UNIT 4

MARK SCHEME

Guidance for examiners

Positive marking

It should be remembered that learners are writing under examination conditions and credit should be given for what the learner writes, rather than adopting the approach of penalising him/her for any omissions. It should be possible for a very good response to achieve full marks and a very poor one to achieve zero marks. Marks should not be deducted for a less than perfect answer if it satisfies the criteria of the mark scheme.

For questions that are objective or points-based the mark scheme should be applied precisely. Marks should be awarded as indicated and no further subdivision made.

For band marked questions mark schemes are in two parts.

Part 1 is advice on the indicative content that suggests the range of computer science concepts, theory, issues and arguments which may be included in the learner's answers. These can be used to assess the quality of the learner's response.

Part 2 is an assessment grid advising bands and associated marks that should be given to responses which demonstrate the qualities needed in AO1, AO2 and AO3. Where a response is not credit worthy or not attempted it is indicated on the grid as mark band zero.

Banded mark schemes

Banded mark schemes are divided so that each band has a relevant descriptor. The descriptor for the band provides a description of the performance level for that band. Each band contains marks.

Examiners should first read and annotate a learner's answer to pick out the evidence that is being assessed in that question. Once the annotation is complete, the mark scheme can be applied.

This is done as a two stage process.

Stage 1 – Deciding on the band

When deciding on a band, the answer should be viewed holistically. Beginning at the lowest band, examiners should look at the learner's answer and check whether it matches the descriptor for that band. Examiners should look at the descriptor for that band and see if it matches the qualities shown in the learner's answer. If the descriptor at the lowest band is satisfied, examiners should move up to the next band and repeat this process for each band until the descriptor matches the answer.

If an answer covers different aspects of different bands within the mark scheme, a 'best fit' approach should be adopted to decide on the band and then the learner's response should be used to decide on the mark within the band. For instance if a response is mainly in band 2 but with a limited amount of band 3 content, the answer would be placed in band 2, but the mark awarded would be close to the top of band 2 as a result of the band 3 content. Examiners should not seek to mark candidates down as a result of small omissions in minor areas of an answer.

Stage 2 – Deciding on the mark

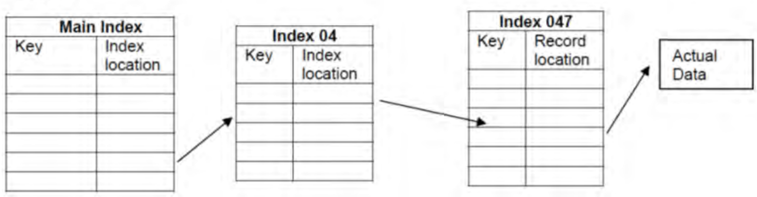
Once the band has been decided, examiners can then assign a mark. During standardising (marking conference), detailed advice from the Principal Examiner on the qualities of each mark band will be given. Examiners will then receive examples of answers in each mark band that have been awarded a mark by the Principal Examiner. Examiners should mark the examples and compare their marks with those of the Principal Examiner.

When marking, examiners can use these examples to decide whether a learner's response is of a superior, inferior or comparable standard to the example. Examiners are reminded of the need to revisit the answer as they apply the mark scheme in order to confirm that the band and the mark allocated is appropriate to the response provided.

Indicative content is also provided for banded mark schemes. Indicative content is not exhaustive, and any other valid points must be credited. In order to reach the highest bands of the mark scheme a learner need not cover all of the points mentioned in the indicative content but must meet the requirements of the highest mark band. Where a response is not creditworthy, that is contains nothing of any significance to the mark scheme, or where no response has been provided, no marks should be awarded.

Q	Answer	Mark	AO1	AO2	AO3	Total
1a	• The address of the next instruction is copied from the PC to the MAR • The instruction is copied to the MDR • The PC is incremented so that it holds the address of the next instruction	1 1 1	1.1b 1.1b 1.1b			3

1b	Using Register R as the working register and Register S as secondary register
	Memory locations containing numbers:
	0001 = location of first number
	0002 = location of second number
	0003 = location to store result
	(accept any consistent use)
	
	`CLR R`
	`LOD R, 0001`
	`LOD S, 0002`
	`ADD R,S`
	`STR R 0003`
	`CLR R`
	
	1 mark for clearing register ready for use and at end.
	1 mark for loading values into registers.
	1 mark for adding and storing result.
		1
		1
		1
					3.1b
					3.1b
					3.1b
						3

