AQA

A-LEVEL
Computing

COMP3 – Problem Solving, Programming, Operating Systems, Databases and Networking
Mark scheme

2510
June 2016

Version: 1.0 Final

Mark schemes are prepared by the Lead Assessment Writer and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all associates participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the students' responses to questions and that every associate understands and applies it in the same correct way. As preparation for standardisation each associate analyses a number of students' scripts. Alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, associates encounter unusual answers which have not been raised they are required to refer these to the Lead Assessment Writer.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of students' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this mark scheme are available from aqa.org.uk

The following annotation is used in the mark scheme:

; - means a single mark
// - means alternative response
/ - means an alternative word or sub-phrase
A - means acceptable creditworthy answer
R - means reject answer as not creditworthy
NE - means not enough
I - means ignore
DPT - means "Don't penalise twice". In some questions a specific error made by a candidate, if repeated, could result in the loss of more than one mark. The **DPT** label indicates that this mistake should only result in a candidate losing one mark, on the first occasion that the error is made. Provided that the answer remains understandable, subsequent marks should be awarded as if the error was not being repeated.

<table>
<tr><th>Qu.</th><th>Part</th><th>Sub-part</th><th>Marking guidance</th><th>Mark</th></tr>
<tr><td>1</td><td>a</td><td></td><td>
<table>
<tr><th>Situation and Procedure</th><th>Authentication</th><th>Authorisation</th><th>Accounting</th></tr>
<tr><td>A web server generating a log of the IP addresses of computers that have accessed it.</td><td></td><td></td><td>✓;</td></tr>
<tr><td>Using a digital signature when sending an e-mail message.</td><td>✓;</td><td></td><td></td></tr>
</table>
R. Responses in which more than one column is ticked on a row
A. Responses in which a symbol other than a tick is used
</td><td>2</td></tr>
<tr><td>1</td><td>b</td><td></td><td>
Virus is (max 2 marks):

Program that attaches itself to / conceals itself within another program/file;
Self-replicating // program can copy itself; N.E. Viruses spread
Has malicious purpose; A. Is a type of malware A. Examples of malicious purposes

Difference to worm (max 2 marks):

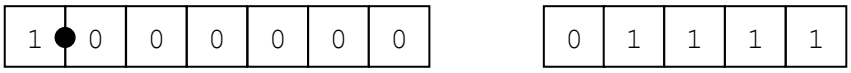
Worm duplicates by exploiting <u>network</u> security weaknesses / across <u>network</u> (whereas virus copies itself by attaching to other files);
Worm is standalone software (whereas virus conceals itself within another file);
Worm replicates without user action (whereas virus relies on user running program to replicate it);

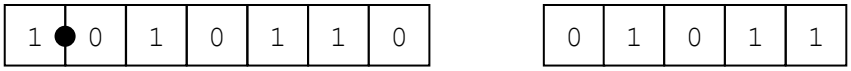
MAX 3
</td><td>3</td></tr>
</table>

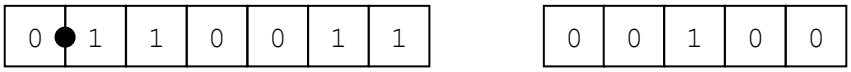
1	**c**		Hash/digest produced/calculated <u>from message contents</u> // (shortened) value <u>calculated from message</u>; **A.** Message is hashed **A.** Message digest created **N.E.** Hash produced Hash encrypted; A's private key is used for the hash encryption; **N.E.** Uses A's private key Encrypted hash is known as the (digital) signature; (Digital) signature is appended to message; **A.** Encrypted hash for digital signature **I.** Description of encryption not related to digital signature **MAX 4**	4

2	**a**		<table><tr><th>Reverse Polish Notation</th><th>Equivalent Infix Expression</th></tr><tr><td>18 9 -</td><td>18 - 9 **A.** (18 - 9) **R.** 9 – 18</td></tr><tr><td>10 4 - 12 ×</td><td>(10 - 4) × 12 **R.** 10 - 4 × 12 **A.** * for ×</td></tr></table> **1 mark** per correct infix expression	2

