Cambridge International A Level

Cambridge International Examinations
Cambridge International Advanced Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

COMPUTING **9691/33**

Paper 3 **May/June 2016**

2 hours

Candidates answer on the Question Paper.

No additional materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name on all the work you hand in.
Write in dark blue or black pen.
You may use an HB pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.
No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

1 A band consists of a number of musicians who give performances at different venues.

A band never has more than one booking on a particular date.

Each band has a manager who arranges bookings for the band.

A manager has a unique ID and may manage several bands.

The managers want to store data about bands, managers and bookings in a relational database.

(a) Consider Design 1:

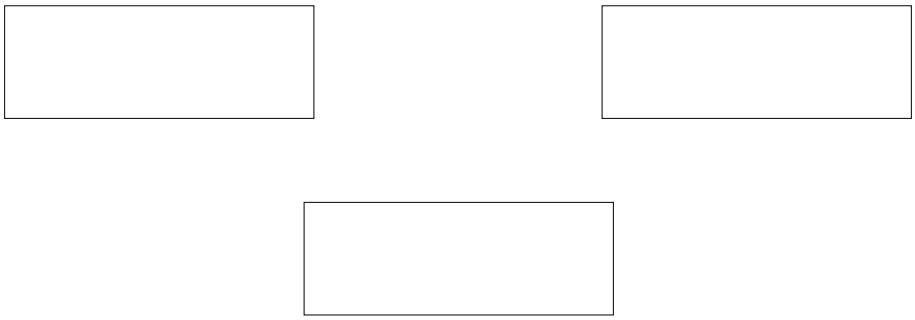
```
BAND(BandName, NumberOfMusicians, Genre, ManagerID)

VENUE(VenueName, Capacity)

BOOKING(BandName, PerformanceDate, StartTime, VenueName)
```

(i) Circle the two foreign keys in Design 1. [2]

(ii) There are relationships between the entities `BAND`, `VENUE` and `BOOKING`.
Complete the entity-relationship (E-R) diagram to show the entity names and **two** relationships.



[2]

(b) The managers want to store more data.

Consider Design 2:

```
BAND(BandName, NumberOfMusicians, Genre, ManagerID,
                                  ManagerName, ManagerPhoneNumber)

VENUE(VenueName, Capacity)

BOOKING(BandName, PerformanceDate, StartTime,
                                  VenueName, NumberOfMusicians)
```

(i) Name the table that is not in Second Normal Form (2NF) and explain why.

Table .....

Explanation .....

.....

.....

Re-design this table to put it into 2NF.

.....[4]

(ii) Name the table which is not in Third Normal Form (3NF) and explain why.

Table .....

Explanation .....

.....

.....

Re-design this table and add a new table to put it into 3NF.

.....

.....

.....

.....[5]

(c) The managers need to display the band names and performance dates for the bands that have played at the Dominion Theatre.

Write a Data Manipulation Language (DML) query to do this.

.....

.....

.....[3]

(d) A booking already exists in the database for the band RUS on the 6th August 2016.
A manager needs to change the start time to 21:00.

Write a DML command to update this record.

.....

.....

.....[3]

2 **(a)** State how a stack operates.

.....

.....[1]

(b) State how a queue operates.

.....

.....[1]

(c) A queue (Queue-A) and a stack (Stack-B) are to be implemented as follows:

- Queue-A is controlled by two pointers, `Head` and `Tail`.
- Stack-B has a single pointer, `TOS`.
- Both Queue-A and Stack-B are initially empty.
- Queue-A and Stack-B are to be implemented using the data structures and variables in the identifier table.

Identifier	Data type	Description
`Stack`	`ARRAY[1 : 20] OF STRING`	Data values in Stack-B
`TOS`	`INTEGER`	Index position of the `Stack` array for the item currently at the top of the stack
`Queue`	`ARRAY[1 : 20] OF STRING`	Data values in Queue-A
`Head`	`INTEGER`	Index position of the `Queue` array for the item currently at the head position
`Tail`	`INTEGER`	Index position of the `Queue` array for the item currently at the tail position

(i) Write pseudocode for a procedure `InitialiseQueue` to initialise Queue-A.

