Component 1 – Programming and System Development

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.

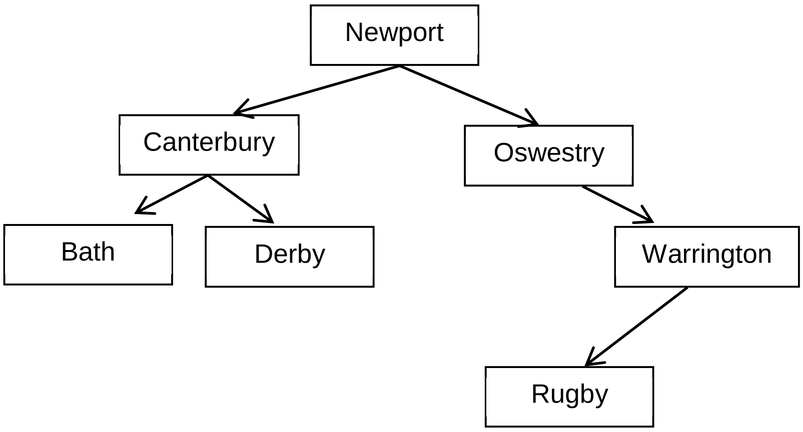
If an answer covers different aspects of different bands within the mark scheme, a 'best fit' approach should be adopted to decide on the band and then the learner's response should be used to decide on the mark within the band. For instance if a response is mainly in band 2 but with a limited amount 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.

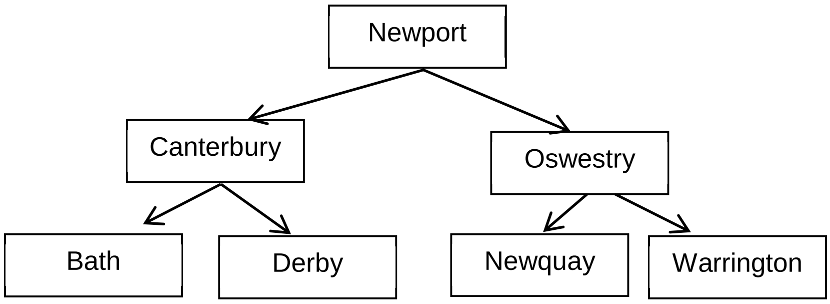
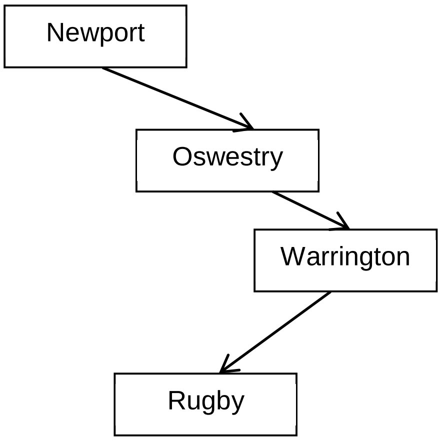
Indicative content is also provided for banded mark schemes. Indicative content is not exhaustive, and any other valid points must be credited. In order to reach the highest bands of the mark scheme a learner need not cover all of the points mentioned in the indicative content but must meet the requirements 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.

<table>
<tr><th>Qu</th><th>Answer</th><th>Mark</th><th>AO1</th><th>AO2</th><th>AO3</th><th>TOT</th></tr>
<tr><td>1
(a)</td><td>


Marking
1 mark for correct root
1 mark for ALL left pointers correct and ALL right pointers correct</td><td>2</td><td></td><td>2.1b</td><td></td><td>2</td></tr>
<tr><td>1
(b)</td><td>
<table>
<tr><th></th><th>Left Pointer</th><th>Data</th><th>Right Pointer</th></tr>
<tr><td>Start 0</td><td>1</td><td>Newport</td><td>2</td></tr>
<tr><td>1</td><td>5</td><td>Canterbury</td><td>6</td></tr>
<tr><td>2</td><td>-1</td><td>Oswestry</td><td>3</td></tr>
<tr><td>3</td><td>4</td><td>Warrington</td><td>-1</td></tr>
<tr><td>4</td><td>-1</td><td>Rugby</td><td>-1</td></tr>
<tr><td>5</td><td>-1</td><td>Bath</td><td>-1</td></tr>
<tr><td>6</td><td>-1</td><td>Derby</td><td>-1</td></tr>
<tr><td>7</td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td></tr>
</table>

