CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the October/November 2012 series

9691 COMPUTING

9691/33 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 2012 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

1 (a) (i) -An attribute/or combination of attributes
-Which ensures the records in the table are unique

(ii) -the primary key from Entity X
- Matches to the foreign key
-The same key in table Y
-Many records from entity Y will have a matching record in entity X [3]

(b) (i) -Two sensible attributes for Student (but none which relate to the Book or Loan tables)
- Two sensible attributes for Book (but none which relate to the Student or Loan tables) [2]

(ii) *Loan table*
-Loan attributes include `StudentID` and `BookID`
-Primary key `StudentID + BookID + IssueDate` [2]

(ii) *Data inconsistency*
-Copies of a data item appear in a table and are not consistent with a copy in a second table
-For example - a person who changes address - address changed in one table but original address still appears in some other table [2]

[Total: 11]

2 (a) (i) –90 [1]

(ii) –38 [1]

(iii) A6 [1]

(b) (i) -1 as a carry bit
-0101 0011 [2]

(ii) -The required result is outside the range of the integers permitted using only 8-bits
-Overflow has occurred
(1 per -, max 1) [1]

(c) (i) +5 [1]

(ii) +22 [2]

[Total: 9]

3 **(a)** **(i)** At step 4 ...
-The contents of the Memory Data Register are copied to the Current Instruction Register

(ii) *At step 3...*
-The Memory Address register contains an address
-Copy contents of this address to the Memory Address Register [1]

(b) **(i)** -The processor can directly understand each instruction
-Instruction requires no translation before they can be executed [1]

(ii) -The assembler software translates each assembly language instruction into machine code
-Takes the source file and produces an object/executable file
-Finds and reports all errors in the source/assembly language program
-Looks up the binary code for each instruction
-Constructs a symbol table for all symbolic addresses used by the programmer
(1 per -, max 3) [3]

(c) **(i)** *Direct addressing*
-the operand part of the instruction is treated as an actual memory address
-e.g. LD 1087 will load the contents of memory address 1087 to the Accumulator [2]

(ii) *Relative addressing*
-all addresses are formed by adding a constant number to each address
-the offset to be added is held in a base register
-relative addressing allows for relocatable code
(1 per -, max 2) [2]

[Total: 10]

4 **(a)** *Compiler*
-an object file
-reports all errors
-symbol table
(1 per -, max 2) [2]

(b) *Advantages of an interpreter ...*
-The entire program does not have to be written before an attempt can be made to execute it
-Program development likely to be faster
-Better diagnostics
(1 per -, max 2) [2]

(c) *Syntax analysis*
-The compiler will make reference to meta-language statements (e.g. BNF)
-The statements describe all possible forms of construction for each keyword
-Similar meta-language rules exist for permitted identifier names
-Errors is either the statement composition or identifier names are reported
(1 per -, max 3) [3]

(d) *Linker and loader*

Linker software
-links segments of code
-which have been compiled independently
-needed when the programmer has developed program libraries (which can then be used by many applications)

Loader software
-loads object/executable code into main memory
-many are relocatable loaders
(1 per -, max 2) [2]

[Total: 9]

5 (a) The first item to join the structure will be the first item to leave
NB Acronym insufficient [1]

(b) (i)
```
PROCEDURE AddToQueue
   IF TailOfQueue = 101
      THEN
         Output "Refused - Queue is already FULL"
   ELSE
      INPUT NewItem
      MyQueue(TailOfQueue) ← NewItem
      TailOfQueue ← TailOfQueue + 1
   ENDIF
END PROCEDURE
```

Mark as follows:
-TailOfQueue = 101
-NewItem assigned
-Assigned to MyQueue[Tail]
-TailOfQueue ← TailOfQueue + 1 [4]

(ii)
```
PROCEDURE RemoveFromQueue
   IF Head + 1 = Tail
      THEN
         Output "Refused - Queue is already EMPTY"
   ELSE
     Output MyQueue(HeadOfQueue)
     HeadOfQueue ← HeadOfQueue + 1
   ENDIF
END PROCEDURE
```

Mark as follows:
-Test for empty queue
Output "Empty queue message"
-Item leaving is MyQueue[HeadOfQueue]
-HeadOfQueue incremented
(1 per -, max 2) [2]

(b) *Application of a queue ...*
-One mark for the application
- Characters arriving in a keyboard buffer
- Spooling of print jobs by the operating system
- Anything reasonable

-Final mark for justification for queue [2]

[Total: 9]

6 **(a)** *Strategies for use of the processor*
Round robin
-each job gets a time slice in turn
-Each job is allocated a set amount of time for use of the processor
-All loaded jobs will get some use of the processor

Allocate priorities for all jobs
-Jobs which require a lot of processing get low priority / jobs which do little processing get high priority
-Aim is to get a high throughput of jobs
Max 2 for each strategy [4]