.....

.....

.....

.....

.....

.....

.....

.....[4]

(ii) Show the additional variable that your pseudocode uses.

Identifier	Data type	Description

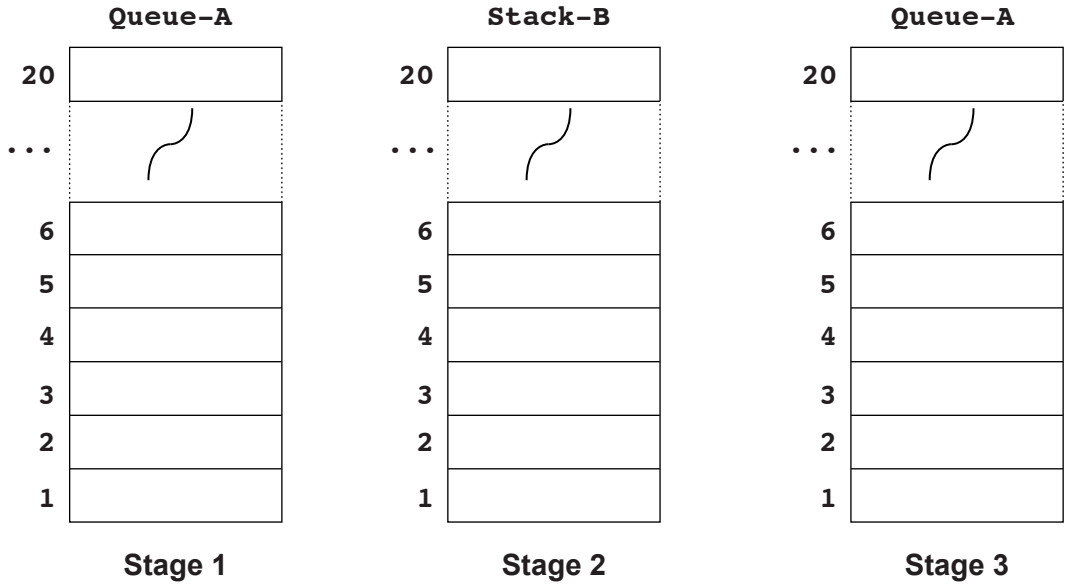
[1]

(d) Queue-A and Stack-B are used to store data values.

(i) The following sequence of six data values are stored into Queue-A:

```
GH01, FG34, HD77, JH91, AS31, LP73
```

Show on the diagram below the contents of Queue-A and its pointers at this stage (Stage 1). [3]



(ii) All the contents of Queue-A are then removed and placed on Stack-B.

Show on the diagram above, the contents of Stack-B and its pointer at this stage (Stage 2). [2]

(iii) **Three** items are removed from Stack-B and stored in Queue-A.

Show on the diagram above the final contents of Queue-A and its pointers at this stage (Stage 3). [2]

(iv) All the remaining items on Stack-B are removed and stored in Queue-A.

Comment on the final contents of Queue-A.

.....

.....[1]

(e) It is suggested that the stack could be implemented using object-oriented programming.

A programmer designs the following pseudocode for the class definition.

(i) Complete the pseudocode.

```
CLASS StackClass
   PRIVATE
      TOS         : INTEGER
      Stack       : ARRAY[1 : 20] OF STRING
   PUBLIC
      PROCEDURE InitialiseStack
         <statements>
      ENDPROCEDURE
      PROCEDURE Push(NewItem : STRING)
         IF ..........
            THEN
               OUTPUT ".......... "
            ELSE
               TOS ← ..........
               .......... ← NewItem
         ENDIF
      ENDPROCEDURE
      FUNCTION Pop
        <statements>
      ENDFUNCTION
ENDCLASS
```

[4]

(ii) The main program uses a variable `MyStack` as an instance of `StackClass`.