Marking
1 mark for correct root with start pointer
1 mark for ALL left pointers correct
1 mark for ALL right pointers correct</td><td>3</td><td></td><td>2.1a</td><td></td><td>3</td></tr>
</table>

<table>
<tr><th>Qu</th><th>Answer</th><th>Mark</th><th>AO1</th><th>AO2</th><th>AO3</th><th>TOT</th></tr>
<tr><td>1
(c)(i)</td><td>
<table>
<tr><th></th><th>Left Pointer</th><th>Data</th><th>Right Pointer</th></tr>
<tr><td>Start 0</td><td>1</td><td>Newport</td><td>2</td></tr>
<tr><td>1</td><td>5</td><td>Canterbury</td><td>6</td></tr>
<tr><td>2</td><td>7</td><td>Oswestry</td><td>3</td></tr>
<tr><td>3</td><td>4</td><td>Warrington</td><td>-1</td></tr>
<tr><td>4</td><td>-1</td><td>Rugby</td><td>-1</td></tr>
<tr><td>5</td><td>-1</td><td>Bath</td><td>-1</td></tr>
<tr><td>6</td><td>-1</td><td>Derby</td><td>-1</td></tr>
<tr><td>7</td><td>-1</td><td>Newquay</td><td>-1</td></tr>
<tr><td>8</td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td></tr>
</table>
Marking
1 mark for Newquay in correct location 7
1 mark for left pointer of Oswestry set to 7
1 mark for left and right pointers of Newquay set to -1
</td><td>3</td><td></td><td>2.1b</td><td></td><td>3</td></tr>
<tr><td>1
(c)
(ii)</td><td>
<table>
<tr><th></th><th>Left Pointer</th><th>Data</th><th>Right Pointer</th></tr>
<tr><td>Start 0</td><td>1</td><td>Newport</td><td>2</td></tr>
<tr><td>1</td><td>5</td><td>Canterbury</td><td>6</td></tr>
<tr><td>2</td><td>-1</td><td>Oswestry</td><td>~~3~~ 4</td></tr>
<tr><td>3</td><td>4</td><td>Warrington</td><td>-1</td></tr>
<tr><td>4</td><td>-1</td><td>Rugby</td><td>-1</td></tr>
<tr><td>5</td><td>-1</td><td>Bath</td><td>-1</td></tr>
<tr><td>6</td><td>-1</td><td>Derby</td><td>-1</td></tr>
<tr><td>7</td><td>-1</td><td>Newquay</td><td>-1</td></tr>
<tr><td>8</td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td></tr>
</table>
Marking
1 mark for right pointer of Oswestry set to point at Rugby 4
</td><td>1</td><td></td><td>2.1b</td><td></td><td>1</td></tr>
<tr><td>1
(d)</td><td>Advantage is that access to data is generally faster or data can be retrieved in different useful orders such as ‘in order’ or ‘post order’
Disadvantage is that there is the overhead of two pointers and the access algorithm is more complicated</td><td>1
1</td><td>1.1b
1.1b</td><td></td><td></td><td>2</td></tr>
</table>

Qu	Answer	Mark	AO1	AO2	AO3	TOT
1 *(e)*	 Newport Canterbury Oswestry Bath Derby Newquay Warrington The above tree is balanced and the maximum number of comparisons to locate an item would be the same as the number of levels, 3 in this case  The above tree is unbalanced and the maximum number of comparisons to locate an item would be the same as the number of items, 4 in this case Marking					4
	1 mark for example of each tree – balanced and unbalanced	2		2.1a		
	1 mark for each description of number of comparisons	2		2.1b		