2	**Indexed sequential file** • Records in key order • Index allows data to be accessed directly / faster / contains key and record address **Multilevel index** • There is a main index which contains the location of the next index • This process may extend to several levels. The last index contains the physical address of the record Diagram:  **Marking of diagram** 1 mark for three suitably labelled tables plus "actual data" 1 mark for three suitable arrows	1 1 1 1 1 1	1.1b 1.1b 1.1b 1.1b	2.1a 2.1a		6
3	**The purpose of a hashing algorithm:** • file where physical location of the record is calculated from the data in the key field • this calculation is carried out by a hashing algorithm **The need for an overflow area:** • A data collision occurs when two data items are hashed to the same location • In this case there needs to be an overflow areas	1 1 1 1	1.1b 1.1b 1.1b 1.1b			6

<table>
<tr><th>Q</th><th>Answer</th><th>Mark</th><th>AO1</th><th>AO2</th><th>AO3</th><th>Total</th></tr>
<tr><td></td><td>where the latest data is stored
The need for the random access file to be re-organised on occasions:
• When there are many items in the overflow area, access may become slow,
• In which case a new hashing algorithm is required and a larger file may be needed</td><td>1
1</td><td>1.1b
1.1b</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>
<table><tr><th>Destination</th><th>Cost</th><th>GoTo</th></tr><tr><td>A</td><td>0</td><td>A</td></tr><tr><td>B</td><td>5</td><td>C</td></tr><tr><td>C</td><td>3</td><td>C</td></tr><tr><td>D</td><td>7</td><td>C</td></tr><tr><td>E</td><td>11</td><td>C</td></tr></table>
1 mark for correct first, second and third row
1 mark for correct fourth row
1 mark for correct fifth row</td><td>1
1
1</td><td></td><td>2.1a
2.1a
2.1a</td><td></td><td>3</td></tr>
<tr><td>5a</td><td>1 mark for description x 2
1 mark for example (must match description) x 2

Any one of:
(Example may not include actual data)

• Data duplication/redundancy (storing of the same data more than once) is likely to occur if a database is not normalised.

Example: the manufacturer M68 has name (Dragon) and contact (Phillips) and is stored three (or more) times in the database, whereas the name and contact need only to be stored once.

• Update anomaly (term accepted not expected): if some data is updated, there may be a need to update other related data a number of times causing a risk of inconsistent data.

Example: if the contact at the manufacturer Dragon changes from Phillips to (say) Smith, then the name Phillips needs replacing with Smith three (or more) times in the database. This increases the chance of incorrect data being included.

• Delete anomaly (term accepted not expected): if some data is deleted, the unwanted side effect may be that other data is lost.

Example: if the data relating to Deluxe Helmet (A7129) is deleted, we also lose all data about manufacturer Safe Heads.</td><td>2
2</td><td>1.1b</td><td>
2.1a</td><td></td><td>4</td></tr>
</table>

Q	Answer	Marks	AO1	AO2	AO3	Total
5b	ITEM (ItemCode, ItemName, ManufCode) SHOP (ShopCode, ShopAddress, ShopManager) MANUFACTURER (ManufCode, ManufName, ManufContact) ITEMS-SHOPS (ItemCode, ShopCode, NoInStock) **Marking** • Four suitably named tables • Each table with a suitable primary key shown (4 correct → 2 marks 2 or 3 correct → 1 mark) • 1 mark for each correct foreign key x 3	 1 2 3		 2.1b 2.1b 2.1b		6
5c	• (Databases often contain huge amounts of data.) It is often more efficient to store data on a number of different computers (probably in different locations) to maximise performance. **Not:** improves security **Not:** will still work if one computer fails etc • It is difficult to ensure that all the data in all the computers is always up-to-date / maintain integrity. • Both processing and data are distributed across the different computers that the data is stored upon.	1 1 1	1.1b 1.1b 1.1b			3
6a	`SELECT PupName, TeachNum FROM PUPIL`	1			3.1b	1
6b	`SELECT PupName FROM PUPIL WHERE TeachNum = '307'`	1			3.1b	1
6c	`UPDATE PUPIL SET TeachNum = '345' WHERE TeachNum = '378'` 1 mark for update; 1 mark for changing TeachNum	2			3.1b	2
6d	`SELECT PupName FROM PUPIL WHERE TeachNum = (SELECT TeachNum FROM PUPIL WHERE PupNum = '14238')` 1 mark for each select (1 mark for main select and one from sub-select)	2			3.1b	2

6e	`CREATE TABLE COMPUTER (`
	`RoomNum Char(5)`
	`SerialNum Char(5)`
	`)`
	
	1 mark for table; 1 mark for fields (any suitable field size acceptable)
		2			3.1b	2

