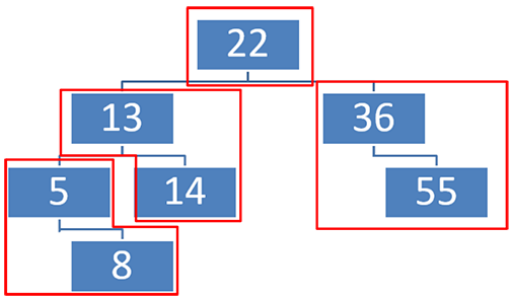
Question			Answer/Indicative content	Marks	Guidance
1	a	i	1 mark each to max 2 • It is a hierarchical structure / not directed • Data is stored in nodes • Nodes are linked by branches/edges • It has a root node • Each node has zero or more nodes 'beneath' it // nodes can link to child nodes • It has leaf nodes / nodes without any lower nodes are leaf nodes • It has no cycles/loops (distinguishing it from a graph)	2	<u>Examiner's Comments</u> This question was generally well answered by most candidates with the most common answers including different types of nodes such as root, child and leaf nodes. Some candidates used vague terminology or used terminology more commonly associated with graphs such as vertices instead of nodes, or edges instead of branches. Imprecise language such as undirected/non-directed was used whereas it would have been clearer to have indicated that trees are hierarchical structures that are rooted. Some candidates erroneously confused binary trees with general trees stating that nodes could only have a maximum of two child nodes, as the tree in this question was not specifically limited to the special case of a binary tree.
		ii	1 mark each • Root node 22 at the start • 13 and 14 in correct order • 5 and 8 in correct order • 36 and 55 in correct order  	4	Do not allow nodes to be drawn downwards. <u>Examiner's Comments</u> Most candidates gained some marks, with many gaining full marks. Occasionally the left/right ordering of child nodes was unclear, so marks could not be given in such instances.

Question			Answer/Indicative content	Marks	Guidance
		iii	1 mark each • Search/traverse tree until the required node is found • Set the parent node pointer to the leaf node to null • Add the deleted node to the free storage list // leave for garbage clear up	2	Examiner's Comments This question was generally less successfully answered by many candidates. There was considerable vagueness in some responses such as 'finding the end node and removing it'. This left questions such as 'Which end node?', 'Removed how?' for the examiner to infer, so were too vague to gain marks. When candidates identified the need to locate the leaf node, most omitted the need for the parent node pointer to be set to null to break the link. Very few candidates indicated that a deleted node would then be added to a free list or that the memory freed could be retrieved via garbage collection.
		iv	1 mark each to max 4 • Check if the root node is equal to search value and if so…. • …return/output/report found • If value is less than root node take left subtree • If value is greater than root node take right subtree • Repeat process with the subtree… • …until search value is found • …until no more branches can be travelled.	4	Examiner's Comments This question was generally well answered by many candidates who appreciated that a Binary Search Tree (BST) could be efficiently searched because it is ordered, rather than traversed. Weaker candidate often confused the search of a BST structure with traversal algorithms such as breadth first or depth first traversals.
		v	1 mark each • Visiting A first… • …Then visiting F, C… • …Then visiting L, T, P… • …Visiting H last Solution: A, F,C, L,T,P, H	4	Examiner's Comments Some candidates confused Depth First Search with other traversal algorithms, most commonly Breadth First Search, but this question was generally answered well by most candidates. Another common mistake was to output the nodes in the order in which they were first encountered rather than the order in which they would be visited/output.

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td></td><td></td><td>vi</td><td>1 mark each to max 2

• When a leaf node is reached…
• …the traversal backtracks to the leaf's parent node
• …backtracks to last node with unvisited children</td><td>2</td><td>Candidates may use an example from the tree in 1a(v) to illustrate their response.

If an answer gives implementational detail of how a stack is used, map to the bullet points given.

<u>Examiner's Comments</u>

Some candidates described the term 'backtracking' in general, so did not answer the question, which required a response in the context of a depth first tree traversal. Many candidates thought that backtracking always began at the leftmost node, which is not true, it starts when a leaf node is reached.
However, a pleasing number of candidates did identify the first example of backtracking in the given tree, from leaf A to parent node C.</td></tr>
<tr><td></td><td>b</td><td></td><td>1 mark for final path A, D, G
1 mark for final distance 14
1 mark for each SECTION or equivalent working shown.

<table>
<tr><th>Node</th><th>Distance travelled</th><th>Previous node</th><th>Marking Guidance</th></tr>
<tr><td>A</td><td>0</td><td>- / N/A / blank / None</td><td rowspan="2">1 Mark</td></tr>
<tr><td>B</td><td>5</td><td>A</td></tr>
<tr><td>C</td><td>2</td><td>A</td><td rowspan="2">1 Mark</td></tr>
<tr><td>D</td><td>10</td><td>A</td></tr>
<tr><td>E</td><td>7</td><td>B</td><td rowspan="2">1 Mark</td></tr>
<tr><td>F</td><td>15</td><td>E</td></tr>
<tr><td>G</td><td><s>19</s>
14</td><td><s>E</s> D</td><td>1 Mark</td></tr>
</table>
</td><td>6</td><td>Nodes should appear in the alphabetical order given if candidates add them as the algorithm progresses but allow other orderings of the nodes.