Qu	Answer	Mark	AO1	AO2	AO3	TOT
2	One mark for each valid reason for following a specific rule, up to a maximum of five marks. No marks for simply stating rules, as question requires reasons for following rules. If rule is implicit within reasoning, award mark for response (e.g. 'data must be kept safe **to prevent data from being stolen that could cause an individual embarrassment'**) – Security rule implied. Indicative content (bold indicates example valid reasoning within a summary, other valid reasoning accepted, if justified) • Programmers should have due regard for public health, privacy, security and wellbeing of others and the environment. **This will ensure that no harm physical, emotional or financial comes to an individual from not taking these factors into account (could also be reasoned with the use of an appropriate example; for instance in the case of privacy, data could be stolen that could cause an individual embarrassment)** • Programmers should have due regard for the legitimate rights of any person or organisation that might be affected by their activities. **This will ensure that the rights of others are respected and no harm comes to the public (could also be reasoned with the use of an appropriate example; for instance the right of an individual that data held is only used for an agreed purpose and not abused)** • Programmers should conduct their professional activities without discrimination on the any grounds. **This will ensure that no individual is denied their rights (could also be reasoned with the use of an appropriate example; for instance that software is carefully designed to consider other groups' needs, for example accessibility for disabled people)** • Programmers should promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society wherever opportunities arise. **This will ensure that there is no technology gap between sectors in society (could also be reasoned with the use of an appropriate example; for instance that a deprived community has opportunity access to the same website as an affluent group)** • Programmers should not claim any level of competence that they do not possess. **This safeguards an employer placing a programmer on a task that could not be completed or would be completed with significant errors which would waste time or money. (could also be reasoned with the use of an appropriate example; for instance a programmer claiming that they could use a given language but could not and then could not write the necessary program)** • Programmers should develop their professional knowledge, skills and competence on a continuing basis, maintaining awareness of technological developments, procedures, and standards that are relevant to their field. **This ensures that the product produced by a programmer is up-to-date and will function in contemporary systems (could also be reasoned with the use of an appropriate example; for instance ensuring that a programmer writes software that will function on a new operating system)** • Programmers should ensure that they have the knowledge and	5	1.1b			5