6f	`INSERT INTO COMPUTER VALUES ('106' , '13457')`
	`INSERT INTO COMPUTER VALUES ('113' , '66870')`
	
	1 mark for insert; 1 mark for values inserted
		2			3.1b	2

Q	Answers	Marks	AO1	AO2	AO3	Total
7ai	11111001_2 00000100_2 + **11111101_2**	1 1		2.1a 2.1a		2
7aii	00001100_2 (mark for negating the 5_{10}) 11111011_2 + (mark for addition operation) **00000111_2**	1 1 1		2.1a 2.1a 2.1a		3
7bi	Mantissa = $00000101 . 1010_2$ Exponent = 0011_2 Normalised: 0.10110100000_2 x 2^{0011} or 0.703125_{10} x 2^3	1 1 1		2.1a 2.1a 2.1a		3
7bii	Mantissa = 0.9375_{10} Exponent = 3_{10} Decimal Equivalent: 0.9375_{10} x 2^3 = 7.5_{10}	1 1 1		2.1a 2.1a 2.1a		3
7biii	Mantissa = 0.859375_{10} Exponent = 3_{10} Decimal Equivalent: 0.8595_{10} x 2^3 = 6.875_{10} **Absolute error** 0.025 / -0.025 / 6.9-6.875 / 1/40 **Relative error** 0.00362319 / 0.025/6.9 / 1/276 = 0.362319% **Modifications (any one of)** • Adjust the mantissa to use more bits / Accept similar wording, e.g. longer mantissa • Reallocate (one) bit; from the exponent to the mantissa • Infer one of the two bits on either side of the binary point and use the freed up bit to store one more significant digit in the mantissa	1 1 1 1 1 1		2.1a 2.1a 2.1a 2.1b 2.1b 2.1b		6
7ci	00111100_2 The effect is multiplying by 4 (i.e. by 100_2 - needs subscript) **Not** gets bigger **Not** "Moves two places to left"	1 1		2.1a 2.1b		2
7cii	(Numeric) **overflow** (number becomes (~~1~~) 00111100) The resulting number is too large to be contained in the eight bits available	1 1		2.1a 2.1a		2
8	• **Big Data** refers to data sets so large and complex that it becomes difficult to process using standard relational database techniques • **Data mining** the analysis of (a large amount of) data (in a data warehouse) • **Predictive analytics** consist of a variety of statistical techniques including modelling, machine learning, and data mining. Example: In business, predictive models analyses patterns found in historical and transactional data to identify patterns that may present risks or opportunities.	1 1 1 1	1.1b 1.1b 1.1b 1.1b			4

Q	Answer	Marks	AO1	AO2	AO3	Total
9a	• More than one job in memory at same time • More than one job processed (apparently) at same time • Time-slice is amount of time allocated to each job by operating system • Scheduling decides which job is to be processed next (may be prioritised) • Partitioning is division of computer memory for different jobs • Paging jobs in and out to make better use of memory • Promotes efficient use of CPU • Achieved by use of interrupts **Marking** • Any point can be extended for an additional mark • Max 6 for brief points only • Max 6 if not covered all four of multiprogramming, time-slicing, scheduling and memory partitioning	1 1 1 1 1 1 1 1	1.1b 1.1b 1.1b 1.1b 1.1b 1.1b 1.1b 1.1b			8
9bi	An interrupt is a signal generated by a device or software, which may cause a break in the execution of the current routine. NOTE – do not accept an interrupt is generated when there is a fault (this is an example not a description) Examples (any three of) • Hardware or Software fault • Input/output device requesting attention e.g. printer out of paper / requesting more data / key press / mouse click • User interrupt e.g. <ctrl> <break> • Operating system generated interrupt e.g. end of time slice • Run time error e.g. division by zero	1 3	1.1b 1.1b			4
9bii	Answers **MUST** be in this order: 1. O/S suspends current interrupt routine 2. Runs the new higher priority interrupt routine 3. Finally, the O/S returns to original interrupt routine and continues	1 1 1	1.1b 1.1b 1.1b			3
10	Any three of: • No possibility of human error (for instance passing a signal at red) • Train's control system could apply the brakes at/before a red signal • Obstruction / train ahead detection could be included • Could govern the maximum speed • Could prevent starting with any doors open • No driver so save salaries, plus no sickness, lateness etc. • Computer may apply optimisation algorithms to ensure trains reach their stations as efficiently as possible (e.g. on time and regular)	3		2.1a		3