2	**b**		Simpler/quicker for a machine/computer to evaluate // simpler to code algorithm **A.** Easier as BOD **R.** To understand Do not need brackets (to show correct order of evaluation/calculation); **N.E.** Does not use brackets **T.O.** No brackets so less storage space used Operators appear in the order required for computation; No need for order of precedence of operators; No need to backtrack when evaluating; **A.** RPN expressions cannot be ambiguous as BOD	1

3	**a**		 **1 mark** for correct mantissa **1 mark** for correct exponent	2

3	**b**		 Mantissa Exponent **1 method mark** for either: • showing correct value of both mantissa and exponent in denary (Mantissa = -0.65625 // -21/32, Exponent = 11) • showing binary point shifted 11 places to right in binary number • indicating that final answer calculated using answer = mantissa x $2^{exponent}$ **1 mark** for correct answer Answer = -1344 *If answer is correct and some working has been shown, award two marks, even if working would not have gained credit on its own.*	2

3	**c**		**2 marks** for working: Correct representation of 12¾ in fixed point binary: 1100.11; **A.** any number of preceding 0s or succeeding 0s Showing the correct value of the exponent in denary (4) or binary (100) // showing the binary point being shifted 4 places left; Showing the correct value of the mantissa in floating point binary: 0.110011; **MAX 2** **1 mark** for correct mantissa and exponent together:  Mantissa Exponent *If answer is correct and some working has been shown, award three marks, even if working would not have gained credit on its own.* *Marks for working can be awarded in the answer.eg correct mantissa and exponent*	3

3	**d**	**i**	Maximises precision/accuracy for given number of bits; **Note: Must have concept of given number of bits or an example of this eg word length.** Unique representation of each number // simpler to test for equality of numbers;	2

3	**d**	**ii**	There is no need to store both the 0 and the 1 // only one of the two bits needs to be stored (as one can be inferred from / worked out from the other); Therefore another bit could be freed up to use in the mantissa // one extra bit of precision could be stored; **A.** A description that clearly implies the mantissa if mantissa is not explicitly stated	2

4	**a**		**Parity Bit:** 1; **Start bit, Stop Bit:** Can be either 0 or 1, but must both be different to get mark;	2

4	**b**		**Definition (1 mark):** Receiver and transmitter (clocks) do not need to be/are not (exactly) synchronised // transmission of data without use of external clock signal // receiver and transmitter clock only synchronised at start of/for length of transmission // data sent as soon as available rather than waiting for clock pulse/ synchronisation symbol; **Explanation of start and stop bits (max 2 marks):** Start bit synchronises receiver (clock) (to transmitter/data) // locks receiver and transmitter in phase // starts receiver's clock // wakes receiver; Stop bit allows start bit to be recognised // allows receiver to process received bits; **A.** Start and stop bits indicate when data is being transmitted/ begins – if neither of the other two marks for start and stop bits have been awarded	3

4	**c**		1010001; **A.** Separator between digits eg comma	1

4	**d**		It is the parity bit; **A.** Odd parity bit **A.** If there are an even or odd number of 1s in the input	1

4	**e**		Only a small quantity of data to send // data transmission speed not important; Widespread availability of USB/serial connections; Serial communication avoids crosstalk // interference between signals on each wire; Serial communication avoids data skew; **A.** Serial communication is cheaper to implement with a suitable reason given **A.** For future flexibility if devices were moved further apart **N.E.** Serial is less error prone / fewer errors **MAX 2**	2

5	**a**		John, Rachel, Paul; **R.** If not in correct order **I.** Incorrect spellings of names, as long as the name is comprehensible **I.** Quotation marks	1

5	**b**		<table><tr><th>Time Complexity</th><th>Tick one box</th></tr><tr><td>O(n)</td><td></td></tr><tr><td>O(log n)</td><td>✓</td></tr><tr><td>O(n^2)</td><td></td></tr></table> **A.** Alternative symbol which clearly indicates just one box eg cross, Y, Yes **R.** Answers in which more than one row is ticked	1