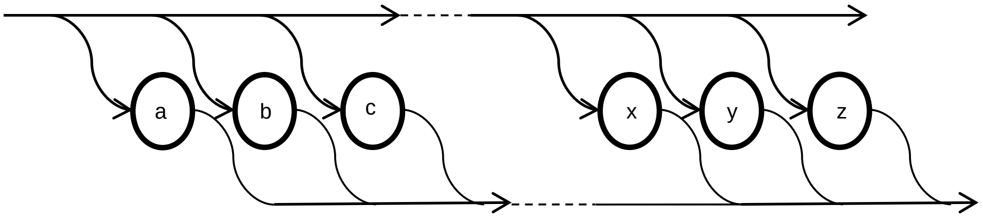
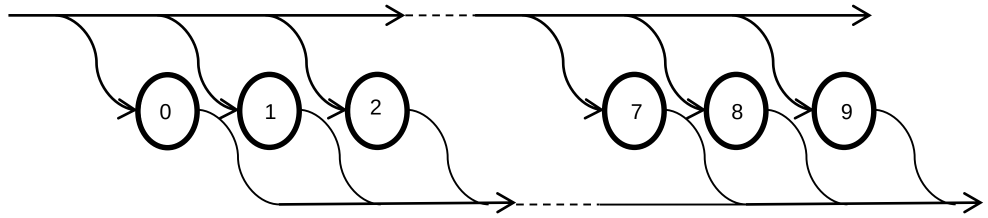
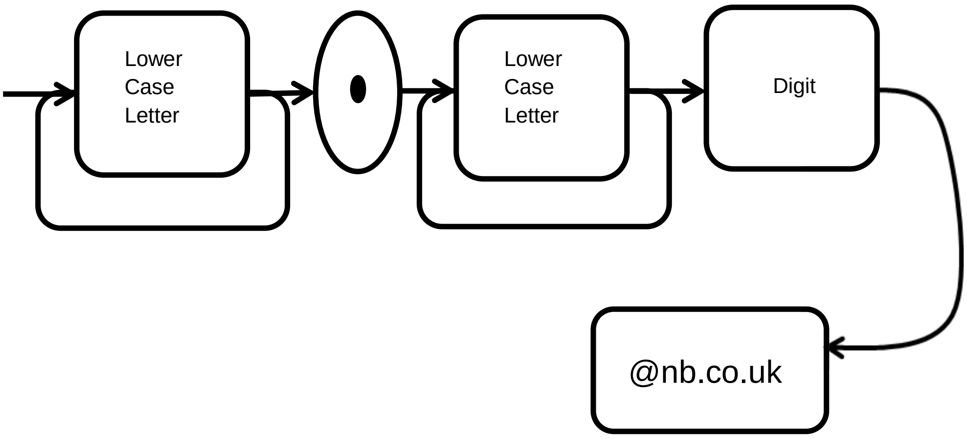
Qu	Answer	Mark	AO1	AO2	AO3	TOT
	understanding of legislation and that they comply with such legislation, in carrying out their professional responsibilities. **This ensures that the programmer does not unwittingly break the law when undertaking their day to day job which could cause embarrassment or losses (could also be reasoned with the use of an appropriate example; for instance developing insecure software that breaches data protection laws)** • Programmers should respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work. **This ensures that all relevant approached and options are considered, and the best one chosen (could also be reasoned with the use of an appropriate example; for instance when developing a user interface all opinions should be considered and the best design used)** • Programmers should avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction. **This ensures that staff are aware that they should consider others before taking action and do not take risks that could injure others (could also be reasoned with the use of an appropriate example; for instance programmers should avoid altering a program that may lose work for others)** • Programmers should reject and not make any offer of bribery or unethical inducement. **This ensures that staff are not open to corruption from others and take actions that could harm a company or client (could also be reasoned with the use of an appropriate example; for instance programmers should not disclose sensitive data if offered an incentive to do so)** • Programmers should carry out their professional responsibilities with due care and diligence in accordance with the employer or client's requirements whilst exercising professional judgement at all times. **This would ensure that programs are developed in line with a client's requirements and that time/money is not wasted in developing other, unrequired areas (could also be reasoned with the use of an appropriate example for instance a programmer should let an employer know if a certain methodology is not working and advise on methods of changing methodology)** • Programmers should seek to avoid any situation that may give rise to a conflict of interest between them and their employer or client. **This would ensure that a programmer does not have conflicting tasks that may result in one not being completed properly (could also be reasoned with the use of an appropriate example; for instance that a programmer should not embark on a personal programming project that competes with that commissioned by a client.)** • Programmers should accept professional responsibility for their work and for the work of colleagues who are defined in a given context as working under their supervision. **This gives ownership of work, and with this, less chance of neglecting the work as the programmer is directly responsible. (could also be reasoned with the use of an appropriate example; for instance if a programmer has a set role in a task, they are likely to feel that they own that task and are more likely to do that task to the best of their ability)** • Programmers should not disclose or authorise to be disclosed, or use for personal gain or to benefit a third party, confidential information except with the permission of their employer or client, or as required by legislation. **This would undermine a client, and possibly result in**					

Qu	Answer	Mark	AO1	AO2	AO3	TOT
	loss if a competitor were to develop a product based on information disclosed. (could also be reasoned with the use of an appropriate example; for instance this prevents a programmer from selling information on a product to a company developing a similar product) • Programmers should not misrepresent or withhold information on the performance of products, systems or services (unless lawfully permitted to do so by a duty of confidentiality) or take advantage of the lack of relevant knowledge or inexperience of others. **This prevents making financial or other gain from overstating the work required for a given task. (could also be reasoned with the use of an appropriate example; for instance could mean that a programmer could charge more money by stating that a simple task took longer to complete than it actually did)** • Programmers should accept their personal duty to uphold the reputation of the profession and not take any action which could bring the profession into disrepute. **This ensures that the profession is not seen negatively by the wider public and not undermined by a lack of trust. (could also be reasoned with the use of an appropriate example – many potential examples)** • Programmers should encourage and support fellow members in their professional development. **This ensures that fellow members are able to support their team in development and that individuals are not undermined or lose out as a result of a lack of knowledge. (could also be reasoned with the use of an appropriate example – many potential examples)** • Programmers seek to improve professional standards through participation in their development, use and enforcement. **This ensures that programmers have ownership of the standards and these standards are more likely to be relevant to programmers as a result. (could also be reasoned with the use of an appropriate example, for instance if there were a new programming certification, the fact that programmers were part of its development would give the certification more status)** • Programmers notify the employer if convicted of a criminal offence. **This ensures that an employer can judge if a programmer can continue in their role as there may be risks if the crime is relevant to their work. (could also be reasoned with the use of an appropriate example – for instance if convicted of fraud, a programmer would not be permitted to program financial systems)**					

