CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Level

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

CAMBRIDGE
International Examinations

1 (a) `MEMBER (MemberID,` – any two sensible attributes e.g. name, DOB [1]
`COURSE` – `CourseID, Instrument, StartDate, Duration, CourseTutor` [1]

(b) `COURSE-MEMBER` (`CourseID, MemberID`)
Correct two attributes 1
Correct primary key 1
If `CourseName` used – refer back to part (a) for possible FT [2]

(c) The table has a repeated attribute //
There are several Instruments for the same `TutorID` [1]

(d) (i) `TUTOR` – has primary key `TutorID` [1]
`TUTOR-INSTRUMENT` – has primary key `TutorID + Instrument` [1]

(ii) Many-to-one // M:1 // ∞ : 1 [1]

(iii)
```
SELECT TutorID
FROM TUTOR-INSTRUMENT
WHERE Instrument = 'saxophone'
```
can be reverse for 1 mark [1] [1] [1]

'saxophone' – allow mixed case / spelling must be correct
Quotes must be present

(iv)

Creates a new record in the `TUTOR-INSTRUMENT` table	FALSE
Amends an existing record in the `TUTOR-INSTRUMENT` table	TRUE
Assigns the `INSTRUMENT` attribute the value 'guitar'	FALSE
Assigns the `INSTRUMENT` attribute the value 'Piano'	FALSE
Make a change to all the existing records for all tutors	FALSE
Changes one record in the `TUTOR` table	FALSE

[1] [1] [1]

(e) (i)
```
INSERT INTO TUTOR-INSTRUMENT
(TutorID, Instrument)
VALUES (57, 'flute')
```
1
1 [2]

(ii) Attempt to add a record in `TUTOR-INSTRUMENT` table 1
But, no corresponding `TutorID` in the `TUTOR` table … 1
Or:
// Delete a record in the `TUTOR` table 1
and, matching records in `TUTOR-INSTRUMENT` table remain 1
Or:
// Allow use of the term 'update' if mentions a change to `TutorID`/foreign key attribute 1 [max 2]

2 (a) (i) (0) 101 1010 [1]

(ii) 5A [1]

(iii) 8 [1]

(b)

Register transfer notation	Description	
MAR ← [PC]	The contents of the Program Counter are copied to the Memory Address Register.	[1]
PC ← [PC] + 1	**(The Contents of) the Program Counter are incremented**	[1]
MDR ← [[MAR]]	The Memory Address Register contains an address. Copy the contents of this address to the Memory Data Register.	[1]
CIR ← [MDR]	**Copy the contents/data in/instruction in the Memory Data Register to the Current Instruction Register**	[1]

(c) (i) Sends control signals (A. commands) from/to the processor and various devices // individual lines are each dedicated to a particular signal // uses timing signals to coordinate various actions 1

Examples:
Timing signal // reset // memory write/read // I/O operation completed // interrupt 1 [2]

(ii) Data bus 1
Connects/used to transport a data value between main memory and the processor // data bus is bi-directional 1

Address bus 1
Carries the memory address about to be accessed // it connects the Memory Address Register to main memory // Address bus is uni-directional from the processor 1 [4]

3 (a)

	Register	
Instruction	**ACC**	**Index Register (IX)**
LIX 902		**2** [1]
LDD 901	**917**	[1]
LDI 901	**13**	[1]
LDX 901	**25**	[1]

(b) Use the text editor to create the assembly language program PROG.ASM
REPEAT

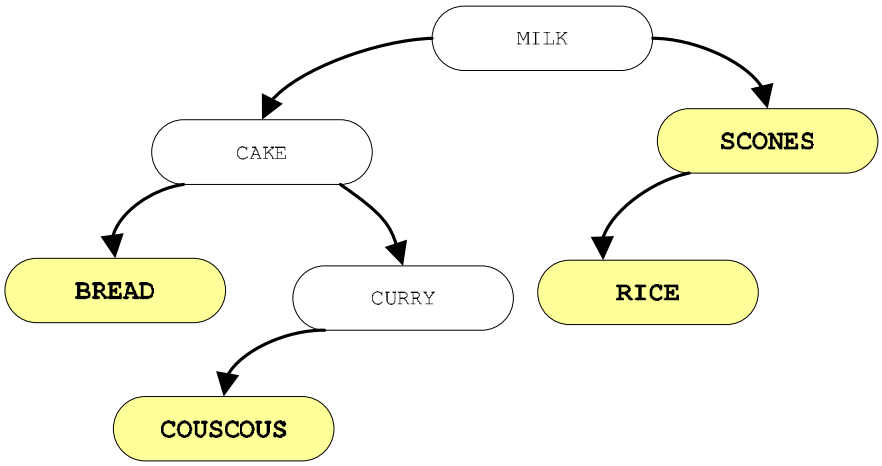
Can be reverse for 1 mark

PROG.ASM is input to the assembler software [1]
IF errors reported [1]
THEN
Amend PROG.ASM using the text editor [1]
ENDIF

UNTIL No errors reported [1]

Produce the PROG.EXE executable file [1]
Execute PROG.EXE

4 (a) (i)



[4]

(ii) Root labelled [1]
Right subtree labelled // FT for their tree [1]

(iii) 3 // FT for their tree [1]

(b)

