CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the October/November 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 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)** `x y – 5 /`   [1]

   (ii) `2` <u>`4 a * 1 +`</u> `/`
            `1`   [1]
         2nd mark for completely correct   [1]

   (b) Evidence for 12 or 6   [1]
       Answer 2   [1]

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

   (ii) `1 2 / b * h *`   [1]

   (iii) Post(-order) traversal // (Traverse/visit 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/ repetition /redundancy   [1]
       Reduced data inconsistencies   [1]
       Better managed /or similar 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 product can be supplied by one supplier // many-to-one // M:1   [1]

   (ii) Many products appear on many orders // many-to-many // M:M   [1]

   (c) **(i)**

       Intermediate table (not labelled PRODUCT, ORDER, etc.)   [1]
       2 X one-to-many relationship   [1]

   (ii) Primary key of `PRODUCT`/Primary key `ProductID` // Primary key of `ORDER`   [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 //
<u>SupplierName</u> and/or <u>SupplierTelNo</u> are dependent on <u>SupplierID</u> [1]

(iv) PRODUCT(<u>ProductID</u>, ProductDescription, RetailPrice, SupplierID)
[1]
SUPPLIER(<u>SupplierID</u>, SupplierName, SupplierTelNumber) [1]

If primary key not-indicated penalise once only

(e) Avoids data duplication/avoids repeated data // reduces data redundancy [1]
Avoids data inconsistencies [1]
Ensures data integrity [1]
MAX 2

(f) SELECT CustomerID, OrderNo [1]
FROM ORDER [1]
WHERE OrderDate = #15/01/2014# AND PaymentMethod = 'D'
(AND ISPaid = TRUE) [1]

Do not penalise imprecise syntax in the WHERE line

[Total: 19]

3 (a) <u>Temporary</u> storage location [1]
general purpose/special (purpose) [1]
Inside the (micro)processor [1]
MAX 2

(b) (i) 3C [1]

(ii) 271 [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

R. Hex is easier to understand/write

(c) (i) 2 bytes [1]

(ii) The Program Counter contains 30 [1]
```
MAR ← [PC]        // MAR given the contents of the PC
PC ← [PC] + 1     // PC is incremented
MDR ← [[MAR]]     // The contents of the address in MAR is copied to MDR
CIR ← [MDR]       // The contents of MDR are copied to CIR
```
[1]
[1]
[1]
[1]

OR … If the candidate's answer uses the suggested instruction:

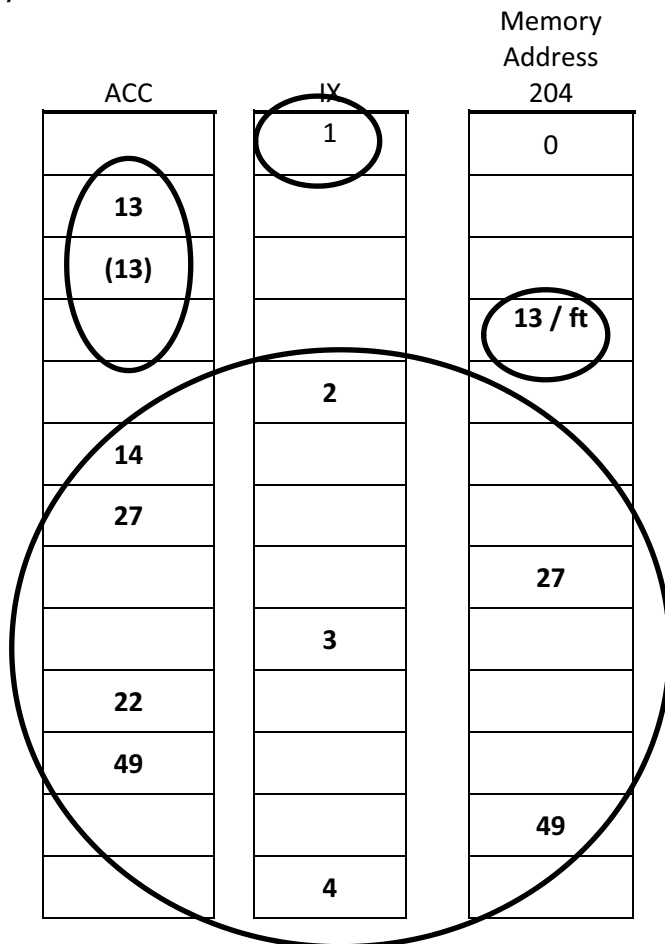The Program Counter contains 30 [1]
PC contents are copied to MAR [1]
PC contents are incremented to 31 [1]
The contents of address 30 / 2150 is copied to MDR [1]
MDR contents / 2150 is copied to CIR [1]