<table>
<tr><th>Qu</th><th>Answer</th><th>Mark</th><th>AO1</th><th>AO2</th><th>AO3</th><th>TOT</th></tr>
<tr><td>3</td><td>Example lookup tables used by the compiler during translation of this language are:
One mark for all reserved words and symbols
One mark for unique hex tokens
<table>
<tr><th>Reserved word</th><th>Token (Hex)</th></tr>
<tr><td>print</td><td>2A</td></tr>
<tr><td>=</td><td>2E</td></tr>
<tr><td>+</td><td>2F</td></tr>
<tr><td>;</td><td>30</td></tr>
<tr><td>(</td><td>31</td></tr>
<tr><td>)</td><td>32</td></tr>
</table></td><td>1
1</td><td></td><td>2.1b
2.1b</td><td></td><td>5</td></tr>
<tr><td></td><td>One mark for all user identifiers
One mark for unique hex tokens
<table>
<tr><th>Used identifier</th><th>Type</th><th>Token (Hex)</th></tr>
<tr><td>NumInput</td><td>Integer</td><td>7A</td></tr>
<tr><td>Total</td><td>Integer</td><td>7B</td></tr>
<tr><td>Count</td><td>Integer</td><td>7C</td></tr>
</table></td><td>1
1</td><td></td><td>2.1b
2.1b</td><td></td><td></td></tr>
<tr><td></td><td>Example of output stream:
7B 2E 7B 2F 7A (accept answer with spaces between tokens or without spaces)</td><td>1</td><td></td><td>2.1a</td><td></td><td></td></tr>
<tr><td>4
(a)</td><td>One possible solution is:
$A . B + A$
$\bar{A} + \overline{B} + A$ [$\overline{A . B} = \bar{A} + B$ De Morgan]
$\overline{B} + 1$ [$\bar{A} + A \rightarrow 1$]
1 [$\overline{B} + 1 \rightarrow 1$]
$\overline{A . B} + A = 1$
Marking
Correctly applying De Morgan’s law – 1 mark
Correctly applying identities to arrive at correct answer – 2 marks
Correctly applying identities but arriving at incorrect answer - 1 mark</td><td>3</td><td></td><td>2.1a</td><td></td><td>3</td></tr>
</table>

Qu	Answer	Mark	AO1	AO2	AO3	TOT
4 *(b)*	Correct answer can be established using different steps / laws / rules / identities / dual relations One possible solution is: A + B.(A + B) + A.(Ā + B) A + B.A + B.B + A.(Ā + B) [distributive law] A + B.A + B + A.(Ā + B) [B.B → B] A + B + A.(Ā + B) [B.A + B → B] A + B + A.Ā + A.B [distributive law] A + B + A + A.B [A.Ā → A] A + B + A [A.B → A] A + B + A [A.B → A] A + A + B [commutative law] A + B [A + A → A] Alternatively A + B.(A + B) + A.(Ā + B) A + B + A.(Ā + B) [B.(A + B) → B] A + B + A.B [A.(Ā + B) → A.B] A + A.B + B [commutative law] A + B [A + A.B → A] Marking Correctly applying rules to arrive at correct answer – 5 marks Correctly applying rules but arriving at incorrect answer then one mark per correct rule applied – max 4 marks	5		2.1a		5
5 *(a)*	It is good programming practice to use constants, such as VAT = 20%, because they can be set once and then used many times throughout the program and if they change then they only have to be changed once. It is good programming practice to use meaningful names for variables because it is then clear what the variable is holding and aids program readability. It is good programming practice to use annotation because other programmers will find the code easier to follow.	1 1 1	1.1b 1.1b 1.1b			3
5 *(b)*	Value = R Reference = A Both must be correct for one mark	1		2.1a		1
5 *(c)*	Value – a copy of the data is passed into the procedure and cannot be altered Which has the advantage of avoiding unwanted side effects like the value being changed by mistake Reference – a copy of the memory address is passed into the procedure therefore the data in the calling environment can be altered	1 1 1	1.1b 1.1b 1.1b			3