5	**c**		**1 mark** for Start Index containing the index of the array item storing the root node (John) **1 mark** for John, Hannah and Rachel being stored, each with left and right pointer values storing the indices of the correct child nodes **1 mark** for Bradley, Jo, Paul and Tina being stored, each with left and right pointers storing appropriate rogue values eg -1, 0, NIL, NULL **R.** A dash or blank as the rogue value **Three example solutions are shown below, but the data items can be stored at any positions within the array, as long as the pointers correctly reflect this.** **Start Index = 1** **Index** / **Left Pointer** / **Data** / **Right Pointer** [1] / 2 / John / 3 [2] / 4 / Hannah / 5 [3] / 6 / Rachel / 7 [4] / -1 / Bradley / -1 [5] / -1 / Jo / -1 [6] / -1 / Paul / -1 [7] / -1 / Tina / -1 **Start Index = 4** **Index** / **Left Pointer** / **Data** / **Right Pointer** [1] / -1 / Bradley / -1 [2] / -1 / Jo / -1 [3] / 1 / Hannah / 2 [4] / 3 / John / 6 [5] / -1 / Paul / -1 [6] / 5 / Rachel / 7 [7] / -1 / Tina / -1 **Start Index = 1** **Index** / **Left Pointer** / **Data** / **Right Pointer** [1] / 2 / John / 5 [2] / 3 / Hannah / 4 [3] / -1 / Bradley / -1 [4] / -1 / Jo / -1 [5] / 6 / Rachel / 7 [6] / -1 / Paul / -1 [7] / -1 / Tina / -1	3

5	**d**		**Difference between Static and Dynamic (2 marks):** Static structures have fixed (maximum) size whereas size of dynamic structures can change // Size of static structure fixed at compile-time whereas size of dynamic structure can change at run-time; Static structures can waste storage space/memory if the number of data items stored is small relative to the size of the structure whereas dynamic structures only take up the amount of storage space required for the actual data; Static structures (typically) store data in consecutive memory locations, which dynamic data structures (typically) do not // Dynamic data structures (can) (require memory to) store pointer(s) to the (next items which static structures typically do not need); **MAX 2** **A.** Just one side of points, other side is by implication **Heap (1 mark):** Memory allocated/deallocated at run-time/for new items (to dynamic data structure); (Provides a) pool of free/unused/available memory; **N.E.** To store new items	3

5	**e**	**i**	Bradley, Hannah, Jo, John, Paul, Rachel, Tina; **R.** If not in correct order **I.** Incorrect spelling of names, so long as name is comprehensible **I.** Quotation marks	1

5	**e**	**ii**	(Ascending) Alphabetic order; **A.** Alphabetic, it is sorted	1

5	**f**		Graph may contain cycles / loops / circuits (so must keep track of which nodes already visited); Graph may not be connected (so some nodes may be unreachable); Graph may be weighted (so a more complex algorithm that accounts for the weights may be required); **N.E.** Graphs can be directed **MAX 1**	1

6			**SUBJECT MARKING POINTS:** **How systems work:** *Rich client:* • Applications run (locally) on computer // all processing done on (local) computer // applications installed locally **A.** On client *Thin client:* • All/most processing done by (central) server // applications not installed on (thin client) workstations // all applications on server; **A.** All software run on server • Keystrokes/mouse clicks/user input transmitted from workstation/terminal to server over network, **A.** Workstations are just interfaces • Image/data needed to produce image transmitted from server to terminal over network • Operating system loaded by clients from server at boot **How hardware differs for thin client:** • Higher bandwidth network connection required • Network must use switch not hub • Slower processor /reduced RAM/ no HDD required in workstations, **A.** Other examples of limited hardware requirements, **A.** 'Dumb terminal' • Server must have multiple processors/a lot of RAM **N.E.** more powerful / less powerful, higher performance / lower performance, cheaper / more expensive Accept the opposite of points eg for "a thin client system could use a slower processor" accept "a thick client system would need a faster processor" but don't award marks for a point and its opposite point. **Why SaaS is a type of thin client:** Software is run on a remote computer (not locally, so an example of thin client) **A.** Server, web server for "remote computer" **N.E.** Accessed via Internet **What distinguishes SaaS from other types of thin client:** • SaaS is accessible anywhere that there is an Internet connection // is used via the Internet • Customers usually purchase access to SaaS instead of buying software outright • SaaS is usually managed by an application service provider /	8

another company / a contractor // company using SaaS does not need to manage software
- SaaS usually works in (web) browser