For the last mark in the table there must be a clear indication that G 19 from E is overwritten by G 14 from D.

Allow equivalent discrete maths approach or textual description.

Check diagram for annotations / solution.

<u>Examiner's Comments</u>

Candidates answering 'by inspection' could gain 2 marks for the final path and distance, and many less successful responses thus scored 1 or 2 marks by doing so, demonstrating little real understanding of how Dijkstra's algorithm operates. A common error was the incorrect solution ACDG that showed ignorance of not continuing the search from the node with the least distance travelled that has not already been marked as visited. For full marks there had to be an indication that the first path to G (19 from E) was overwritten by G (14 from D) for the last mark in the table. Relatively few candidates demonstrated this.</td></tr>
</table>

Question			Answer/Indicative content	Marks	Guidance
			Total	24	

Question			Answer/Indicative content	Marks	Guidance
2			**Mark Band 3 – High level** **(7–9 marks)** The candidate demonstrates a thorough knowledge and understanding of both computational thinking methods; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level** **(4–6 marks)** The candidate demonstrates reasonable knowledge and understanding of both computational thinking methods; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.* **Mark Band 1 – Low Level** **(1–3 marks)** The candidate demonstrates a basic knowledge of both computational thinking methods with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgements if made are weak and unsubstantiated. *The information is basic and comunicated*	9	**AO1: Knowledge and Understanding** **Indicative content** • Problem recognition is identifying that there is a problem to be solved, to determine exactly what the problem is from a description/scenario and to determine if the problem can be solved with computational methods • Decomposition is splitting the problem down into subproblems that can be solved independently **AO2: Application** • Problem recognition: identifying the need for the scheduling system, what it will take as its inputs, what will need to be output etc. • Decomposition: subproblems could include: ◦ inputting the requirements ◦ generating possible routes ◦ evaluating the routes ◦ outputting the schedule **AO3: Evaluation** Computational methods allow the e.g.: • programmer to determine what the problem, what the challenges may be and what additional information is required before starting to code the solution • identification of the key features for programmers to focus on • splitting of the task into smaller, more manageable/solvable problems which allows for a solution to be developed quicker • design of an effective/efficient solution that makes best use of a processor • splitting of a task to allow programmers to focus on areas they specialise in. **Examiner's Comments** Many Level 1 responses gave simplistic definitions of problem recognition and/or decomposition, often with more successful descriptions for decomposition. Relatively few candidates could clearly state that problem recognition starts with identifying inputs and outputs. Some degree of

Question			Answer/Indicative content	Marks	Guidance
			in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear. 0 mark No attempt to answer the question or response is not worthy of credit.		application to the question scenario context was required for Level 2. Most candidates' marks clustered around Level 2 with 4 or 5 marks. Many generic answers focused on problem recognition and/or decomposition as learnt terms, with associated benefits given, but little more than that, which limited marks to Level 2. For Level 3 an evaluation of ideas relevant to the context of the scenario given was expected. Some good responses indicated recognition of a pathfinding type problem with a need for shortest path/A* algorithm, approximation to travelling salesman type solutions or the need to employ heuristics. Candidates giving this level of insight often accessed Level 3.
			Total	9	

Question			Answer/Indicative content	Marks	Guidance
3	a		1 mark each • `headPointer`: To indicate the first element in the list • `freeListPointer`: To indicate the next index to store data in (the freeList)	2	Examiner's Comments While many candidates did gain full marks, a noticeable number of candidates struggled with the use of technical language in this question. Where a candidate's intention or meaning was clear, for example, '~~headPointer~~ is the first item' or '~~freeListPointer~~ is the first free space', marks were given. Some candidates did not show an understanding that these pointers represented the start of two separate linked lists within the static array structure.
	b		It doesn't point to another node Indicates the end of the linked list	1	Examiner's Comments This question was generally well answered, with many candidates gaining marks. A generic 'no assigned value' was not given marks, as the response had to be answered in the context of the linked list, so end of list/not pointing to another node was required, rather than stating pointing to nothing.
	c		• first output red… • …remainder of list correct e.g. red blue grey green purple orange	2	Examiner's Comments Again, this question was generally well answered by most candidates. The most common erroneous answer was 'Blue' as it was the first item in Figure 3 given at index 0. This was invariably given as a response when candidates did not know what the function of the `headPointer` was.