Qu	Answer	Mark	AO1	AO2	AO3	TOT
6	Explanation including four from the points below: A debugger is a software tool used to detect, locate and correct faults in a program and will permit an individual: • to see the progress through the program - which statements / procedures are being executed at any time using program trace (step through / step-into) • to temporarily halt execution at a fixed point in the code in order to ascertain the value of variables at that point (or to step through the program from that point) using a break point • to temporarily to halt execution when a condition is satisfied in order to ascertain the value of variables at that point using a conditional break point • to view the value of a variable at specific points during the execution using variable watch • to view the entire contents of memory at a specific point using store dump • to view error messages relating to errors in the program using error diagnostics • to see the values of variables at the point where the program failed using post-mortem routines. Any other facilities that assist debugging if explained accurately may be worthy of credit.	4	1.1b			4

Qu	Answer	Mark	AO1	AO2	AO3	TOT
7 *(a)*	Lower case Letter  Digit  Email address  Lower Case Letter Lower Case Letter Digit @nb.co.uk Marking of email address One mark for Lower Case Letter repeated twice Two marks for correct sequence of Lower Case Letter, dot, Lower case Letter and digit One mark if mistake in correct sequence (for example dot missing or in incorrect sequence)	1 1 3		2.1b 2.1b 2.1b		5
7 *(b)*	<letter> ::= a|b|c . . . |z <digit> ::= 0|1|2 . . . |9 <name> ::= <letter>|<letter><name> <email_ad> ::= <name>.<name><digit>@nb.co.uk Marking One mark for recursion: Same item Left and Right are needed Cannot gain full marks unless completely correct	1 1 1 1		2.1b 2.1b 2.1b 2.1b		4

Qu	Answer	Mark	AO1	AO2	AO3	TOT
7 *(c)*	The syntax of computer languages follow strict unambiguous rules but the English language can be ambiguous	1	1.1b			3
	One mark for both comparison to syntax diagrams and BNF					
	Syntax diagrams provide a diagrammatic representation which are easy to follow	1	1.1b			
	BNF definitions can easily be coded when writing a compiler	1	1.1b			
8	Perfective is when the system functionality is improved	1	1.1a			6
	For example reducing the search time for stock control records	1		2.1a		
	Adaptive is when the system is changed to run on a new hardware or software platform	1	1.1a			
	For example the retail company could switch from a Windows based system to Macintosh or Linux	1		2.1a		
	Corrective is fixing bugs in the system that were not detected during development or system testing	1	1.1a			
	For example the users might find that some data input for new stock is not validated and causes program to crash or produce inaccurate output	1		2.1a		
9 *(a)*	A recursive algorithm is one:					2
	That calls itself	1	1.1a			
	And has a terminating (base) condition to stop the calling	1	1.1a			
9 *(b)*	Could run out of memory (stack space)	1	1.1a			4
	due to high number of recursive calls	1	1.1b			
	Difficult to debug (dry run) if producing incorrect results	1	1.1a			
	due to difficulty in determining which recursive call produced the error	1	1.1b			
10	A class is a template defining methods and attributes used to make objects	1	1.1a			3
	Inheritance is when a new class is created that is based on another class and	1	1.1a			
	It can take on the methods and attributes of the parent class	1	1.1a			
11 *(a)*	Input 10 integers into an array	1			3.1b	3
	Correct control structure – repetition using a loop	1			3.1b	
	Identifying (selection using IF) and outputting largest value	1			3.1b	
11 *(b)*	Sort the array of 10 integers into ascending order Output the 10 integers in ascending order Marking of the sort Correct control structures (nested loops) – 2 marks Comparing values for ascending sort - 1 mark Successful swapping of elements – 1 mark Algorithm terminates correctly – 1 mark Algorithm deals with identical values – 1 mark	6			3.1b	6