HOW TO AWARD MARKS:

Mark Bands and Description	
7-8	*To achieve a mark in this band, candidates must meet the subject criterion (SUB) and all 5 of the quality of written communication criteria (QWCx).* *SUB* Candidate has written a detailed explanation of how thin client systems work in comparison to rich client systems, and has also made a good comparison of the hardware required for both types of system. Some points have been made about how SaaS is distinguished form other types of thin client system. The candidate has made at least seven subject-related points. *QWC1* Text is legible. *QWC2* There are few, if any, errors of spelling, punctuation and grammar. Meaning is clear. *QWC3* The candidate has selected and used a form and style of writing appropriate to the purpose and has expressed ideas clearly and fluently. *QWC4* Sentences (and paragraphs) follow on from one another clearly and coherently. *QWC5* Appropriate specialist vocabulary has been used.
5-6	*To achieve a mark in this band, candidates must meet the subject criterion (SUB) and 4 of the 5 quality of written communication criteria (QWCx).* *SUB* Candidate has made some points in both of the areas of how thin client systems work in comparison to rich client systems, and how the hardware requirements of each type of system vary. The candidate has made at least five subject-related points. *QWC1* Text is legible. *QWC2* There may be occasional errors of spelling, punctuation and grammar. Meaning is clear. *QWC3* The candidate has, in the main, used a form and style of writing appropriate to the purpose, with occasional lapses. The candidate has expressed ideas clearly

	and reasonably fluently. *QWC4* The candidate has used well-linked sentences (and paragraphs). *QWC5* Appropriate specialist vocabulary has been used.
1-4	*To achieve a mark in this band, candidates must meet the subject criterion (SUB) and 4 of the 5 quality of written communication criteria (QWCx).* *SUB* Candidate has made some relevant points, but these are superficial or narrow in scope. *QWC1* Most of the text is legible. *QWC2* There may be some errors of spelling, punctuation and grammar but it should still be possible to understand most of the response. *QWC3* The candidate has used a form and style of writing which has many deficiencies. Ideas are not always clearly expressed. *QWC4* Sentences (and paragraphs) may not always be well-connected. *QWC5* Specialist vocabulary has been used inappropriately or not at all.
0	Candidate has made no relevant points.

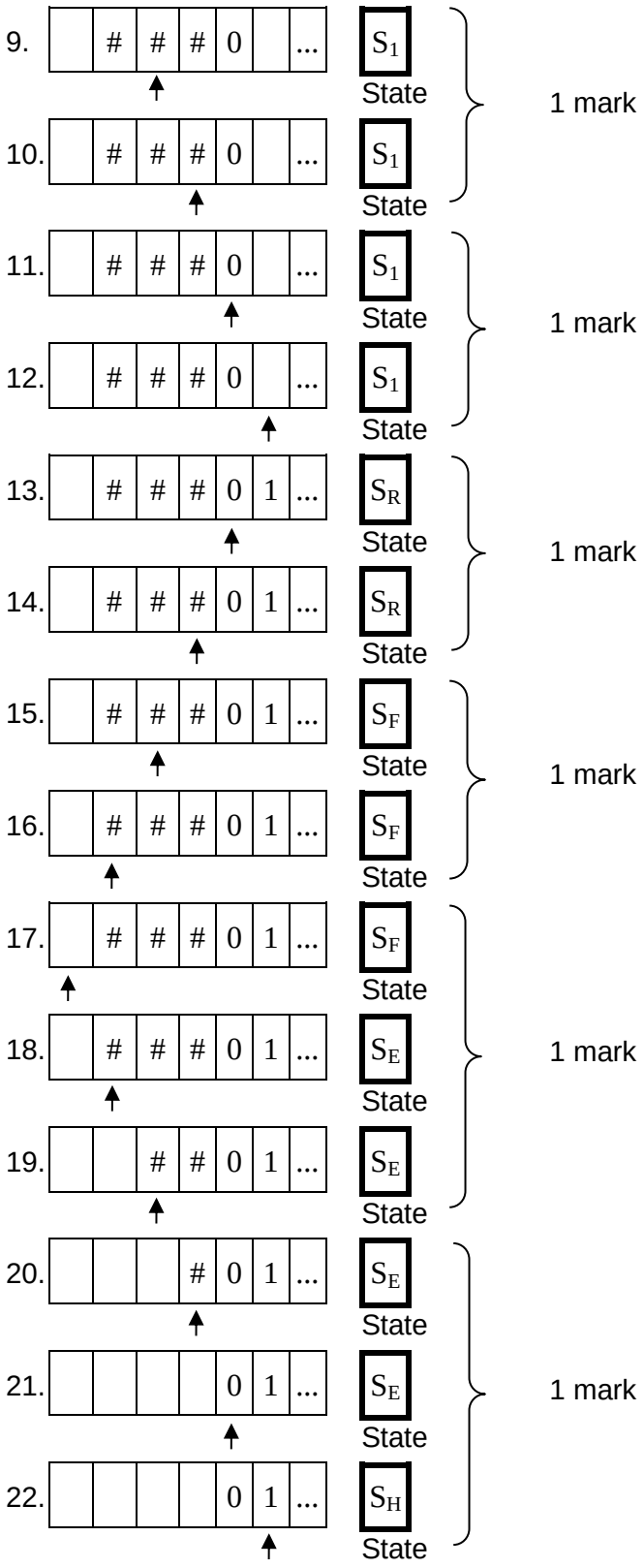
Note: Even if English is perfect, candidates can only get marks for the points made at the top of the mark scheme for this question.