Question			Answer/Indicative content	Marks	Guidance
	d		1 mark each to max 4 Check space available in the free list • Check to make sure `freeListPointer` is not Null Add new data item to first free space in free list • Insert new data item at index `freeListPointer` (index 4) Append e.g. • Traverse to / locate the end of the list (index 3 ‘orange’) • Set the pointer of the last item in the linked list to `freeListPointer` (pointer at index 3 ‘orange’ changes from Null to 4)… • … update `freeListPointer` to the location that new data item pointer is pointing to at present. (`freeListPointer` changes from 4 to 5) • … update pointer from new data item to Null (index 4 pointer changes from 5 to Null) Prepend e.g. • Update `freeListPointer` to point to the location that the pointer from the first item in the free list is pointing to (`freeListPointer` changes from 4 to 5) • … Update pointer from new data item to `headPointer` (index 4 pointer changes from 5 to 1) • … Update `headPointer` to the index of new data item (headPointer changes from 1 to 4)	4	Note descriptions could be for either appending an item to the end of the current list or prepending it to the start. There are different ways to achieve this. Allow answers that illustrate solutions by example from the table in Fig 3 at the start of the question. Reponses must refer to the relevant pointers or give clear exemplifications. <u>Examiner’s Comments</u> More successful responses gave good clear examples to illustrate a potential sequence of operations to describe the process, and many therefore achieved 3 or 4 marks. For example ‘new item added to location 4 indicated by `FreeListPointer` and its pointer set to `Null`; Existing list searched from `headPointer` to its end at *Orange* at index 3, and its pointer updated to the new last item in the list at index 4.’ Some candidates lacked appreciation that this was a static record structure in this scenario, so locations from the free list had to be used. Many candidates were unclear as to how the end of the current linked list was found, and just gave answers from that point onward without saying how it was found by traversing to the end of the existing list. There were relatively few prepend type solutions, but they were accepted as valid where seen.

Question			Answer/Indicative content	Marks	Guidance

	e		1 mark for each statement
			
			`function findNode(toFind, headPointer, linkedList)`
			`  currentNode = headPointer`
			`  while(currentNode != NULL)`
			`  if linkedList[currentNode].data == toFind then`
			`  return currentNode`
			`  else`
			`  currentNode = linkedList[currentNode].pointer`
			`  endif`
			`  endwhile`
			`  return -1`
			`endfunction`
				5	Ignore case of identifiers in pseudocode
					
					Only penalise excessive spaces within identifier names if obvious.
					
					Examiner's Comments
					
					This question required exact answers only. Many candidates gained some marks, and there was a good distribution of marks. More successful programmers tended to get most of the marks available.

			Total	14	

Question			Answer/Indicative content	Marks	Guidance
4	a	i	1 mark for: • `isInteger` • `number` • `result` • `count` • `asciiValue`	1	Penalise excessive spaces in identifiers such as *ascii Value* instead of *asciiValue* Examiner's Comments This question was generally well done (although slightly less well done than parts (a) (ii) and (a) (iii)), but many candidates did very well. The most common erroneous responses were giving the names of predefined functions/properties or giving relational operators.
		ii	(0)5	1	Examiner's Comments This question required an exact answer only and was answered correctly by the majority of candidates.
		iii	(0)3	1	Examiner's Comments This question also required an exact answer only and was answered correctly by the majority of candidates.
	b		1 mark each 03 • Loop through each of the **characters/digits** in the `number` string (parameter) 04 • Find the ASCII value of the current **character/digit** 09 • Return true if the value is an integer and false otherwise	3	Examiner's Comments This question required the *purpose* of the lines of code to be described, but many candidates just described the functionally of the lines of code such as 'line 03 is a counter controlled loop from 0 to number.length -1'. The expected purpose of this line of code was to set up a loop to iterate through each character in the input string parameter. While many candidates could describe the purpose of at least one of the lines of code given, few could clearly describe the purpose of all three lines.

Question			Answer/Indicative content	Marks	Guidance
	c		1 mark each to max 2: • One piece of code can be used many times / in multiple places / makes code more efficient • No need to write the same code multiple times • Takes less time to plan/design/code the program • Easier error detection as fix once and it corrects in each place // less likely to have errors as code is not written multiple times • Makes it easier to maintain the program	2	<u>Examiner's Comments</u> Many less successful responses gave vague generalities such as 'saves time' or 'more efficient' without specifying why or how. Points given must be qualified in some way at A Level. For example, 'saves development time as pre-written routines are available'. Pre-written or pre-tested, and saving development time due to already being written, were the most popular answers.
			Total	8	

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>5</td><td>a</td><td></td><td>1 mark each to max 3

• The function calls itself….
• …..such as line 05 / 07
• Each recursive call will create a new copy of the values in the function….
• ….and add all of the values of the copy the call is being made from to a stack
• There is a base case // condition that stops the recursive calls…
• …condition in line 02
• There may be more than one base case</td><td>3</td><td>Allow answers in context as long as they are clear what the features are.