Q	Answer	Marks	AO1	AO2	AO3	Total
11	Need for • Cryptography is needed in computer systems, both online and locally to protect the information stored on individual devices and whilst being transmitted over computer networks The purpose of • Cryptography is to encode messages or information in such a way that only parties with the appropriate decryption key can read it • The message or information, referred to as plaintext, is encrypted using an encryption algorithm and a key, generating cipher text that can only be read if decrypted using the original key	1 1 1	1.1b 1.1b 1.1b			3

12	**Indicative content**
	• Parallel computing is a form of computation in which many calculations are carried out simultaneously
	• It operates on the principle that large problems can often be divided into smaller ones, which are then solved concurrently
	• Parallel computer programs are more complex to design and to write than sequential ones
	• Concurrency introduces several new classes of potential software bugs
	• Race conditions are the most common class of potential software bug
	• Communication and synchronisation between the different subtasks are typically some of the greatest obstacles to getting efficient parallel program performance
	• The maximum possible speed-up of a single program as a result of parallelisation is known as Amdahl's law:
	  ○ $T(n) = T(1)(B + \frac{1}{n}(1 - B))$
	  ○ $Where$:
	    ▪ $T(n) = time\ taken\ on\ n\ threads$
	    ▪ $n = number\ of\ threads$
	    ▪ $B = fraction\ of\ algorithm\ that\ is\ sequential$
	• Example: If a program needs 10 hours using a single processor core, and a particular portion of the program which takes one hour to execute cannot be parallelised, while the remaining 9 hours (90%) of execution time can be parallelised, then regardless of how many processors are devoted to a parallelised execution of this program, the minimum execution time cannot be less than that critical one hour.
	• Using the formula above with one thread (n=1) we get:
	  ○ $T(n) = T(1)(B + \frac{1}{n}(1 - B)$
	  ○ $T(1) = T(1)(B + \frac{1}{1}(1 - B)$
	  ○ $T(1) = T(1)(0.9 + (1 - 0.9) = 1\ hour\ minimum$
	• The speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program
		10	1.1b			10

Band	**AO1.1b** **Max 10 marks**
3	**8 - 10 marks** The candidate has: • written an extended response that has a sustained line of reasoning which is coherent, relevant, and logically structured • shown clear understanding of the requirements of the question and a clear knowledge of the indicative content. Clear knowledge is defined as a response that provides eight to ten relevant detailed points on the limiting factors to parallelisation in parallel processing, which relate to an extensive amount of the indicative content • addressed the question appropriately with minimal repetition and no irrelevant material • has presented a balanced discussion and justified their answer with examples • used appropriate technical terminology referring to the indicative content confidently and accurately.
2	**4 - 7 marks** The candidate has: • written a response that has an adequate line of reasoning with elements of coherence, relevance, and logical structure • shown adequate understanding of the requirements of the question and a satisfactory knowledge of the indicative content. Satisfactory knowledge is defined as a response that provides four to seven points on the limiting factors to parallelisation in parallel processing as signalled in the indicative content. • has presented a discussion with limited examples • used appropriate technical terminology referring to the indicative content.
1	**1 - 3 marks** The candidate has: • written a response that that lacks sufficient reasoning and structure • produced a discussion which is not well developed • attempted to address the question but has demonstrated superficial knowledge of the topics specified in the indicative content. Superficial knowledge is defined as a response that provides one to three points on the limiting factors to parallelisation in parallel processing as signalled in the indicative content • used limited technical terminology referring to the indicative content.
0	**0 marks** Response not credit worthy or not attempted.

UNIT 1

Coverage of Assessment Objectives

Assessment Objective		Elements	Question 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
AO1	**Demonstrate knowledge and understanding of the principles and concepts of computer science, including abstraction, logic, algorithms and data representation**	a – Demonstrate knowledge of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate				1	2											**3**
		b – Demonstrate understanding of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate	6	6	3		2	7	6	3		2			6	8	8	**57**
		TOTAL AO1	**6**	**6**	**3**	**1**	**4**	**7**	**6**	**3**	**0**	**2**	**0**	**0**	**6**	**8**	**8**	**60**
AO2	**Apply knowledge and understanding of the principles and concepts of computer science, including to analyse problems in computational terms**	a – Apply knowledge and understanding of the principles and concepts of computer science			7	2				9	6	3	5					**32**
		b – Analyse problems in computational terms																**0**
		TOTAL AO2	**0**	**0**	**7**	**2**	**0**	**0**	**0**	**9**	**6**	**3**	**5**	**0**	**0**	**0**	**0**	**32**
AO3	**Design, program and evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions**	a – Design computer systems that solve problems																**0**
		b – Program computer systems that solve problems												6				**6**
		c – Evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions										2						**2**
		TOTAL AO3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**2**	**0**	**6**	**0**	**0**	**0**	**8**
		TOTAL AO1 + AO2 + AO3	**6**	**6**	**10**	**3**	**4**	**7**	**6**	**12**	**6**	**7**	**5**	**6**	**6**	**8**	**8**	**100**

