CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Level

MARK SCHEME for the May/June 2015 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 2015 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

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

(ii) ClassName and ClassLevel and ClassLeader is repeated for each MemberNo **[1]**

(b) (i)

MemberNo	MemberType	Trainer
510	SF	SAF
808	SS	OLO
756	J	DAV

[1]

(ii)

MemberNo	ClassName	ClassLevel	Trainer
510	Yoga B	B	OLO
808	Swimathon	A	ROG
756	Circuits	I	VAR

Any three correct rows from the original table

All 3 correct – 2 marks

2 correct – 1 mark

1 correct only scores 0

[2]

(iii) 8 **[1]**

(iv) One to many // 1-to-M **[1]**

(v) Primary key / MemberNo in the MEMBER table (1)
Links to foreign key in the MEMBERCLASSES table (1) **[2]**

(c) (i) MemberNo + ClassName **[1]**

(ii) There are a non-key attribute(s) dependant on only part of the primary key // there are partial dependencies (1)
ClassLevel/ClassLeader is dependent on ClassName (1) **[2]**

(iii) MEMBERCLASSES(MemberNo, ClassName)

CLASS(ClassName, ClassLevel, ClassLeader)

mark as follows:
MEMBERCLASSES has only MemberNo, ClassName (1)
(ignore primary key for MEMBERCLASSES)
new table CLASS (1)
CLASS has 3 attributes ClassName, ClassLevel, ClassLeader (1)
ClassName as primary key (1)

[Max 3]

(d) (i) There are non-key attributes which are dependent (may be stated as part of the attribute description) // transitive dependencies (1)
MemberTypeFee is dependent on MemberType (1)
There is no need to store the MemberTypeFee in the MEMBER table (1)

[Max 2]

(ii) MEMBER(MemberNo, MemberType, Trainer) (1)
FEES(MemberType, MemberTypeFee) (1) **[2]**

[Total: 19]

2 (a) Alternatives // OR **[1]**

(b) Rule 2 (1)
The rule is defined in terms of itself / calls itself (1) **[2]**

(c) (i) Valid (1)
All five rules are used once only (1) **[2]**

(ii) Invalid (1)
5, 3 // 3, 5 (only) (1) **[2]**

(iii) Valid (1)

Rule 1 – three times

Rule 2 – three times

Rule 3 – once

Rule 4 – once

Rule 5 – at least once (1) **[2]**

(iv)

5	`<Packet> ::= <Start><String><Stop> | <Start><HashString><Stop>`
6	`<Hash> ::= #`
7	`<HashString> ::= <Hash>|<Hash><HashString>`

Mark as follows:

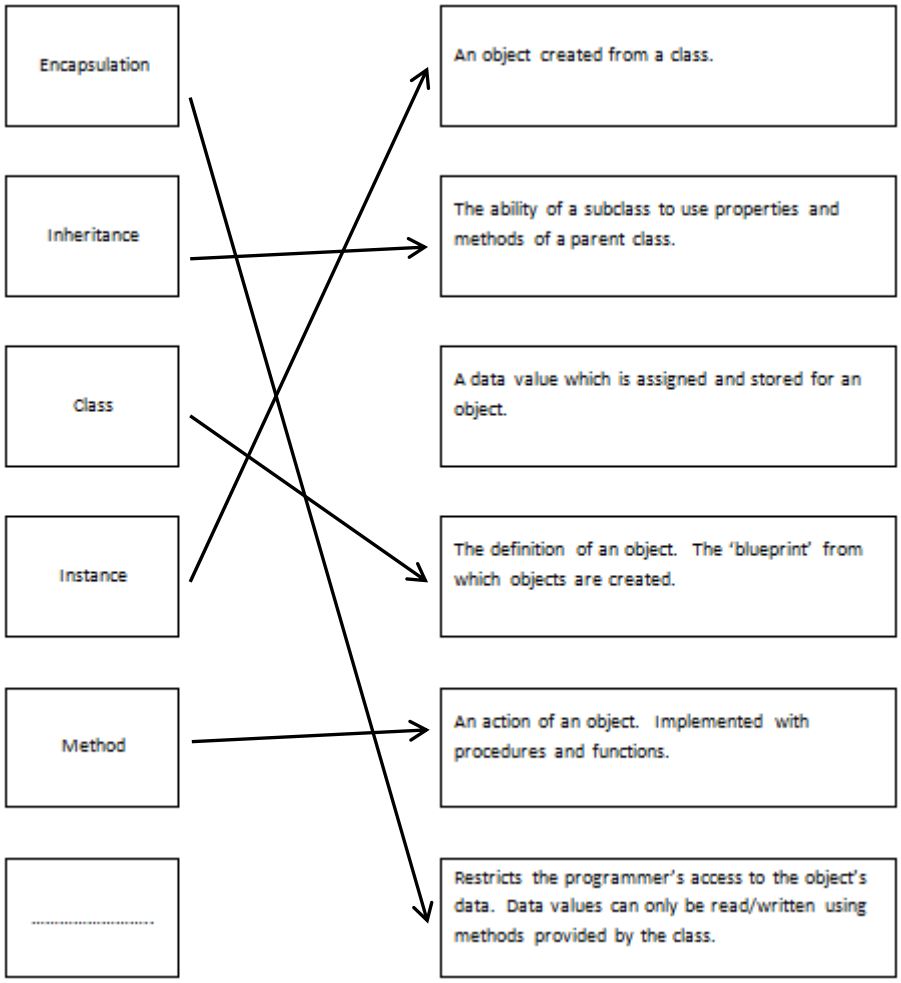
```
<Hash> ::= #                                                    (1)

<HashString> ::= <Hash>|<HashString><Hash>                      (1)

<Packet> ::= <Start><String><Stop> |
                  <Start><HashString><Stop>                     (1)  [3]
```