<u>Examiner's Comments</u>

Weaker responses were limited to one or two points for 'calls itself' with example such as 'line 05'. Fewer candidates went on to identify the requirement for a base case. A variety of terminology was used for the base case such as terminating case and stopping case. If candidates were clearly referring to the case where the recursive calls stopped, and the recursion started to unwind, the mark was given.</td></tr>
<tr><td></td><td>b</td><td></td><td>1 mark for final return value 29 (award in working or answer space) 1 mark each for working

• First call with 10 and second call with 7
• Remainder of calls 6, 3, 2
• Final call value -1
• Adding/showing return values (1 + 2 + 3 + 6 + 7 + 10)

e.g.

<table><tr><th>Function call</th><th>value</th><th>return</th></tr><tr><td>recursiveAlgorithm(10)</td><td>10</td><td>29</td></tr><tr><td>recursiveAlgorithm(7)</td><td>7</td><td>19</td></tr><tr><td>recursiveAlgorithm(6)</td><td>6</td><td>12</td></tr><tr><td>recursiveAlgorithm(3)</td><td>3</td><td>6</td></tr><tr><td>recursiveAlgorithm(2)</td><td>2</td><td>3</td></tr><tr><td>recursiveAlgorithm(–1)</td><td>–1</td><td>1</td></tr></table></td><td>5</td><td>The table is given as guidance, but actual process may be presented in different ways.

<u>Examiner's Comments</u>

While many candidates continue to find recursion a challenging topic there were many who encouragingly achieved full marks. Weaker candidates traced the initial sequence of calls but found it harder to identify the last call to recursiveAlgorithm(-1) that triggered the base case and then found it harder again to calculate the unwind sequence.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>8</td><td></td></tr>
</table>

Question	Answer/Indicative content	Marks	Guidance

6	1 mark each to max 6
	
	• Taking number as input
	• Calculating remainder after division by 8
	• Calculating integer after division by 8
	• Correct loop until 0 is reached (or equivalent method)
	• Concatenating each remainder // storing each remainder in an array/list
	• Outputting the correct result
	
	e.g. pseudocode
	`number = input("Enter a number")`
	
	`endResult = ""`
	
	`while number != 0`
	`  remainder = number MOD 8`
	`  number = number DIV 8`
	`  endResult = str(remainder) + str(endResult)`
	`endwhile`
	`print endResult`
		6	Note candidates can reverse the string before output if they don't concatenate in the order given in the example.
			E.g.
			`endResult = str(endResult) + str(remainder)`
			
			The final markpoint can only be awarded where the correct output will be produced by the algorithm.
			
			<u>Examiner's Comments</u>
			
			In general, there was a very poor standard of pseudocode algorithms from less successful responses. This demonstrated poor programming and limited problem-solving ability.
			
			While there was no requirement to define a function the strongest candidates often did so, but then some forgot the requirement specified in the question for user input, rather than just presenting a parameterised function in isolation.
			
			There were several common errors that were observed. A few candidates performed the octal conversion for just two digits rather than generalising the solution for a denary number of any length. Many candidates simply used '/' for division without specifying integer division through //, DIV or floor functions. Many candidates did not assign the result of a calculation to a variable for later use, presenting lines of code such as '`denaryNum MOD 8`' which was not given marks. The order required for appending the remainder was frequently incorrect in many of the solutions that were presented, although some candidates did reverse the string at the end of the process if they did '`outString += remainder`' style solutions, which was acceptable.
			
			Most candidates achieved at least 1 mark for taking user input, and then the majority also used either modulus to calculate the remainder or integer division to calculate the value for the next iteration. Only the strongest candidates scored 5 or 6 marks.
			
			Exemplar 1

Question			Answer/Indicative content	Marks	Guidance

					Value = inPut ("enter a number")
					outPut = ""
					while value != 0
					outPut = ~~value~~ Str(value MOD 8) + outPut
					value = value DIV 8
					endwhile
					if outPut == "" then
					outPut = 0
					endif
					Print(outPut)
					This candidate response is not language specific but the logic and the operations used are clear. It shows clear logical structure with appropriate indentation of logical blocks. It was given full marks.