If a candidate meets the subject criterion in a band but does not meet the quality of written communication criteria then drop mark by one band, providing that at least 4 of the quality of language criteria are met in the lower band. If 4 criteria are not met then drop by two bands.

7	**a**			2

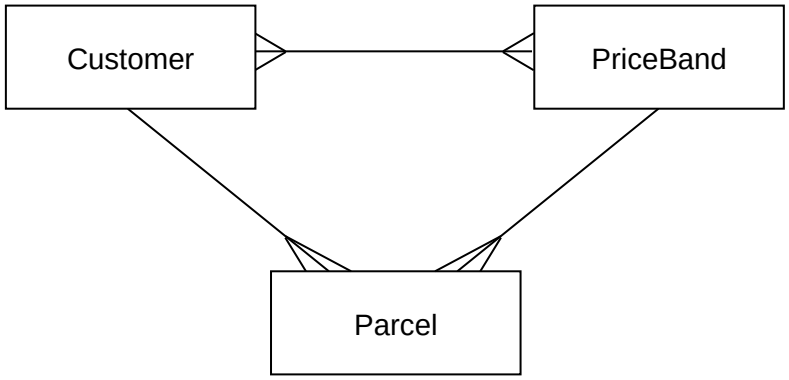
Turing machine component	**Number (1-5) of modern computer system component with most similar role**
Transition function	5; **A.** Program
Tape	3; **A.** Main Memory / Memory

7	**b**		If (and only if) an algorithm exists to solve a problem then a Turing machine can be designed to solve the problem; **A.** Statement made in reverse ie "if a Turing machine exists..." **A.** Statement made as a negative ie "if no algorithm exists ..." **A.** A Turing machine can compute any algorithm // any algorithm can be computed by a Turing machine	1

7	**c**		 **Must have correct tape contents and state for each mark**	6

			A. Blank symbols instead of empty cells **A.** Content written anywhere on the tape as long as the position is correct relative to the read/write head **A.** Read/write head drawn off left hand end of tape at stage 17 if tape contents are written at left hand end of tape **DPT** If the read/write head is not drawn on some rows, this should result in the loss of the mark on the first occasion that it is missing only. Marks should be awarded for subsequent rows, even if the read/write head is not drawn.	