(b) **(i)** *Any two ...*
-A peripheral e.g. printer
-to inform the processor it is out of paper/paper jam/ or similar

-user
-has pressed the 'Reset' button

-keyboard
-has generated an interrupt to say data has been entered and requires saving

-mouse
-has generated a signal e.g. click which will result in some action e.g. a refresh of the screen

-clock interrupt
-must complete the current f-e cycle

-software generated interrupt
-divide by zero error
(1 per -, max 2) [4]

(iii) -Determine the source of the interrupt
-Mask out all interrupts of a lower priority
-Save the contents of the Program Counter
-Save the contents of all other registers (on the stack)
-Load the appropriate Interrupt Service Routine (ISR)
-Run the ISR code
-Restore the contents of the registers
-Restore the contents of the PC
-Restore all lower priority interrupts
-Resume the next process [5]

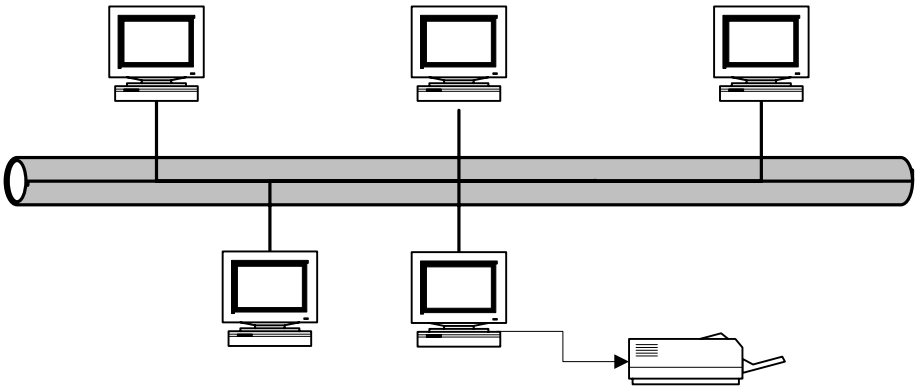
[Total: 13]

7 **(a)** Two different media
Copper wire
-Many different variations - coaxial - twisted pair - thick Ethernet - thin Ethernet

Optic fibre cabling
-Separate fibres used for separate signal
-Data travels very fast

Radio/Microwave signals
-Wireless communication

[4]



(b) **(i)** -Single cable run
-Four computers attached
-Computer with printer attached
-Terminators
NB It the graphic is not a clear attempt at a computer - it must be clear that the 'boxes' are <u>labelled</u> in some way to indicate 'computer'
(1 per -, max 3) [3]

(ii) -WAN is a collection of computer or networks which are <u>connected</u>
-Over a wide geographical area
-Or by example e.g. the Internet
(1 per -, max 2) [2]

(iii) -Use of user Ids with password
- Used to gain access to the network/authorise use of the computer syste

-Use of encryption
- Transaction data/debit/credit card data is particularly sensitive
- Safeguard the data against unauthorised reading
- Data should be sent using SSL / over a secure connection only

-Authentication
- General description of any technique which addresses the issue of the need to 'authenticate' the user of the computer system

-Authorisation
- General description that the user is 'authorised' - using network account settings - to perform certain actions
- Access to some software is available to selected users from certain computers only
- File access rights determine 'who can do what'

(max 2 for each technique)

[Total: 15]

8 **(a)** **(i)** function name: `SumRange`
parameters: `ThisInteger1` and `ThisInteger2` [2]

(ii) 42 [1]

(iii) Error [1]

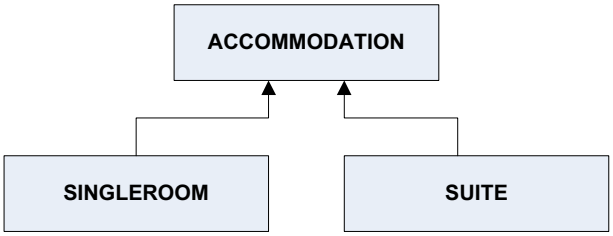
(iv) Error [1]

(v) Error [1]

(b) A function always returns a value - (procedures may/may not return a value) [1]

[Total: 7]

9 **(a)** *Mark as follows:*
3 Classes
Correct hierarchy
Arrows point to parent class



[3]

(b) -A class is the design / the blueprint (from which objects are later created)
-An object is an instance of a class
-An object must be based on a class definition
-Many objects can exist for the same class
-A class consists of properties and methods
(1 per -, max 2) [max 2]

(c) -Encapsulation means an object can only supply its property values through methods designed for reading and storing of the data
- (E.g. The ACCOMMODATION class)
-would require both a `'getRoomNo'`
-and `'setRoomNo'` method [2]

[Total: 7]