			Total	6	

Question			Answer/Indicative content	Marks	Guidance
7	a		**Mark Band 3 – High level** **(7–8 marks)** The candidate demonstrates a thorough knowledge and understanding of Big O; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level** **(4–6 marks)** The candidate demonstrates reasonable knowledge and understanding of Big O; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.* **Mark Band 1 – Low Level** **(1–3 marks)** The candidate demonstrates a basic knowledge of Big O with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgements if made are weak and unsubstantiated. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be*	9	**AO1: Knowledge and Understanding** **Indicative content** • Big O measures the number of steps and memory usage change according to the data as the amount of data being processed increases • Linear - grows in proportion to amount of data • Exponential – the rate of increase is at the rate kn as n increases • Constant - it does not change • Logarithmic – means the rate of increase gets smaller as the amount of data increases time / time increases at a rate of logkn as n increases. **AO2: Application** • Algorithm 1 – The time taken increases as the data set grows. The space taken also significantly increases. This algorithm is not memory efficient. • Algorithm 2 – The time increases significantly and therefore this algorithm is not time efficient. The space will never change which means the amount of memory will not change as the data set grows. • Algorithm 3 – The time will grow less fast as the data set grows relative to the other algorithms. The space required will also increase, but not insurmountably. This is therefore an efficient algorithm with large data sets compared to algorithm 1 and 2 overall. **AO3: Evaluation** • Number of elements is unknown. Exponential is least appropriate because this could increase significantly and be unmanageable. • Constant is the most ideal as the time will not increase. • Algorithm 3 is more suitable because it has a logarithmic time complexity, so it increases less quickly than the other algorithms. It will be reasonable with a small amount (2 items) of data, but then when very large amounts (2 billion items) are needed it will not be significantly more.

Question	Answer/Indicative content	Marks	Guidance
	clear. **0 mark** No attempt to answer the question or response is not worthy of credit.		<u>Examiner's Comments</u> Several candidates confused logarithmic and exponential growth rates, and a significant number of candidates thought that exponential complexity was the polynomial function $O(n^2)$ rather than $O(2^n)$. Many candidates also defined terms such as exponential complexity as 'where it grows exponentially', such circular definitions were not given marks. Level 1 responses identified some characteristics of Big O for constant, logarithmic, linear and exponential complexity with some occasional errors or inconsistencies. Level 2 responses gave reasonable descriptions of both time and space complexity alongside accurate definitions for the complexity terms with an identification of algorithm 3 as being the best overall choice. Level 3 responses had far more evaluative AO3 analysis but were few and far between. In this scenario candidates identified that it depended on the value of n, as if n was very small, exponential time growth was not an issue, so algorithm 2 might be preferential, but as n could grow to 2 billion this would be intractable, so algorithm 3 was chosen as the most practical. The data set in question had 2 billion items at most so $\log_2 2{,}000{,}000{,}000 = 30$ meant space overheads in algorithm 3 were manageable with modern storage capacities. Exemplar 2

Question	Answer/Indicative content	Marks	Guidance
			Order is used to evaluate the complexity of an algorithm, in respect to how its performance changes with a change to the size of input. This may be measuring how much longer it takes to complete, or how much additional load is placed on components such as memory. Linear complexity means that the time (space) taken is proportional to the size of the input, and will grow at a steady rate as input size increases. Constant complexity means that no matter the size of input, the time / load will always be the same regardless of the data being processed. Exponential complexity grows very quickly as input increases, whilst logarithmic increases, though at a Algorithm 1 ~~scales poorly~~ - though linear time is acceptable for small inputs, it quickly becomes unreasonable ~~for~~ with ~~bigger~~ data sets, and the exponential space complexity will mean the data quickly requires much more processing power for just a small increase in input volume. Though Algorithm 2's constant space complexity is ideal, the exponential time complexity means many of the larger possible data inputs will take exceptionally long to complete, and so is impractical and shouldn't be used. Since algorithm 3 has both Time and space complexity as logarithmic it will scale ~~more~~ input volume can increase greatly with well, as This response clearly demonstrates that a high scoring Level 3 response is possible within the space provided. The second page of the response shows clear AO3 evaluation between the merits of the three different algorithms and the scenario.

Question			Answer/Indicative content	Marks	Guidance
	b	i	1 mark for each to max 2 • Processes happen at the same time // processes overlap • One process can start before another one finishes • Each process is given a slice of processor time • Different processes can be executed (in parallel) by different processors/cores	2	Examiner's Comments Several candidates confused parallel processing (executing more than one instruction simultaneously) with concurrent processing (where most than one process/task is running at the same time). A lack of technical vocabulary was observed with many candidates giving responses such as 'many things processed at the same time', without specifying what the 'things' were. Occasionally candidates erroneously thought that processor pipelining was an example of concurrent processing. While many candidates successfully identified that multiple processes are run simultaneously, fewer went on give the mechanism by which this could be achieved. Where they did so, these included timeslicing switching between different processes on a single processor, or use of multiple cores/parallelism to simultaneously execute different processes.
		ii	1 mark each to max 2 e.g. • More efficient processor use // Less idle time for processor // Greater throughput • Long running tasks do not delay short running tasks • Tasks requiring preconditions can wait and then resume execution • User is able to interact with the computer while other tasks are running	2	Examiner's Comments Candidates found it difficult to give well-qualified responses to this question. Many candidates gave the definition for concurrency running multiple tasks at the same time as a benefit, which was not mark worthy. Quicker processing/improved performance was not enough on its own without specifying that this was within a given time unit. Concurrent processing does not increase the actual speed of the CPU. Efficiency as a benefit on its own was insufficient, whereas 'less CPU idle time' was a well-qualified example of a benefit.