Qu	Answer	Mark	AO1	AO2	AO3	TOT
12 *(a)*	One mark for both correct $10^3 = 1000$ $100^3 = 1\ 000\ 000$	1		2.1a		1
12 *(b)*	Evaluation of algorithm Multiplication The only multiplication appears in the k loop. Since this loop will iterate a total of n^3 times, it will execute exactly n^3 multiplication operations Addition There are two addition operations in the k loop: (I) Answer(i, j) = Answer(i, j) + A(i, k) * B(k, j) (II) Incrementing k Each of these will happen n^3 times. Therefore there are $2n^3$ additions generated by the k loop The j loop has one addition operation Incrementing j which happens n^2 times The i loop has one addition operation as well Incrementing i which happens n times. Adding these up we the number of addition operations which is $2n^3 + n^2 + n$ As n gets very big then n^3 will dominate therefore it is $O(n^3)$ NOTE: Calculations might include assignment operations but these will not affect overall time so ignore **Marking** One mark for identifying i loop will execute n times One mark for identifying j loop will execute n^2 times One mark for identifying that k loop will execute n^3 times. One mark for correct number of additions $2n^3 + n^2 + n$ One mark for determining that the order will be dominated by n^3 as n gets very big giving $O(n^3)$ for the algorithm	5			3.1c	5
12 *(c)*	Algorithm will need to store 3 matrices A, B and Answer Each matrix will require N x N items = N^2 Total storage will be 3 x N x N = $3N^2$ As n increases the storage requirements will increase n^2 as constant (3) will be insignificant so storage requirements will be Order (n^2) **Marking** One mark for identifying storage requirements as $3N^2$ One mark for determining the storage requirements will be $O(n^2)$	2			3.1c	2

Qu	Answer	Mark	AO1	AO2	AO3	TOT
13	**Indicative content** When developing a system in modules then it is desirable for them to be highly cohesive and loosely coupled **Algorithms and programs** Use of constants – allow modules to be easily updated, for example when the rate of VAT changes a module that calculates Gross Price can be updated by only changing the constant value for the rate of VAT and likewise fir income tax, NI contributions, pensions, etc… Use of standard functions that have already been written can easily be integrated into a modular system Use of meaningful identifiers are essential for team members to be able to understand and amend modules written by other programmers Use of local variables that are not used outside the module will help ensure the system is highly cohesive and loosely coupled Use of global variables should be avoided as will make the system 'tightly coupled' Use of parameters will help ensure the system is highly cohesive and loosely coupled All variables and parameters should be clearly documented for future and other programmers Modules can be tested individually before integration Ensure there is data independence so underlying data structures can be altered without affecting existing modules **Principles of programming** Many different computer languages facilitate modular programming A class in OO languages can be considered a module Modular programming is suitable for bottom up approach to system development Modular programming is suitable for a structured top down approach to system development (stepwise refinement) Modules must be written to agreed standards to allow easy integration Modules must be written to agreed standards to allow system to be maintained	12	1.1b			12

	Systems analysis Modular programming is suitable for both agile and waterfall approach to system development Modules can be integrated into system one module at a time Modules can easily be replaced with improved modules Modules can easily be replaced with modules with additional or different functionality Existing modules can be re-used Modules have been tried and tested so less chance of future errors Different programmers can work on different modules at the same time so the critical time path is shorter Programmers with expertise in specific areas can write modules in their specialised area **Software engineering** Integrated Development Environment can be used to debug modules which are easier to debug as smaller more manageable amount of code Integrated Development Environment can be used to create a library of modules Program version management software can be used when team are developing modules to ensure only latest version of module is used **Program construction** No need to re-compile modules Modules need to be ‘re-locatable’ in RAM Dynamic link loader loads therefore saving time					

Band	AO1.1b
	Max 12 marks
3	**9-12 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 topics as specified in the indicative content. Clear knowledge is defined as responses that provide relevant detailed points of the implications of writing programs in modules, 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 • effectively drawn together different areas of knowledge, skills and understanding from all relevant areas across the course of study • used appropriate technical terminology confidently and accurately.
2	**5-8 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 topics as specified in the indicative content. Satisfactory knowledge is defined as responses that provide relevant points of the implications of writing programs in modules, which relate to the indicative content. • presented a discussion with limited examples • drawn together different areas of knowledge, skills and understanding from a number of areas across the course of study • used appropriate technical terminology.
1	**1-4 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 responses that provide limited relevant points of the implications of writing programs in modules which relate to a limited amount the indicative content. • used limited technical terminology.
0	Response not credit worthy or not attempted.