7	**d**		To reverse a (binary) string/number // to produce a copy of a (binary) string/number with the order of the characters/digits reversed; **R.** Flips bits, but **A.** Flips order of bits **A.** Mirror the input	1

8	**a**		**1 mark** for any one correct relationship drawn **2 marks** for all three correct relationships drawn **I.** Any additional writing on diagram  	2

8	**b**		`UPDATE PriceBand`
			`SET Price = 5.99`
			`WHERE ServiceSpeed = "Express"`
			`  AND MinWeight = 1000`
			`  AND MaxWeight = 4999`
			
			1 mark for `UPDATE PriceBand`
			1 mark for `SET Price = 5.99`
			1 mark* for `ServiceSpeed = "Express"`
			1 mark* for either `MinWeight = 1000` or `MaxWeight = 4999` (or both joined by `AND`). **A.** use of >= and <= instead of = if conditions given for both `MinWeight` and `MaxWeight`.
			
			To award both marks indicated by * symbol, the conditions must be joined by `AND`s.
				4

			A. Double or single quotes around `Express` **A.** `Express` written in any case **A.** `£` symbol before `5.99` **A.** Table names before fieldnames **DPT** for fieldname before table name. **DPT** for unnecessary punctuation eg quotes where they should not appear. Allow one semicolon at the very end of the statement, but not at the end of each clause. **DPT** use of incorrect equality operator eg ==	

8	**c**		**Alternative 1**
			
			`SELECT DateSent, Postcode, ServiceSpeed, Price`
			`FROM Parcel, PriceBand`
			`WHERE CustomerID = 109`
			`AND Parcel.ServiceSpeed = PriceBand.ServiceSpeed`
			`AND Parcel.Weight >= PriceBand.MinWeight`
			`AND Parcel.Weight <= PriceBand.MaxWeight`
			`ORDER BY DateSent`
			
			Alternative 2
			
			`SELECT DateSent, Postcode, ServiceSpeed, Price`
			`FROM Parcel INNER JOIN PriceBand ON`
			`Parcel.ServiceSpeed = PriceBand.ServiceSpeed`
			`AND Parcel.Weight >= PriceBand.MinWeight`
			`AND Parcel.Weight <= PriceBand.MaxWeight`
			`WHERE CustomerID = 109`
			`ORDER BY DateSent`
			
			1 mark for `SELECT` clause with correct four fields
			1 mark for `FROM` clause with correct two tables
			1 mark for `CustomerID = 109`
			1 mark for `Parcel.ServiceSpeed=PriceBand.ServiceSpeed`
			1 mark for `Parcel.Weight >= PriceBand.MinWeight AND Parcel.Weight <= PriceBand.MaxWeight`
			1 mark for `ORDER BY DateSent`
			
			MAX 2 of the 3 marks for conditions if not joined by `AND`s
			
			Conditions linking the two tables can be present in either the `FROM` or `WHERE` clause or a mixture of both, as long as they are syntactically and logically correct.
			Marks for correct files/tables in `SELECT` and `FROM` statements should not be awarded if additional fields/tables included, except allow the inclusion of the `CUSTOMER` table in the `FROM` statement so long as it has been correctly linked to the `PARCEL` table.
				6

			Marks can be awarded for the conditions in the `WHERE` statement even if the required tables are not present in the `FROM`. **A.** Table names before fieldnames. **A.** Use of `Alias`/`AS` command eg `FROM Parcel AS P` then use of `P` as table name (note some dialects of SQL do not require `AS` eg `FROM Parcel P`) **A.** Insertion of spaces into fieldnames. **A.** `109` with no delimiters or delimited using `"` or `'`. **A.** Use of `BETWEEN` command for weight range eg `Parcel.Weight BETWEEN PriceBand.MinWeight AND PriceBand.MaxWeight` **A.** `ORDER BY` written as one word `ORDERBY`. **A.** `ASC` at the end of `ORDER BY`. **I.** Unnecessary brackets. **DPT** for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause. **DPT** for fieldname before table name. **DPT** use of incorrect equality operator eg == **Refer responses using nested SQL queries to team leaders.**	