MAX 5

(d)

[4]

[Total: 15]

4 (a) 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 <u>instance</u> 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:

`ADMIN` + `PROJECTSTAFF`   subclasses of `EMPLOYEE`   [1]

`PROGRAMMER` + `TECHAUTHOR`   subclasses of `PROJECTSTAFF`   [1]

Recognised notation for inheritance   [1]

`EMPLOYEE class FullTimeIndicator  : BOOLEAN // CHAR`   [1]
`              Salary(Grade) : any except DATE/BOOLEAN`   [1]

`ADMIN class        Department : STRING`   [1]

`PROJECTSTAFF class ProjectTeam : STRING`   [1]

`PROGRAMMER class   ProgrammingLanguage : STRING`   [1]

`TECHAUTHOR class   SoftwareSpecialism : STRING`   [1]

NB: check for any attribute repeated in a child class. If present score 0.

MAX 8

(c) *Encapsulation*
Combining together of an object's properties/data and the methods   [1]
Restricts the programmer's access to the object's data // provides for 'data hiding'   [1]
Data values can only be read/written using the methods of the class   [1]
MAX 2

[Total: 13]

5 (a) Boolean [1]
Flags when the input <u>name</u> is found [1]

```
//Serial search algorithm
INPUT SearchName                                                    [1]
IsFound ← FALSE
Index ← 1

REPEAT
    IF  Customer[Index] = SearchName    Allow '(' and ')'          [1]
        THEN
            IsFound ← TRUE
            OUTPUT "FOUND" at position Index
        ELSE
            Index ← Index + 1                                      [1]
    ENDIF
UNTIL (IsFound = TRUE) OR Index=101 / >100                          [1]

IF IsFound = FALSE // Index = 101/>100                              [1]

    THEN
        OUTPUT "Customer name was NOT FOUND"
ENDIF
```

(b) 50 // half the number of customers [1]

(c) (i) Items in order [1]

(ii) The function makes a call to itself (in two places) [1]

(iii) BinarySearch (Surname, "Hwang", 1, 11)

Low	High	Middle	RETURNS ...
1	11	6	
(1)	5	3	
4	(5)	4	4

[4]

[Total: 14]

6 **(a)**

−126	1	0	0	0	0	0	1	0
−5	1	1	1	1	1	0	1	1	+
	0	1	1	1	1	1	0	1

    1   1

Mark as follows ...

−126 binary [1]

−5 binary [1]

Correct final pattern (f/t from their −126 and −5) [1]

Answer is incorrect since <u>outside range</u> possible represented with single byte // answer overflows// final bit pattern is NOT −131 [1]

(b) **(i)** Mantissa: +13/16 [1]
Exponent: +3 [1]
Number: +13/16 × 2^{+3} // evidence of shifting the mantissa three places [1]
6.5 [1]
MAX 3

(ii) The mantissa starts with the digits 01
// the first two bits in the mantissa are different [1]

(iii) More bits used for the mantissa will result in greater <u>accuracy</u>/precision [1]
More bits used for the exponent will result in larger <u>range</u> of numbers [1]

[Total: 10]

7 Possible answers include:

(a) Encryption of email traffic [1]
Email data if intercepted cannot be read [1]

Encryption of passwords // logging-in to "something" [1]
Designed to prevent unauthorised access [1]

Hospital patient records [1]
Will safeguard the privacy/confidentially of data [1]

(b) *Plain text*
The (message) text/data/ before encryption // unaltered text/original text [1]
Cipher text
The (message) text <u>after encryption</u> [1]

(c) *Symmetric encryption*
The plain text /data is encrypted using ... [1]
An <u>encryption</u> key [1]
Decryption is done using the <u>same/ or by implication</u> key …… [1]
and a matching decryption <u>algorithm/process</u> [1]
MAX 3

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

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

[Total: 11]