Question			Answer/Indicative content	Marks	Guidance
	c	i	1 mark each to max 5 • The data list is split into two lists • These sublists continue to be (recursively) split… • …until each sublist is one item • The first element in two different sublists is compared… • …the smaller item is then selected… • …and written to a new list • …until both sublists fully merged • Repeated until all sorted sublists are recombined	5	Allow array/list as equivalent Examiner's Comments Many candidates were not precise in describing how the initial data set was repeatedly halved and did not explain how the data set was actually broken down into individual items. Very few candidates could accurately describe how the merge phase of a merge sort combines two separate ordered lists together, and frequently this was omitted altogether. A number of candidates continued to demonstrate the misconception that data is sorted within sub-lists, rather than by the actual merging of two separate ordered lists together.  Misconception Many candidates continue to think that a merge sort performs sorting of data within lists. Merge sort performs the sorting part of the operation when it merges together two separate sub-lists that are already in a sorted order. This is done with the use of a pointer to each of the two sub-lists that acts as a placeholder. The two positions in the separate lists are compared, and the smaller item is added to the new sorted list and the pointer is incremented. The merging process then repeats until the sub-lists have been completely merged.

Question			Answer/Indicative content	Marks	Guidance
		ii	1 mark for benefit e.g. • More efficient time complexity (for large data sets) // takes fewer steps to sort the data • Time complexity O(n log n), rather than O(n^2) • Uses divide and conquer • Can apply concurrent processing to reduce sorting time 1 mark for drawback e.g. • More difficult to implement // needs more complex code • Less efficient space complexity // uses more memory with more data items • Space complexity of O(n)/linear, rather than O(1) / constant • Merge sort is always O(nlog2n) whereas the best case for bubble sort is O(n)	2	Examiner's Comments Most candidates recognised that merge sort is typically quicker than bubble sort, but there were some unqualified answers. Clear responses were phrased in terms of Big O complexities or the numbers of data items to be sortonses were phrased in terms of Big O complexities or the numbers of
	d		1 mark for identification, 1 for description of feature e.g. • Error diagnostics • … to locate and fix errors • Breakpoints • …stop a program running at a point to check variables • Syntax highlighting • … to identify key words, variables and help identify syntax errors • Stepping // step through • … run the program line by line to check variable values at each stage • Variable watch window • …view how variables change while the program executes • Auto-complete • … start typing a command/identifier and it completes it	6	Consider awarding description without feature. Allow other suitable answers. Examiner's Comments Most candidates scored well on this question which lent itself to a wide variety of potential answers. Some candidates quoted 'autocorrect' rather than auto complete or suggested predictive code. IDEs can have integrated run-time environments and translation software built-in or attached, but some responses were rather vague or included repetition and did not make it clear that they were referring to an integrated system. Sometimes the descriptive expansion or the initial identification of the point was vague and required the identification plus the description boxes to be taken in combination to give the mark. This was particularly evident when candidates gave a very vague 'debugger/debugging' identification.
			Total	26	

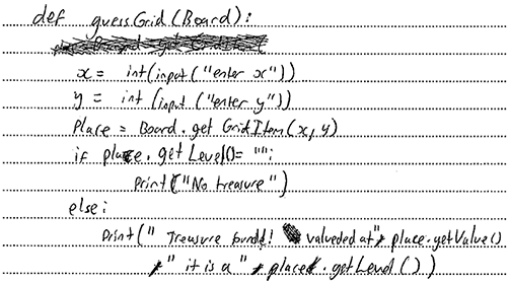
Question			Answer/Indicative content	Marks	Guidance
8	a		1 mark for each input to max 2 • Username • Password 1 mark for output e.g. • Message to request input • Message to state login successful • Message to say login unsuccessful	3	Examiner's Comments This question was generally well answered by most candidates, but answers had to be given in the context of the scenario. Some less successful responses mistakenly identified input/output devices rather than specific examples of inputs and outputs.
	b		1 mark each to max 2 e.g. • Connect to database • Access usernames in file/database • Check username against file/database • Hash password • Access password/hash in file/database • Check password entered/hashed vs stored • Output result	2	Allow other suitable subprocedures that link to the scenario. Examiner's Comments This question was also generally well answered, but answers had to be given in the context of the scenario. Thinking procedurally is specification point 2.1.3 which requires suitable procedures for a given context to be identified. The context was explicitly for logging in to a system, not for creating user accounts or setting up/validating password construction. The most common responses were checking if the username existed and checking whether the username/password combination was correct.
			Total	5	

