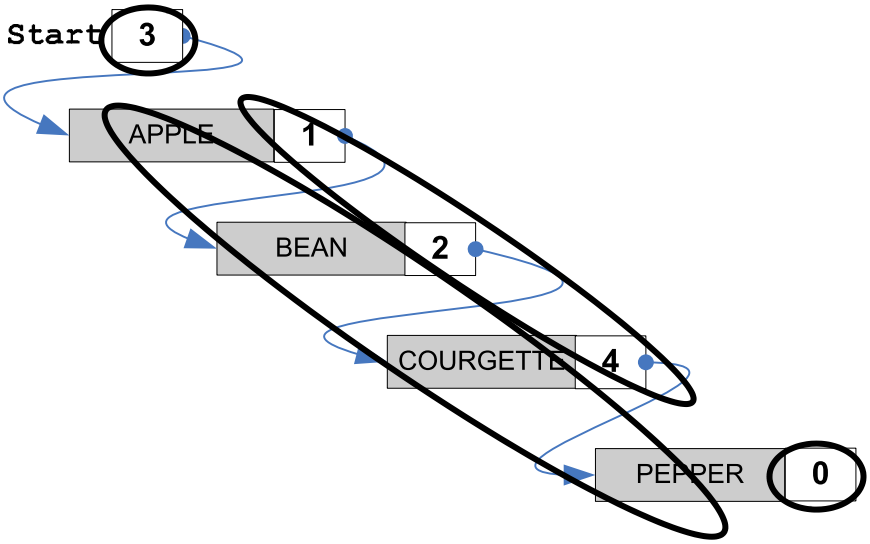
[Total: 11]

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) `MyList[Start].Data = APPLE` [1]
`MyList[3].Pointer = 1` [1]

(c)



[4]

(d) (i) ListTraversal(MyList[Index].Pointer) [1]

(ii)

```
ListTraversal(3)
MyList[3].Pointer <> 0 is TRUE
    ListTraversal(1)
    MyList[1].Pointer <> 0 is TRUE
        ListTraversal(2)
        MyList[2].Pointer <> 0 is TRUE
            ListTraversal(4)
              MyList[4].Pointer <> 0 is FALSE
              OUTPUT MyList[4].Data // OUTPUT PEPPER
            ENDPROCEDURE
        OUTPUT COURGETTE
        ENDPROCEDURE
      OUTPUT BEAN
    ENDPROCEDURE
OUTPUT APPLE
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: 15]

5 (a)
- Interpreter translates one instruction, runs it before going on to the next // Compiler translates all the instructions before run [1]
- Compiler creates object code/executable file // Interpreter does not [1]
- Interpreter makes for easier debugging //compiler errors produced away from the execution [1]
- Compiled programs will execute faster // interpreted code will execute slower [1]
- Interpreter must be present to run the program // compiler software not needed at runtime [1]
- Interpreter will translate code in loops more than once // Compiler only once [1]
- Once compiled no further translation needed // Interpreter needed every program execution attempted [1]

MAX 4

(b) (i) The keyword table contains:
all the language keywords/reserved words + with a matching token [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]
Labels are looked up in the symbol table [1]
Labels are converted to actual addresses [1]
MAX 2

[Total: 9]

6 (a) Batch is X [1]

There is a time delay before processing [1]
All the (data) is processed together/at the same time [1]
The payslips are generated as a batch [1]
Processing cannot start until all data has been collected/input // all data entered by the 18th so processing can be done on the 25th [1]
There is no user involvement [1]
MAX 2

(Interactive processing is Y)
The user continually wants to see the effect of the changes/design produced [1]
There is data input by the user [1]
MAX 1

(b) (i) STAFF17 can be loaded [1]

(ii) Partition 3 is too small [1]
It will not allow all 12 students to log-on at 09:00 [1]

(iii) Operating system // specific modules e.g. interrupt handler/scheduler, etc. [1]
device drivers [1]
examples of system software or utilities [1]

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

(iv) Running [1]
The job currently has use of the processor [1]

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

(c) (i) The program is divided into fixed sized units called pages
The memory is divided into areas (with the same size) called page frames [1]
Page management is done using a 'page frame table' [1]
Pages will be swapped in and out of page frames as required. [1]
An area on the hard disc can act as virtual memory (to speed up the swapping) [1]
Virtual memory extends memory capacity / acts as pseudo-memory
MAX 2

(ii) Not all pages of the program need to loaded [1]
saves memory [1]
more jobs can be run [1]
MAX 1

[Total: 16]

7 (a) (i) True / Yes [1]

(ii) False / No [1]

(iii) COMPILE ERROR [1]

(iv) COMPILE ERROR [1]

(b)

```
FUNCTION CalcNetPay (EmpGrade:CHAR/STRING, HoursWorked:SINGLE/INTEGER)
                                                    RETURN SINGLE
```

[1] [1] [1]

A: `RETURN REAL/CURRENCY`

[Total: 7]

8 (a) Building a model of the system ..... [1]
The model records over time the result of changing parameters/conditions //
Models the behaviour of the system [1]

(b) A computer <u>program</u> can be written to build the model [1]
The computer system can process results very quickly / can reduce the time frame [1]

Weather forecasting has many changes which are based on mathematical equations [1]
Inputs will originate from various sensors [1]
e.g. wind speed / temperature / air pressure [1]
powerful computers process results (from many sensor sources) [1]
Can use parallel processing [1]
Outputs will be produced which are based on all the available data [1]
MAX 4

[Total: 6]