Cambridge International Examinations
Cambridge International Advanced Level

COMPUTING **9691/33**

Paper 3 Written Paper **May/June 2016**

MARK SCHEME

Maximum Mark: 90

Published

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.

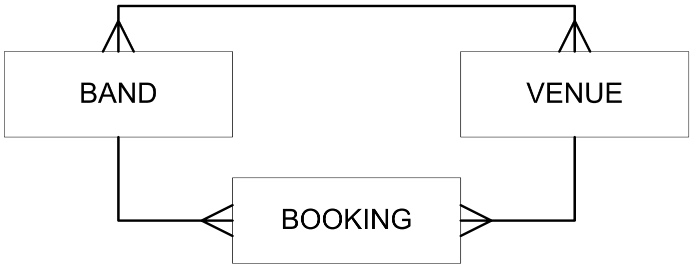
Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2016 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

This document consists of **7** printed pages.

1 (a) (i) `BOOKING(BandName, PerformanceDate, StartTime, VenueName)` **[2]**

(ii) **[2]**



2 × correct relationships

(b) (i) *Not in 2 NF ….*
`BOOKING` **[1]**

`NoOfMusicians` is only dependant on knowing part of the PK (i.e. the `BandName`) there is a non-key attribute which is dependent on only one of the PK attributes // **[1]**
there are partial dependancies **[1]**

`BOOKING(BandName, PerformanceDate, StartTime,VenueName)` **[1]**
All correct ..

(ii) *Not in 3NF ….* **Max [5]**
`BAND`

Since there are dependent non-key attributes
`ManagerName` and `ManagerPhoneNumber` are both dependent on `ManagerID`

Re-design …
`BAND(BandName, NumberOfMusicians, Genre, ManagerID)`

New table `MANAGER` …

`MANAGER(ManagerID, ManagerName, ManagerPhoneNumber)`

(c)

```
SELECT BandName, PerformanceDate                                  [1]
FROM BOOKING                                                      [1]
WHERE VenueName = 'Dominion Theatre';                             [1]
```

(d)

```
UPDATE BOOKING                                                    [1]
SET StartTime = '21:00'                                           [1]
WHERE BandName = 'RUS' AND PerformanceDate = #06/08/2016#         [1]
```

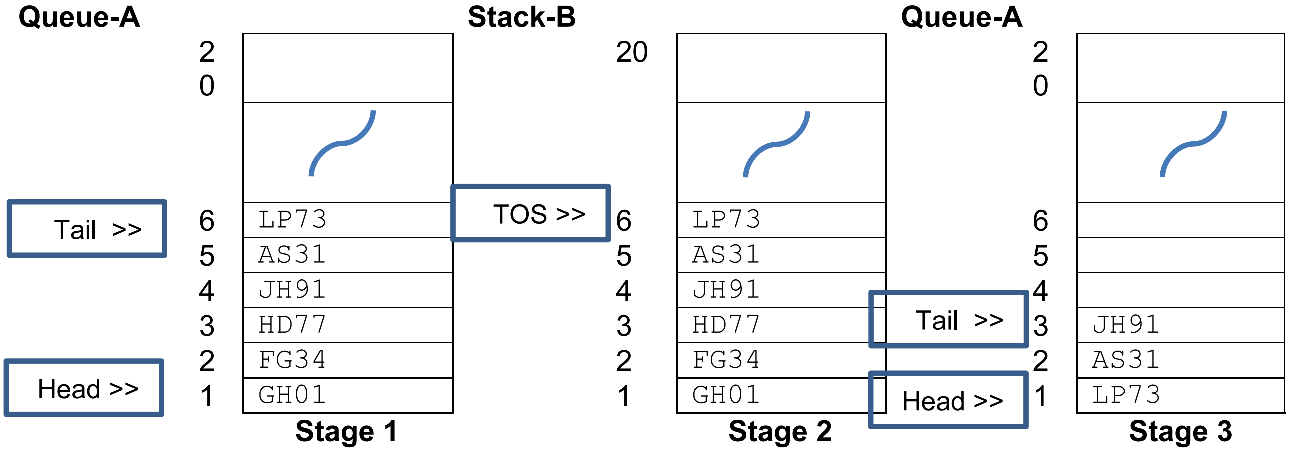
2 **(a)** Last item added will be the first item to leave // First item added will be the last item to leave **[1]**

(b) First item added will be the first item to leave **[1]**

(c) **(i)** **Max [4]**

```
PROCEDURE InitialiseQueue1
    DECLARE Item : INTEGER                    (1)
    Head ← 0
    Tail ← 0                                  (1)

    FOR Item ← 1 TO 20                        (1)
        Queue[Item] ← ""                      (1)
    ENDFOR
ENDPROCEDURE
```

(ii) `Item – INTEGER` – Loop counter // index for the Queue array **[1]**



(d) **(i)** Data items (1) **[3]**
`Head` (1)
`Tail` (1)

(ii) Data items (1) **[2]**
`TOS` (1)

(iii) Data items (1) **[2]**
`Head + Tail` (1)

(iv) Original items are reversed on the queue **[1]**

(e) (i) **[4]**

```
PROCEDURE Push(NewItem : STRING)
   IF TOS = 20
      THEN
         OUTPUT "STACK is FULL"
      ELSE
         TOS ← TOS + 1
         Stack[TOS] ← NewItem
   ENDIF
ENDPROCEDURE
```

