CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the October/November 2013 series

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 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 October/November 2013 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

CAMBRIDGE
International Examinations

1 **(a)** **(i)** `a b + 7 /` [1]

(ii) `2 3 z * 5 + /`
`1` [1]
2nd mark for completely correct [1]

(b) evidence for 12 and 4 [1]
3 [1]

(c) **(i)** In-order traversal // (Traverse each subtree in the order) left-root-right [1]

(ii) `E M c 2 ^ * =` [1]

(iii) Post-order traversal // (Traverse each subtree in the order) left-right-root [1]

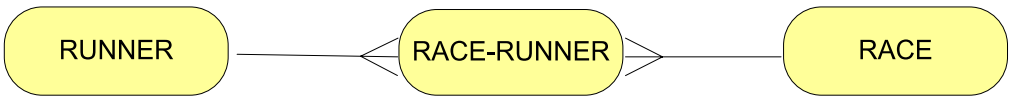
[Total: 8]

2 **(a)** Security is improved/better managed [1]
Different users can have different 'views' of/access to data [1]
Program-data independence // Changing a field does not require an applications program re-write [1]
Queries and reports quickly produced [1]
Reduced data duplication/redundancy [1]
Reduced data inconsistencies [1]
Better managed data integrity/data validation // Validation code does not need to be present in all applications programs [1]
If implemented with a DBMS it will allow concurrent access to the database [1]
MAX 3

(b) **(i)** many runners compete in many races // many-to-many // M:m [1]

(ii) one club organises many races // one-to-many // 1:M [1]

(c) **(i)**



Intermediate table (not labelled `RUNNER, RACE, CLUB`, etc.) [1]
2 X one-to-many relationship [1]

(ii) Primary key of RACE/Primary key `RaceDate`
// Primary key of RUNNER/Primary key `MemberID` [1]
Is used as a foreign key in the link table [1]

(d) **(i)** (Yes) since there is a not a repeated group of attributes [1]

(ii) (Yes) Since there is only a single attribute primary key
// there are no partial dependencies
// all non-key attr. are dependent on the primary key [1]

(iii) There are dependent non-key attributes // `ClubAddress` is dependant on `ClubName` [1]

(iv) `RUNNER(MemberID, RunnerName, RunnerDOB, ClubName)` [1]

`CLUB(ClubName, ClubAddress)` [1]

If primary key not indicated penalise once only

(e) Avoids data duplication/repeated data [1]
Avoids data inconsistencies [1]
Ensures data integrity [1]

(f)
```
SELECT RaceDate, OrganisingClubName                    [1]
FROM RACE                                              [1]
WHERE RaceDate > #01/01/2013# AND Distance < 10        [1]
```

Do not penalise imprecise syntax in the WHERE line

[Total: 19]

3 **(a)** a single processor
program consists of a sequence of stored instructions [1]
Instructions + data [1]
are stored (in a continuous block) of primary/main memory [1]
instructions are executed in sequence [1]
MAX 2

(b) **(i)** 122 [1]

(ii) 5C [1]

(iii) Fewer digits used to represent any number // long string difficult to interpret [1]
Less likely to make a mistake when copying/converting a digit string [1]
Easy to convert from binary to hex (vice versa) than binary to denary [1]
MAX 1

(c) **(i)** 16 bits [1]

(ii)

Fetch stages	Special purpose registers				Busses	
	PC	MAR	MDR	CIR	Address bus	Data bus
	7A					
MAR ← [PC]		7A			√	
PC ← [PC] + 1	7B					
MDR ←[[MAR]]			2150			√
CIR ← [MDR]				2150		

For the buses column penalise once for any additional incorrect ticks MAX 5

(d)

Instruction	Register	
	Accumulator (ACC)	Index Register (IX)
LIX 200		3
LDD 201	216	
LDI 201	96	
LDX 201	63	

1 per contents [4]

[Total: 15]

4 A class is the design/blueprint/template (from which objects are later created) [1]
A class consists of properties/attributes and methods/procedures/functions [1]

An object is an instance of a class [1]
An object must be based on a class definition [1]
Many objects can exist for the same class [1]
MAX 3

(b) The class diagram includes:

`BOOK + RECORDING` subclasses [1]

`FILM + MUSIC` subclasses of `RECORDING` [1]

Recognised notation for inheritance [1]

`RESOURCE` class `Title : STRING`
`OnLoan : BOOLEAN` [1]

`BOOK` class `Author : STRING` [1]

`FILM` class `RunningTime : INTEGER` [1]

`MUSIC` class `NoOfTracks : INTEGER` [1]

`RECORDING` class `ReleaseDate : DATE` [1]

MAX 8

(c) *Encapsulation*
Combining together of an object's properties and the methods [1]
Restricts the programmer's access to the object's data // Hiding of data [1]
Data values can only be read/written using the methods of the class [1]

[Total: 13]

5 (a) Last item added is the first item to leave // or equivalent wording [1]
R. LIFO

(b) (i) HARRIS [1]
17843 [1]

(ii)
```
PROCEDURE PushJob
IF TopOfStack = 1000                                          [1]
   THEN
      OUTPUT "Stack is already FULL"
   ELSE
      INPUT NewUserID
      INPUT NewReferenceNo                                    [1]
      TopOfStack ← TopOfStack + 1                             [1]
      SpoolJob[TopOfStack].JobReference ← NewReferenceNo
      SpoolJob[TopOfStack].UserID ← NewUserID                 [1]
   ENDIF
ENDPROCEDURE
```

(c)
```
PROCEDURE PopJob
    IF TopOfStack = -1                                    [1]
        THEN
            OUTPUT "There are no print jobs waiting"      [1]
        ELSE
            PROCESS SpoolJob[TopOfStack]
                TopOfStack ← TopOfStack - 1               [1]
    ENDIF
ENDPROCEDURE
```

(d) May not be a fair way to order the outputs [1]
Some print jobs may wait a long time before printing [1]
Better choice is a queue [1]
Since first print job sent will be the first to be output // First in – First out [1]
MAX 3

[Total: 13]

6 (a) (i) File allocation table
Storage space is organised into allocation units/clusters [1]
There is a record for each allocation unit/cluster [1]
Records are marked as either used // available // unusable [1]
Allocation units/clusters for each file are maintained as a linked list [1]
There is a separate FAT for each logical volume/partition [1]
MAX 2

(ii) Allocation units allocated to the file … [1]
Have their record status changed to 'available' [1]

(b) (i)
1. Save the contents of the program counter on the **stack** [1]
2. Also save **contents of all other registers** [1]
3. Load and run the appropriate **Interrupt Service routine (ISR)** [1]
4. Restore all other registers
5. Restore the **Program Counter** [1]
6. Continue execution of the interrupted process

(ii) Disable interrupts of a lower priority (before step 1) [1]
Check for receipt of interrupt (during Step 3) [1]
If interrupt received before completion of step 3, go to step 1
// Save the registers for the current process – the ISR [1]
Compare priority with level below which interrupts already disabled [1]
Enable interrupts of a lower priority (after Step 5) [1]
MAX 3

[Total: 12]

7 **(a)** Possible answers include:
Encryption of email traffic [1]
Email data if intercepted cannot be read [1]

Encryption of passwords [1]
Designed to prevent unauthorised access [1]

(b) *Encryption algorithm ...*
The calculation/process/sequence of steps for converting the message text/data [1]

Encryption key
A number/parameter used by the encryption algorithm // e.g. the displacement shift for transposing characters [1]

(c) *Asymmetric encryption ...*
Private key is known only to the owner//Public key is known by both parties [1]
Public and private keys are obtained from the purchase of a digital certificate //
Keys are generated at the start of a secure (e.g. web or email) session [1]
EITHER …
Sender will use their own private key [1]
Receiver decrypts using the sender's public key [1]

OR ….
Sender uses the recipient's public key [1]
Receiver decrypts using their own private key [1]
MAX 3

(d) *Authorisation ...*
Different permissions granted to different users [1]
Restricted access to certain data files/directories/physical devices [1]
User IDs MAX 1

Authentication
Passwords [1]
(Digital) signature // (Digital) certificate [1]
Use of biometric data and methods [1]
MAX 1

[Total: 11]