UNIVERSITY OF CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Level

MARK SCHEME for the October/November 2011 question paper for the guidance of teachers

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 must be read in conjunction with the question papers and the report on the examination.

- Cambridge will not enter into discussions or correspondence in connection with these mark schemes.

Cambridge is publishing the mark schemes for the October/November 2011 question papers for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level syllabuses and some Ordinary Level syllabuses.

1 **(a)** -Temporarily storing data…
-ensures jobs are kept separate (// ensures no jobs are lost)
-to compensate for different speeds of operation of devices
-when jobs sent to a single device
(1 per -, max 2) [2]

(b) -Print jobs are stored on central/temporary storage
-Reference to job is stored
-along with location of print job on the storage medium
-Jobs are held in print queue
-Jobs may be given a priority
-the job at the top of the print queue/ highest priority is the next to be printed
(1 per -, max 4) [4]

2 **(a)** **(i)** -Stores the address of the memory location to be used next
-The value/address in the PC is loaded into the MAR…
-to show the address of the instruction to be fetched

-The address /operand of the current instruction …
-in the CIR (is loaded into the MAR)
(1 per -, max 3) [3]

(ii) -Modifies the address held in the CIR…
-by the addition of the contents of IR/an integer…
-used in indexed addressing
(1 per -, max 3) [3]

(b) -Buses connect up the different registers/components/devices in the computer
-Data bus carries contents of a memory location/contents of a register/a data value/an address/an instruction
-Data bus is bi-directional // data bus used to read/write data // Address bus is uni-directional //
-Address bus carries an address of a memory <u>location/device</u>
-the address bus carries an address from the processor to main memory / a device
(1 per – max 3) [3]

3 **(a)** **(i)** 395 = 0011 1001 0101
(1 per nybble)

(ii) 395 = 18B
(1 per digit) [3]

(b) **(i)** 01111111 01111111
(1 per byte) [2]

(ii) 11111111 10000000 OR 10111111 10000000
(1 per byte) [2]

(iii) 11111101 = –128 +(64+32+16+8+4+1) = –3
01101000 = ½ + ¼ + 1/16 = 13/16
Number represented = 13/16 * ½^3 {1/8}
= 13/128 (or .1015625)
OR:
11111101 = –128 +(64+32+16+8+4+1) = –3
(01101000 = 0.1101)
= 0.1101 * 2^–3
= 0.0001101
= 1/16 + 1/32 + 1/128 = 13/128

Accept mantissa: –3/128
exponent: +104

(1 per line, max 4) [4]

4 **(a)** -Danger of unauthorised access to the data // intrusion of privacy
-Data may be used against the patient's interests
-Data may be corrupted/inaccurate (making the information poor quality)
-Data may be used for purposes that the patient does not agree with // e.g. sale to drug companies …
(1 per -, max 3) [3]

(b)

Measure …	Explanation …
-Use of passwords	to control access to the data
-Agreements of data use	to restrict how the data can be used
-patient permission must be given	before data is passed to third party
-some access to the data is made read only // different users have different access rights	to control who can see/amend what data
-Data is encrypted	to make it incomprehensible
-protected by firewalls	to safeguard against unauthorized access
-Data is kept physically safe	example
-backing up files	to safeguard data security
-validation checks done on data input/amendments	safeguards data integrity
-patients allowed access to their own data	so that accuracy can be verified/ corrections can be made
-punishment e.g. fines	to discourage misuse of data

Mark as follows:
3 × Measure + explanation
2 × Measure only [5]

5 (a) E.g. -Touch sensor/pressure sensor/infrared sensor/other sensible
-Needed to tell robot when components arrive/To investigate orientation of component/to tell when it has applied enough pressure to pick it up

E.g. -Actuator (electric motor/stepper motor/end effecter) of some sort
-Needed to move robot arm/to physically interact with component/to screw the two components together
-(Speaker/LCD display) conditional on:
-a description of error reporting (2 or 0 marks)

(1 per -, max 4) [4]

(b) e.g.-Cheaper, do not need to be paid
-Work 24/7
-Do not require heat, light, space, ventilation, facilities
-robots can work in hazardous environments
-Items/actions produced are all to a consistent high standard // fewer errors
-Reliable/workers can be off work/will never strike
-Actions are more accurate than those of human.
(1 per -, max 4) [4]