8	**d**		Parcel(<u>ParcelID</u>, ServiceSpeed, Weight, DateSent, CustomerID, RecipientName, HouseNumber, Postcode) PostcodeLookup(<u>Postcode</u>, Street, Town, County) **1 mark** for identifying that a new PostcodeLookup relation is required. Purpose must be clear; it is not sufficient to just make a new relation. Purpose could be made clear by any one of: appropriate name of relation (**A.** Address), approximately the correct attributes (allow, for example, incorrect inclusion of house number or CustomerID) in relation or having Postcode as the primary key. **1 mark** for correct attributes in PostcodeLookup relation and identifying the Postcode as the primary key **1 mark** for correct attributes left in Parcel relation and correct primary key **A.** Answers given as SQL commands. As the question did not ask for this, perfect syntax is not required. **A.** Alternative names for entities, so long as meaning is clear. **A.** Spaces in entity and attribute names. **A.** PostcodeLookup relation called Postcode even through this is the same as an attribute name. **A.** Addition of unnecessary new relation for recipients which is not required for this question. **R.** Do not award marks for correct attributes in a relation if additional attributes included.	3

9	**a**		Processor management // Allocation of processors // Allocation of processor time // (process) scheduling // thread management; **A.** Processing management, CPU management Allocation/management of RAM / memory // allocation of buffers; Allocation/management of / control of I/O devices/peripherals // I/O management // device driver management; File / backing store / secondary store management / access / organisation; Power / battery management; Interrupt handling; **A.** Provision of Application Program Interface / API **A.** interface between hardware and applications **A.** Provision / management of (windows in) user interface **A.** Management of system security **A.** Answers by example, only one example of each type **A.** A description of a type of software management but not just "software management". eg loading of programs, software installation, registering DLLs. **A.** Managing network connections but **R.** Network management **R.** Software management alone unless role of OS in this is clear eg installation of new software, updating registry **MAX 3**	3

9	**b**		User and computer in direct/two-way communication // User makes input to computer then waits for output before making next input; **A.** System, software, program, OS for computer	1

10	**a**		It hides the detail of how the list will be stored/implemented from the programmer // a programmer working on the rest of the program does not need to know how the LinkedList class works // a programmer working on the rest of the program needs only concern themselves with the interface to the LinkedList class; **A.** "user" for "programmer" as BOD mark	1

10	**b**		The procedures/functions are public as a programmer (writing the rest of the program) will need access to the operations defined in the procedures and functions from outside of the class / elsewhere in the program (so they must be public); **A.** just one of procedures or functions **A.** pPocedures/functions will be accessible The data items are private to prevent them being changed directly from outside of the class // to avoid the integrity of the data structure being damaged / changed accidentally (from outside the class); **A.** "elsewhere in program" for "outside of the class" So that the implementation of `LinkedList` can be changed and programs written using only the public functions and procedures will still work; **MAX 2**	2

10	**c**		**OVERALL GUIDANCE:** Solutions should be marked on this basis: • Up to **5 marks** for correctly locating the position to delete the item from. • Up to **3 marks** for deleting the item and updating pointers as required. The addition of any unnecessary steps that do not stop the algorithm working should not result in a reduction in marks. Responses should be accepted in pseudo-code or structured English but not in prose. ***If you are unsure about the correctness of a solution please refer it to a team leader.*** **SPECIFIC MARKING POINTS:** *Correctly locating deletion point (5 marks):* 1. Initialising `Current` to `Start` before any loop; 2. Use of loop to attempt to move through list (regardless of correct terminating condition); 3. Advancing `Current` within loop; 4. Correctly maintaining the `Previous` pointer within loop; 5. Sensible condition to identify position to delete from (suitable terminating condition for loop); *Correctly deleting item (3 marks):* 6. Update `Next` pointer of node before node to delete to point to node after it; 7. Test if item to delete was first item in list, and if so update `Start` pointer instead of `Next` pointer of node before the one to delete; 8. Release the memory used by the item being deleted back to the operating system; Mark point 2 should be awarded if, within the loop, `Current` is being changed (even if not correctly changed). Mark point 4 can be awarded if `Previous` is set to `Current` before `Current` is changed, even if `Current` is not being correctly updated. Mark point 5 can be awarded if there is a sensible condition, even if `Current` is not correctly updated.	8

Mark point 6 can be awarded even if the value of `Previous` was not correctly maintained in the loop.

Mark points 6 and 7 can only be awarded if `Current` has not already been released (or attempted to be released).