Write pseudocode statements that would carry out the following sequence:

- initialise `MyStack`
- add (push) to the stack `"JH45"` followed by `"HH90"`
- remove an item (pop) from `MyStack` and assign the value to variable `DeletedItem`

.....

.....

.....

.....[3]

3 The table below gives a subset of the assembly language instruction set for a processor. The processor has a single general purpose register, the Accumulator (ACC).

Instruction			
Opcode	Operand	Opcode (binary)	Explanation
LDD	<address>	0000 0100	Direct addressing. Load the contents of the given address to ACC
LDV	<number>	0000 0101	Load the given number to ACC
STO	<address>	0000 1000	Store the contents of ACC at the given address
STI	<address>	0000 1001	Indirect addressing. At the given address is the address to be used. Store the contents of ACC at this address
LDI	<address>	0000 0110	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
INC	ACC	0000 0011	Add 1 to the contents of ACC
INC	<address>	0000 0100	Increment the contents of the address
OUTCH		1000 0001	Output the character corresponding to the ASCII character code in ACC
IN		1001 0000	Input a denary number from the keyboard and store in ACC
JMP	<address>	1100 1000	Jump to the given address
CMP	#<number>	1100 1001	Compare the contents of ACC with the given number
JPE	<address>	1110 0111	If the result of the previous compare instruction was a match, jump to the given address
END		1111 1111	Returns control to the operating system

(a) The given table of instructions shows the binary representation used for the opcode of each instruction.

All instructions in machine code are stored as a 16-bit pattern:

- the first 8 bits are the opcode
- the second 8 bits are the operand

Consider the instruction:

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

(i) Describe what this instruction does.

.....

.....[2]

(ii) Write this instruction in hexadecimal.

.....[1]

(iii) Give a reason why programmers prefer to write machine code instructions in hexadecimal.

.....

.....[1]

(iv) Write the binary pattern for the instruction:

Store the contents of the Accumulator at address 90 (denary).

[2]

(v) A programmer makes the statement:

"For this instruction set, some of the instructions do not require an operand"

Circle whether this statement is true or false.

True / False

Explain your choice with reference to the instruction set given.

.....

.....

.....

.....[2]

The instruction set is repeated here for ease of reference.

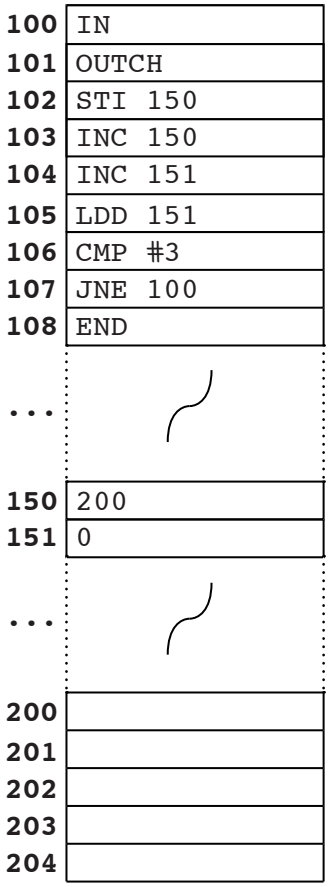
Instruction		Opcode (binary)	Explanation
Opcode	Operand		
LDD	<address>	0000 0100	Direct addressing. Load the contents of the given address to ACC
LDV	<number>	0000 0101	Load the given number to ACC
STO	<address>	0000 1000	Store the contents of ACC at the given address
STI	<address>	0000 1001	Indirect addressing. At the given address is the address to be used. Store the contents of ACC at this address
LDI	<address>	0000 0110	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
INC	ACC	0000 0011	Add 1 to the contents of ACC
INC	<address>	0000 0100	Increment the contents of the address
OUTCH		1000 0001	Output the character corresponding to the ASCII character code in ACC
IN		1001 0000	Input a denary number from the keyboard and store in ACC
JMP	<address>	1100 1000	Jump to the given address
CMP	#<number>	1100 1001	Compare the contents of ACC with the given number
JPE	<address>	1110 0111	If the result of the previous compare instruction was a match, jump to the given address
END		1111 1111	Returns control to the operating system

ASCII code table (part)

Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

(b) The user runs the following program and inputs the numbers 76, followed by 79 and 87.