UNIT 2

Coverage of Assessment Objectives

Assessment Objective		Elements	Question									Total
			Section: A						B			
			1	2	3	4	5	6	1	2	3	
AO1	Demonstrate knowledge and understanding of the principles and concepts of computer science, including abstraction, logic, algorithms and data representation	a – Demonstrate knowledge of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate										0
		b – Demonstrate understanding of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate										0
		TOTAL AO1	0	0	0	0	0	0	0	0	0	0
AO2	Apply knowledge and understanding of the principles and concepts of computer science, including to analyse problems in computational terms	a – Apply knowledge and understanding of the principles and concepts of computer science										0
		b – Analyse problems in computational terms	10	6	6	6	8					36
		TOTAL AO2	10	6	6	6	8	0	0	0	0	36
AO3	Design, program and evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions	a – Design computer systems that solve problems									4	4
		b – Program computer systems that solve problems						8	4	8		20
		c – Evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions										0
		TOTAL AO3	0	0	0	0	0	8	4	8	4	24
		TOTAL AO1 + AO2 + AO3	10	6	6	6	8	8	4	8	4	60

UNIT 3

Coverage of Assessment Objectives

Assessment Objective		Elements	Question 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
AO1	**Demonstrate knowledge and understanding of the principles and concepts of computer science, including abstraction, logic, algorithms and data representation**	a– Demonstrate knowledge of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate							1									**1**
		b – Demonstrate understanding of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate		4	6		1		2	1		6	4	12	13			**49**
		TOTAL AO1	**0**	**4**	**6**	**0**	**1**	**0**	**3**	**1**	**0**	**6**	**4**	**12**	**13**	**0**	**0**	**50**
AO2	**Apply knowledge and understanding of the principles and concepts of computer science, including to analyse problems in computational terms**	a – Apply knowledge and understanding of the principles and concepts of computer science	5			9	2		1		1							**18**
		b – Analyse problems in computational terms	9	3						7								**19**
		TOTAL AO2	**14**	**3**	**0**	**9**	**2**	**0**	**1**	**7**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**37**
AO3	**Design, program and evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions**	a – Design computer systems that solve problems																**0**
		b – Program computer systems that solve problems						7										**7**
		c – Evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions									6							**6**
		TOTAL AO3	**0**	**0**	**0**	**0**	**0**	**7**	**0**	**0**	**6**	**0**	**0**	**0**	**0**	**0**	**0**	**13**
		TOTAL AO1 + AO2 + AO3	**14**	**7**	**6**	**9**	**3**	**7**	**4**	**8**	**7**	**6**	**4**	**12**	**13**	**0**	**0**	**100**

UNIT 4

Coverage of Assessment Objectives

Assessment Objective		Elements	Question															Total
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
AO1	**Demonstrate knowledge and understanding of the principles and concepts of computer science, including abstraction, logic, algorithms and data representation**	a – Demonstrate knowledge of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate																**0**
		b – Demonstrate understanding of the principles and concepts of abstraction, logic, algorithms, data representation or others as appropriate	3	4	6		5			4	15		3	10				**50**
		TOTAL AO1	**3**	**4**	**6**	**0**	**5**	**0**	**0**	**4**	**15**	**0**	**3**	**10**	**0**	**0**	**0**	**50**
AO2	**Apply knowledge and understanding of the principles and concepts of computer science, including to analyse problems in computational terms**	a – Apply knowledge and understanding of the principles and concepts of computer science		2		3	2		17			3						**27**
		b – Analyse problems in computational terms					6		4									**10**
		TOTAL AO2	**0**	**2**	**0**	**3**	**8**	**0**	**21**	**0**	**0**	**3**	**0**	**0**	**0**	**0**	**0**	**37**
AO3	**Design, program and evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions**	a – Design computer systems that solve problems																**0**
		b – Program computer systems that solve problems	3					10										**13**
		c – Evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions																**0**
		TOTAL AO3	**3**	**0**	**0**	**0**	**0**	**10**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**13**
		TOTAL AO1 + AO2 + AO3	**6**	**6**	**6**	**3**	**13**	**10**	**21**	**4**	**15**	**3**	**3**	**10**	**0**	**0**	**0**	**100**