Mark point 8 should only be awarded if this is done after and a loop to search for the item to delete, regardless of whether or not the correct item would be found or if it is done inside the loop but also within an if statement that correctly identifies the item to delete.

A. Deletion takes place inside of loop if the correct item to delete had been identified with an if statement and the loop will be exited at some point after deletion.
A. Use of any type of condition controlled loop, as long as logic is correct.
A. Use of alternative variable names and instructions, so long as the meaning is clear.
A. Use of clear indentation to indicate start/end of iteration and selection structures.
A. Responses written in structured English, so long as variable names are used and the descriptions of what will be done are specific.
A. Use of Boolean variable to control loop as long as it is set under the correct conditions and has been initialised.
R. Responses written in prose.
R. Do not award mark points if incorrect variable names have been used, but allow minor misspellings of variable names.

EXAMPLE SOLUTIONS:

The examples below are complete solutions that would achieve full marks. **Refer recursive solutions to Team Leaders.**

Example 1

```
If Start.DataValue = DelItem Then
  Start ← Start.Next
  Release(Start)
Else
  Current ← Start
  Repeat
    Previous ← Current
    Current ← Current.Next
  Until Current.DataValue = DelItem
  Previous.Next ← Current.Next
  Release(Current)
EndIf
```

			Example 2
			
			`Current ← Start`
			`While Current.DataValue ≠ DelItem`
			`  Previous ← Current`
			`  Current ← Current.Next`
			`EndWhile`
			`If Current = Start Then`
			`  Start ← Current.Next`
			`Else`
			`  Previous.Next ← Current.Next`
			`EndIf`
			`Release(Current)`
			
			Example 3
			
			`If Start.DataValue = DelItem Then`
			`  Start ← Start.Next`
			`  Release(Start)`
			`Else`
			`  Deleted ← False`
			`  Current ← Start`
			`  While Deleted = False`
			`  If Current.DataValue = DelItem Then`
			`  Previous.Next ← Current.Next`
			`  Release(Current)`
			`  Deleted ← True`
			`  Else`
			`  Previous ← Current`
			`  Current ← Current.Next`
			`  EndIf`
			`  EndWhile`
			`EndIf`	

11	**a**		**Implementation One** would need to use a linear search // would need to look at every word in the array (before the one that is being searched for) // lookup time is proportional to number of words in list // lookup is O(N); **N.E.** "search" without further clarification that this would be linear **Implementation Two** would use the hash function/hashing to directly calculate where the word would be stored // could jump directly to the correct position/location/index for the word in the array // lookup time is constant regardless of how many words in list // lookup is O(1); **A.** No need to go through words in list	2

11	**b**		The (record for) each word/both words would be stored at the same position/index/location in the array; **A.** The second word would be stored over/replace the first; **N.E.** A collision has occurred Store record/word in the next available position in the array // store a pointer (in each array position) that points to a list of records that have all collided at the position // rehash the word; **A.** Idea that each array position could store more than one record eg five records per location, if explained. **A.** Example of what "next available" might be. **R.** The use of a different hashing function at all times ie not just rehashing.	2

11	**c**		The hash function could compute the same value/location for more than one/two English word(s), so need to verify if the English word stored at the location is the one that is being looked up; To avoid returning a French translation that is for a different English word, which is stored at the same location as the word that is being looked up // if a collision occurred (when storing the words) it will not be possible to tell if the translation is correct; **A.** More than one word could be stored in each location **R.** So that French to English translation can be done **MAX 1**	1

12	**a**		Syntax diagram; **A.** Railroad diagram	1

12	**b**		3	1

12	**c**		**Mark is conditional upon a correct answer to 12b** It requires that a signed binary number starts with a + or a - // it won't allow a signed binary number that starts with a bit/digit/1 or 0 // + and – are not optional /are required; **A.** 'String' or 'number' for 'signed binary number'	1

12	**d**		Some example correct regular expressions are listed below but award a mark for any regular expression that would correctly represent the language accepted by the FSA. a ((ba)|c)* // a((ba)*|c*)* // a((ba)*|c)* // a((ba)|c*)* // a((ba)*|c*)$^{+}$ // a(c*(ba)?)* **A.** Missing brackets around ba as BOD	1