(ii) **Max [3]**

`MyStack.InitialiseStack`	(1)
`MyStack.Push("JH45")`	(1)
`MyStack.Push("HH90")`	(1)
`DeletedItem ← MyStack.Pop()`	(1)

3 (a) (i) Loads the number to …. the ACC **[2]**
129

(ii) 0581 **[1]**

(iii) Fewer digits to write // less chance of an error in writing the code // easy conversion to/from a binary value **[1]**

(iv) **[2]**

0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0

1 mark for each byte

(v) True **[2]**
`OUTCH`// `IN` // `END` // or using a good explanation (only) of either

(b) **[6]**

ACC	**Address**					**OUTPUT**
	150	**151**	**200**	**201**	**202**	
	200	0				
76						L
	201	1	76			
1						
79						O
				79		
	202	2				
2						
87						W
	203	3			87	
3						
With nothing after the '3'						

4 (a) **Max [8]**

The program instructions are stored in a continuous block of main memory.
The Program Counter stores the address of the next instruction to be fetched.
Stage 1. The contents of the Program Counter are copied to the MAR.
Stage 2. The contents of the PC are then incremented.
Stage 3. The value in the Memory Address Register is loaded to the address bus. The data value found at this address is loaded on to the data bus and copied to the MDR.
Stage 4. The contents of the Memory Data Register are copied to the CIR
and its contents processed to separate the (op code and the operand).
The instruction can now be executed.
Note: final two can be inter-changed and are 1 mark only.

(b) (i) **Max [2]**

Case 2

The address in CIR must be loaded to the MAR / address bus
The data value must be retrieved from this address / address 139

(ii) **Max [2]**

Case 1

The operand is a register // the register is part of the CPU // it is using only the Accumulator
There is no memory access

5 (a) (i) **[2]**

Variable	**Data Type**	**Description**
`i`	`INTEGER`	Loop counter
`IsFound`	`BOOLEAN`	Flags to `TRUE` when item found

1 mark per (Identifier name + Data type + Description)

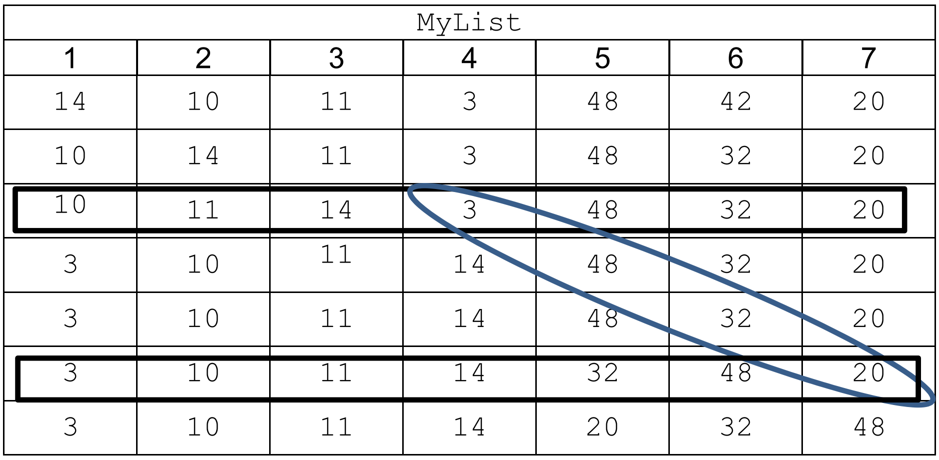
(ii) **Max [6]**

```
INPUT SearchItem

IsFound ← FALSE
i ← 1
REPEAT
   IF MyList[i] = SearchItem
      THEN
       IsFound ← TRUE and initialised earlier
       OUTPUT "FOUND at position", I
      ELSE
       i ← i + 1 and initialised earlier
UNTIL (IsFound = TRUE) OR i = 7

IF IsFound = FALSE
    THEN
     OUTPUT "Item was NOT FOUND"

ENDIF
```

(iii) 125 comparisons **[1]**

(b) The items in the `MyList` array are not in order **[1]**

(c) **[3]**



MyList						
1	2	3	4	5	6	7
14	10	11	3	48	42	20
10	14	11	3	48	32	20
10	11	14	3	48	32	20
3	10	11	14	48	32	20
3	10	11	14	48	32	20
3	10	11	14	32	48	20
3	10	11	14	20	32	48

Mark as follows:
1 mark for correctly circled data items
Highlighted row × 2 marks

6 **(a)** **(i)** Any **five** from: **Max [5]**

Running – The process currently has use of the processor

Ready – the process is able to use the processor but the processor is currently allocated to another process

Suspended/Blocked – the job is unable to use the processor

When a process is suspended the processor will have a strategy/ by example
For deciding which process gets next use of the processor
Any example of a process changing states
A second example of a process changing states
The next process to get the processor is at the head of the Ready queue

(ii) Interrupt signal is used to trigger a change of state …………… **[2]**
for the process in the running state

(b) Any **two** from: **Max [2]**
Maintain a file directory
Detail – file name, file size, date saved (2 or more items for the mark)

Manage the unallocated storage units // Use of a FAT

(c) Any **one** from: **Max [1]**
The input/output of data // the peripherals
Provision of a user interface
Main memory management