Question			Answer/Indicative content	Marks	Guidance
9	a	i	1 mark for each description to max 2 and 1 mark for example e.g. • Removal of unnecessary detail… • ….to allow programmers to focus on core aspects of the problem…. • ….simplifies a complex problem Examples, e.g: • Treasure objects are replaced with text labels // no images of treasure are used • Island is set of coordinates and no info as to environment/layout and other objects	3	Allow other suitable examples that are relevant to the treasure game. <u>Examiner's Comments</u> Many candidates identified that abstraction simplified or removed unnecessary detail to gain some marks, but then found it harder to give an example relevant to the scenario. Many candidates gave examples that related to graphics whereas the scenario explicitly stated that the game was text based.
		ii	1 mark each to max 3 e.g. • Reduces programming time • Reduces complexity of code (through abstraction by generalisation) • Reduces amount of memory required / computational power • Simplifies the problem so it's easier to solve / understand (by recognising common patterns) • Allows programmers to focus on core aspects of the problem	3	<u>Examiner's Comments</u> Many candidates gained some marks, but most found it difficult to identify three distinct reasons, and repetition was often observed in responses. Again, some candidate answers did not provide suitable levels of qualification for the points given.

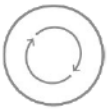
Question			Answer/Indicative content	Marks	Guidance
	b	i	1 mark each • Defining class Treasure • Defining the private attributes value and level • Defining a new public procedure… • …Taking two parameters (integer and string) • Correctly assigning both parameters to the attributes e.g. class Treasure private value private level public procedure new(valueP, levelP) value = valueP level = levelP endprocedure endclass	5	Allow use of this/self or equivalent dependent on language public procedure new(value, level) this.value = value this.level = level endprocedure Python answers must either use comments to indicate private attributes or use the double underscore private attribute convention to be credited. self.__level self.level # private <u>Examiner's Comments</u> This question either tended to be very poorly answered or very well answered. There was a clear division between candidates who were comfortable writing OOP code and class constructors and those that were not. Weaker candidates with little OOP experience often confused the class declaration with the instantiation method or offered no response. Many candidates often assigned the parameters to the private attribute values rather than setting the attributes to the parameters being passed in. Where candidates gave responses in programming languages and the intent of the response was clear, marks were given. However, sometimes Python programmers did not make it clear (by convention/comment) that class attributes were private.

Question			Answer/Indicative content	Marks	Guidance
		ii	1 mark each • get level method header with no parameter • Returning level attribute e.g. public function getLevel() return level endfunction	2	Note Python *self* will appear, but no other parameters def getLevel(self): Examiner's Comments While there were many good responses it was also noted that a significant number of candidates erroneously tried to send a parameter to a getter() function and then return the same parameter. Some candidates did not return a value or simply tried to print the value. Other errors included writing the getter method as a function call by omitting any indication that a function was being defined.
		iii	1 mark each • Encapsulation • Allowing an attribute to only be accessed/changed via a method	2	Examiner's Comments Some candidates randomly picked an erroneous OOP term like polymorphism. Those candidates who could correctly identify the use of encapsulation did not always go on to specifically state that this meant access was controlled by a public getter() method.
	c		1 mark for each completed statement public procedure new() for row = 0 to 9 for column = 0 to 19 grid[row, column] = new Treasure(-1,"") next column next row endprocedure	5	Examiner's Comments Most candidates scored at least the first 2 mark-points for the grid dimensions, but far fewer could construct OOP code requiring the attribute and the relevant default parameter value to be given.
	d		1 mark each to max 7 • Procedure declaration taking parameter • Taking two inputs for row and column from the user • Accessing item at grid position… • …using correct get methods getGridItem • Checking (treasure) object's level/value… • …using correct get method getLevel getValue • …outputting "No treasure" if empty • …otherwise outputting value and level e.g. procedure guessGrid(gameboard) rCoord = input("Enter R coordinate")	7	Note candidates may attempt to access private attributes directly gameboard.grid(x,y) for example, instead of gameboard.getGridItem(x,y). Credit cannot be given for the dependent second mark using appropriate get method if they do this, but FT marks can be awarded for later points if a reasonable attempt has been made. Examiner's Comments Many candidate responses gave very little beyond the procedure declaration and taking in the x and y coordinates as inputs, thus scoring 2 marks at most. There was less successful understanding of how to use the get() methods provided in the

Question	Answer/Indicative content	Marks	Guidance

	cCoord = input("Enter C coordinate")
	treasureItem = gameBoard.getGridItem(rCoord, cCoord)
	if treasureItem.getLevel() = "" then
	print("No treasure")
	else
	print("This treasure is level ", treasureItem.getLevel(), " with value ", treasureItem.getValue())
	endprocedure
			scenario to access the data. This is an area of the specification that candidates require extensive practical experience of to be able to fluently answer questions like this in examination conditions.
			
			Many candidates tried to used grid[x,y] by inserting the coordinates into the parameter values and did not identify it as an instance of an object that needed its attributes to be accessed via the appropriate getter methods. Those candidates who tried to access the attributes directly without the appropriate getter methods did achieve some further marks with follow through marks.
			
			Competent coders often gave responses worthy of full marks within just a few lines of code.
			