Trace the execution of the program by completing the trace table.



ACC	Address					OUTPUT
	150	151	200	201	202	
	200	0				

[6]

4 **(a)** The following text includes a description of four stages of the fetch-execute cycle.

Use the terms below to complete the text:

Memory Data Register (MDR)	address
Memory Address Register (MAR)	data bus
main memory	control bus
Program Counter (PC)	address bus
operand	op code
Current Instruction Register (CIR)	control unit

The program instructions are stored in a continuous block of ..... .

The Program Counter stores the ..... of the next instruction to be fetched.

Stage 1 The contents of the Program Counter are copied to the ..... .

Stage 2 The contents of the ..... are then incremented.

Stage 3 The value in the Memory Address Register is loaded to the

..... .

The data value found at this address is loaded on to the

..... and copied to the ..... .

Stage 4 The contents of the Memory Data Register are copied to the

..... and its contents processed to separate the

..... and the ..... .

The instruction can now be decoded and executed.

[8]

(b) Consider two assembly language instructions that were given in **Question 3**.

Instruction		Explanation
Opcode	Operand	
`INC`	`ACC`	Add 1 to the contents of ACC
`STO`	`<address>`	Store the contents of ACC at the given address

Consider the following two cases:

Case 1
On completion of the fetch stage, the instruction is decoded. The instruction is then executed without further use of the address bus.

Case 2
On completion of the fetch stage, the instruction is decoded. Once decoded, the address bus will be used again before the execution of the instruction can be completed.

For each instruction below, circle either Case 1 or Case 2. Give a reason for your choice.

(i) `STO 139`

Case 1 / Case 2

.....

.....

.....[2]

(ii) `INC ACC`

Case 1 / Case 2

.....

.....

.....[2]

5 A program will use the following array `MyList` with the integers shown:

`MyList`

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

(a) A sequential search algorithm is to be coded to find the number input by a user.

(i) Complete the identifier table:

Variable	Data type	Description
`MyList`	`ARRAY[1 : 7] OF INTEGER`	The given list of numbers
`SearchItem`	`INTEGER`	The number input

[2]

(ii) Write pseudocode for this search.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

(iii) For an array with 250 items, state how many comparisons on average would be made to locate a particular item.

.....[1]

(b) An alternative algorithm to search for an item is a binary search.

State why a binary search could **not** have been performed on the data stored in the `MyList` array.

.....

.....[1]

(c) The integers in the `MyList` array are to be sorted using an insertion sort algorithm.

The original list is:

MyList

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

As the insertion sort algorithm is applied, the numbers in the array will move.

Each item in the array is considered in turn.

The trace table below shows the list after the second item has been considered.

The next item to be processed is 11.

In the table below, show the changing positions of the items. Circle the data item which is considered in each row.

MyList

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

[3]

6 The operating system of a computer system allows multi-programming. More than one process can be loaded into main memory at the same time.

(a) At any given time, a process will be in one of the three states: ready, running or suspended. When a process is initially loaded it will be in the ready state.

(i) Describe how the state of a process changes before execution completes.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

(ii) Describe how interrupts will be used in the management of the processor time.

.....

.....

.....

.....[2]

(b) The backing store must also be managed by the operating system.

State what is used by the operating system to manage the backing store.

.....

.....[2]

(c) Name **one** other resource which must be managed by the operating system.

.....[1]