[Total: 12]

3 (a)



Each term matched to its correct description × 5 (5)
Missing term – Property / **A.** Attribute (1) **[6]**

(b) The class diagram includes:

`PERMANENT + CONTRACT` subclasses (1)

`PROGRAMMER + WEBDESIGNER` subclasses of `PERMANENT` and no other subclasses (1)

Note: for the two above marks – correct class names only

Recognised notation for inheritance (from `CONTRACT` and `PERMANENT` only) (1)

Note: property/group of properties cannot be repeated in any subclasses

`EMPLOYEE` class	`DateFirstJoined : DATE/STRING`	(1)
`PERMANENT` class	`SalaryGrade : STRING/INTEGER/CHAR`	
	`CourseList : STRING`	(1)
`WEBDESIGNER` class	`MarkupLanguage : STRING`	(1)
`PROGRAMMER` class	`Language : STRING`	(1)
`CONTRACT` class	`AgencyName : STRING`	
	`HourlyRate : REAL/CURRENCY`	(1)
	`JobRole : STRING`	

[8]

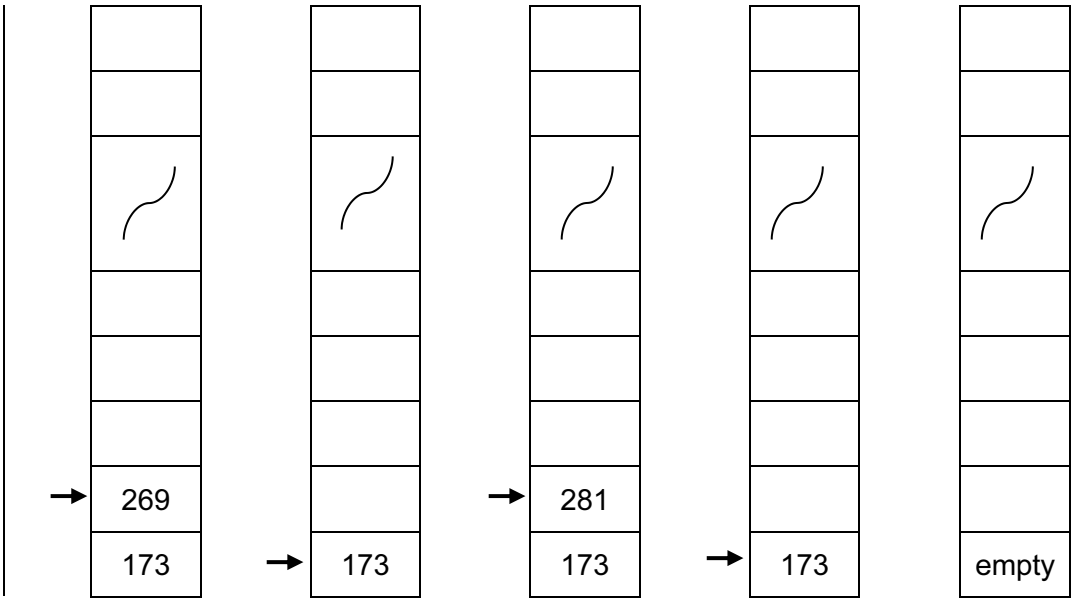
Note: accept any reasonable variations for the property identifiers