(c) -May involve simply changing from one stored program to another
-set new parameters for current program
-edit program / writing new program code
-by physically being moved through intermediate positions …
-…which the system can then replicate
(1 per -, max 3) [3]

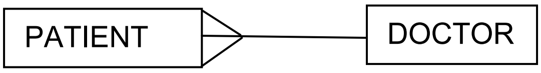
6 -scheduling is designed to maximise use of resources

-Scheduling uses three states for jobs
-**ready**/runable/Waiting in ready queue
-**blocked**/suspended because they are waiting for resource to become available
-**running** the job being processed

-HLS manages which job is the next to be loaded into ready queue
-LLS manages which runable job is allocated processor time next

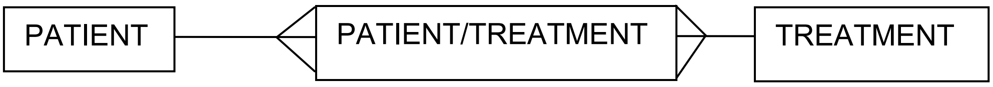
-Name of a scheduling algorithm, e.g. round robin, priority queue
-…with explanation
(1 per -, max 6) [6]

7 **(a) (i)**



[1]

(ii)



(1 per relationship, 1 for sensibly named link table) [3]

(b) (i) -Attribute/Field which is unique to record and is used to identify it // identifier for a tuple
-e.g. PatientID in PATIENT table [2]

(ii) -Attribute in one table which links to the primary key in another table
-e.g. DoctorID in PATIENT table [2]

(iii) -a field/attribute used to sort/search/index the table (on an attribute other than the primary key)
-e.g. Patient name in the PATIENT table to search for a patient by name // Illness in the patient table to find a list of all patients with a particular illness [2]

8 **(i)** -Describes machine code/assembly language
-languages which use the basic machine operations of the processor
-close to the architecture of the processor
-assembly language has a one-to-one mapping with machine code
-assembly language uses mnemonics/labels

(1 per -, max 2) [2]

(ii) -problems are modelled with objects
-objects are defined in a class
-Objects contain both the properties/data/attributes and the methods (needed to manipulate the properties)
-properties can be read or written using methods
-Uses inheritance to allow some objects to use the data and methods of a parent class
-Mention of data encapsulation
(1 per -, max 2) [2]

(iii) -describes what is to be accomplished
-not how no algorithm written
-the user states what is to be found/set a goal
-Consists of a set of facts and rules
-Rules are applied to the data until the goal is reached
-Mention of backtracking/instantiation
(1 per -, max 2) [2]

(iv) -Program describes how to solve the problem in a sequence of steps/algorithm
-lends itself to top-down design / modularisation
-using procedures/functions
(1 per -, max 2) [2]

9 (a) (i) -comes after the lexical analysis stage
-tokenised version of program is scanned
-Check on format/grammar of statements // or by example e.g. matching brackets
-Error diagnostics are issued as appropriate
-produces code ready for the code generation stage
Also accept:
-Jump destinations/labels checked for existence
-data type mismatch
-Check that variables have been declared
-Check for existence of library modules
(1 per -, max 4) [4]

(ii) -In lexical analysis stage keywords are identified by comparing to list of accepted words
-the format of instruction/token string is compared …
-… to forms for acceptable expressions and statements.
-as defined by the meta language used
-example of a syntax error e.g. IF THEN x=3
(1 per -, max 3) [3]

(b) (i) -object code is difficult to interfere with
-object code runs faster than interpreted source code
-compiler can optimise executable code
-the code is not translated each time the program is run
-Compiler does not need to be present when the program is run
-Compiled code will be free from syntax errors
(1 per -, max 2) [2]

(ii) -Errors are (more) easily located…
-reports errors when source code is present...
-stopping at the point of the error
-Parts (only) of program can be tested / testing can be started before all the program is written
-errors when found can be immediately corrected.
(1 per -, max 2) [2]

10 **(a)** **(i)** if there is a digit at the end it must be a <non-zero-digit> // can't end in a 0

(ii) W is not defined as a <letter> // W is not allowed

(iii) can't end with two digits [3]

(b) 5 is a <non-zero-digit> therefore it is a <digit>
6 is a <non-zero-digit>
y is a <letter> and therefore is a <group>
A is a <letter>, hence Ay is <letter><group> therefore it is a <group>
(1 mark per line) [4]