RootPtr	0				
0	1		MELON		2
1	3		BEETROOT		
2	4		TURNIP		
3			APPLE		
4			PARSNIP		5
5	6		SWEDE		
6			QUINCE		

Mark as follows:

Root showing 0	1	
Data values in correct positions	1	
Correct left pointers	1	
Correct right pointers	1	[4]

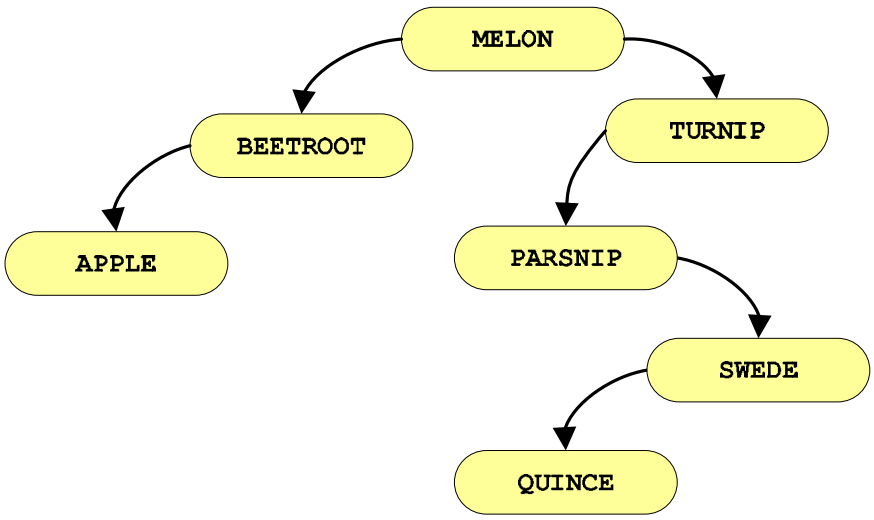
Ignore values showing any unassigned pointers

(c) (i)

```
//binary tree search
INPUT SearchFood
IsFound ← FALSE                                          [1]
Current ← RootPtr                                        [1]
   REPEAT
      IF Food[Current] = SearchFood                      [1]
         THEN
            //found
            OUTPUT "Found"
            ISFound ← TRUE                               [1]
         ELSE
            IF SearchFood < Food[Current]
               THEN
                  // move left
                  Current ← LeftPtr[Current]
               ELSE
                  Current ← RightPtr[Current]            [1]
            ENDIF
         ENDIF
      UNTIL IsFound = TRUE OR
                      Current = 0/Null/-1/Unassigned     [1]

      IF IsFound = FALSE
         THEN
            OUTPUT SearchFood, "Not Found"
      ENDIF
```

(d) (i)



Mark as follows:

Left subtree	1	
Right subtree	1	[2]

(ii) Correct conclusion is made from 'their' tree [1]

5 (a) (i) Building a model of the system/real-life situation // Models the behaviour of the system 1
R. 'imitates'
The simulation records the result of some changing parameters/conditions/ circumstances 1 [2]

(ii) A computer program can be written to build the model 1
Computer system can process results very quickly/can change the time frame // can process large volumes of data 1
Computer avoids any health and safety issues 1 [max 1]

(b) (Current) traffic flows between Town A and Town B 1
Usage of the road(s) by pedestrians 1
Number of houses to be built 1
Number of vehicles per house 1 [max 2]

(c) (i) BEWARE : NOT traffic lane changes (given in the rubric)

Width of the road 1
Time interval between traffic signal changes 1
Signal changes activated by sensors 1
Rate of arrival of vehicles // vehicle speed 1
Variation in the type of vehicle using the junction 1
Left/right filter used 1
Time of day 1
Weather conditions 1
Outside influences e.g. zebra crossings 1 [max 2]

(ii) Queue length 1
Waiting time 1
Vehicle throughput to/from Housing development to Town A/Town B 1 [max 1]

6 (a) (i) 4 [1]

(ii) 3 [1]

(iii) ERROR [1]

(iv) 0 [1]

(v) ERROR [1]

(b) (i) A – Parameter identifiers labelled 1
B – (RETURNS) INTEGER 1 [2]

(ii) `RejectCount ← CountBig` `(Rejects, 83, 'Y')`
1 mark 1 mark
or
`RejectCount ← CountBig` `(Rejects, 82, 'N')`
1 mark 1 mark [2]

(c)
```
FUNCTION StringFound (ThisArray : ARRAY OF STRING ,
                      ThisValue : STRING) : BOOLEAN
```

Mark as follows:

`FUNCTION StringFound( .....` 1
`ThisArray : ARRAY OF STRING` 1
`ThisValue : STRING` 1
`RETURNS BOOLEAN` // `: BOOLEAN` 1 [4]

7 (a) A number of connected (allow: 'linked') computers/devices … 1
Sited in a small geographical area // room/building/site 1 [2]

(b) (i) Network Interface card // NIC [1]

(ii) (Use of user IDs and) passwords // use of biometrics/by example 1
Firewall // Proxy server 1 [2]

(c) Storage/Management of all user data/files 1
Management of centrally stored software 1

Granting of access rights/permissions to various users 1
Note: refuse just mention of 'security'

Sharing of peripherals // Control of all output to a printer device 1
Management of user accounts/Log-ons 1
Recording/Monitoring the use of the network // Accounting 1 [max 3]

(d) (File) server // Network attached storage (NAS) [1]