[Total: 14]

4 **(a)** Last item in is the first item out // First item in is the last item out

R. LIFO

(b) **(i)**



Mark as follows:
1 mark per correct change × 5
Note: Final 'empty' contents is conditional on one value only in the previous stack
1 mark for consistent `TOS` pointing to 'their' stack contents (allow omitted from final stack)

[Max 5]

(ii)

```
PROCEDURE PushAddress
  IF TOS = 100
    THEN
      OUTPUT "Stack/memory is FULL"                    (1)
    ELSE
      INPUT NewAddress                                 (1)
      TOS ← TOS + 1                                    (1)
      Stack[TOS] ← NewAddress                          (1)
  ENDIF
ENDPROCEDURE                                           [4]
```

(c)

```
PROCEDURE PopAddress
  IF TOS = 0 // TOS < 1                                          (1)
    THEN
          OUTPUT "There are no current procedure calls"
    ELSE
      OUTPUT "Address " Stack[TOS]
      TOS ← TOS - 1                                              (1)
  ENDIF
ENDPROCEDURE
```

[2]

[Total: 12]

5 **(a)** **(i)** 111
6F (…)

(ii) -29 (1)
E3 (1) **[2]**

(b) -128 **[1]**

(c) Fewer digits used to represent any number // long string difficult to interpret (1)

Less likely to make a mistake <u>when copying/converting</u> a digit string (1)

Easy to convert from binary/denary to hex (vice versa) (than binary to denary) (1)

[Max 1]

(d)

124	0	1	1	1	1	1	0	0
7	0	0	0	0	0	1	1	1
	1	0	0	0	0	0	1	1

+

124 and 7 correct pattern (1)
Correct addition // ft (1)

Overflow has occurred // the answer should be 131/their ‘ft’ value is outside the possible range // the final pattern is a negative value (1) **[3]**

(e) **(i)** `9837` **[1]**

(Exact – with no additional characters)

(ii) `1101` is not a valid BCD digit string // `1101` represents 13 **[1]**

[Total: 11]

6 **(a)** Systems flowchart

(b)
- 1 – Source code in language XYZ
- 2 – Text editor
- 3 – Source code in assembly language
- 4 – Error report
- 5 – Program library code
- 6 – Linker
- 7 – Loader **[7]**

(c) Benefit:

Interpreter makes for easier debugging // better diagnostics (1)

Testing can be done without all the code being written (1)

(Max 1)

Drawback:

Interpreter needed/source code always present every time program execution attempted (1)

Execution will be slower (1)

(Max 1)

[2]

[Total: 10]

7 (a) • Twisted pair

Two copper wires insulated from each other and twisted together

• Coaxial cable

Central copper wire shielded from outer metal mesh

• Optical fibre

Glass strands to send light/optical signals

• Electro-magnetic / long wavelength communication

radio waves /microwave // satellite communication // mast relays
'wireless' but not in the context of WiFi

2 × (Name – 1 mark + Description – 1 mark)

[Max 4]

(b) Mark as follows:

End terminator for the LAN cable X 2	(1)
C4 computer + Laser printer connected to the cable	(1)
File server labelled Server Y connected to the cable	(1)
Firewall / Proxy server + Indication of a connection to the WAN/other shop	(1)
Router at Shop A / Shop B / Shop C's LAN to connect to the WAN/other shop	(1)
Modem + Indication of a connection to the WAN/other shop	(1)

[Max 4]

(c) (i) Web server **[1]**

(ii) (Web) browser **[1]**

(iii) Information being communicated may be sensitive/confidential/secure // needs protection from being seen by unauthorised people // content only available within the organisation
Good control of who can access/update the content
Information on system will be relevant/accurate/reliable
Should reduce paperwork
Presents information using a familiar interface/browser software // Provides web server content to client computers
Intranet uses the same communication protocols as the Internet

[Max 2]

[Total: 12]