			Exemplar 3
			
			 
			def guessGrid(Board):
			x = int(input("enter x"))
			y = int(input("enter y"))
			Place = Board.getGridItem(x,y)
			if place.getLevel()= "":
			Print("No treasure")
			else:
			Print("Treasure found! valueded at" place.getValue()
			" it is a " placed.getLevel())
			
			This response clearly shows a logical and well-structured algorithm that uses the appropriate get() methods to access the relevant attributes of the passed parameter object. Candidates who are fluent in the use of OOP can present elegant and concise solutions.

Question			Answer/Indicative content	Marks	Guidance
	e		1 mark each to max 4 e.g. • Code can easily be reused… • …classes can be used in other programs • …inheritance can be to extend upon existing classes • …as a class can be based on an existing class • Easier to maintain…. • ….as classes can be modified or extended • …debugging can be easier as encapsulation limits how attibutes are changed. • Code can be more secure… • … as access to attributes can be restricted to being via methods. • Better for coding as part of a team… • …as classes can be distributed between team members.	4	1 mark per benefit identified and 1 mark per expansion. Max 2 benefits and 1 expansion per benefit. <u>Examiner's Comments</u> Many candidates were more successful on this part of Section B as it was an AO1 book learnt topic rather than the preceding AO2 OOP programming application elements. However, benefits were often poorly described, or OOP components were just named without a relevant expansion as to how they could be used.
	f		Mark Band 3 – High level (7–9 marks) The candidate demonstrates a thorough knowledge and understanding of parameters and local/global variables; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* Mark Band 2 – Mid level (4–6 marks) The candidate demonstrates reasonable knowledge and understanding of parameters and local/global variables; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the	9	AO1: Knowledge and Understanding Indicative content • Local variable can only be accessed within sub-program/main program it is declared within • Global variable can be accessed by all sub-programs • Parameters are items passed to a subproblem • Passing by reference sends a pointer to the original value, so this will be changed when control is returned • Passing by value sends the a copy of the value, so the original will not be changed when control is returned AO2: Application • If board is local it can only be accessed in the main program • This will need to be passed to any sub-programs that need to use it • If the board needs to be changed it will need passing by reference, so that the board is updated • If it only needs to be accessed and not changed it can be passed by value AO3: Evaluation • If global then this would be present in memory throughout hence using more

Question	Answer/Indicative content	Marks	Guidance
	most part appropriate, although one or two opportunities for development are missed. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.* Mark Band 1 – Low Level (1–3 marks) The candidate demonstrates a basic knowledge of parameters and local/global variables with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgements if made are weak and unsubstantiated. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* 0 mark No attempt to answer the question or response is not worthy of credit.		memory • …however the board will be required throughout the program so may be as efficient as passing it through parameters • …if global then the programming may be more straight forward, and less likely to have errors with passing the board incorrectly to subprograms, i.e. it may not be updated when it needs to be • Using local means that the board can be manipulated by subprograms without affecting the actual board if needed, for example to simulate potential changes. Examiner's Comments A good range of Level 2 responses were observed with competent descriptive definitions of local versus global and byVal/byRef parameter passing. There were far fewer high level AO3 evaluations of relative memory use within the context of the scenario when passing byVal/byRef for a relatively large grid object. Good, contextualised responses took the elements of the scenario in section B of the paper and talked about the grid object and passing parameters like x and y coordinates. Responses such as this were relatively rare.  Assessment for learning Candidates would benefit from producing solutions centred around the scenario presented in Section B of the paper. Having extensive OOP programming experience in a relevant high level language will help candidates to successfully tackle questions where algorithms have to be presented. Sample Python code based on Section B is presented below for consideration. Assessment for learning # 2023 Section B class Treasure:

Question	Answer/Indicative content	Marks	Guidance

			def __init__(self, pValue: int, pLevel: str):
			self.__value = pValue
			self.__level = pLevel
			
			def getValue(self):
			return self.__value
			
			def setValue(self,pValue: int):
			return self.__pValue
			
			def getLevel(self):
			return self.__level
			
			def setLevel(self,pLevel: str):
			return self.__pLevel
			
			class Board():
			
			def __init__(self):
			self.__grid = [[Treasure(-1,"") for _ in range(20)] for _ in range(10)]
			
			def getGridItem(self,x: int, y: int):
			return self.__grid[x][y]
			
			def setGridItem(self,x: int,y: int, pTreasure: Treasure):
			self.__grid[x][y] = pTreasure
			def guessGrid(b: Board):
			row = int(input("Row: "))
			col = int(input("Col: "))
			t = b.getGridItem(row, col)
			if t.getLevel() == "":
			print("No treasure")
			else:
			print(f"Treasure found at {row} {col}!")
			print(f"Level {t.getLevel()} Value {t.getValue()}")
			
			# Test code - Treasure @ 2,2 all other locations no treasure
			island = Board()
			prize = Treasure(5,"Prize!")
			island.setGridItem(2,2,prize)
			guessGrid(island)

	Total	40	