Mark scheme

Question			Answer/Indicative content	Marks	Guidance

1	a		1 mark for each completed statement
			`function binarySearch(array, numberToFind)`
			`lowerbound = 0`
			`upperbound = array.length - 1`
			
			`while true`
			`if(upperbound <lowerbound) then`
			`return -1`
			`else`
			`mid = (upperbound + lowerbound)DIV 2`
			`if(array[mid] <numberToFind) then`
			`lowerbound = mid + 1`
			`elseif(array[mid] > numberToFind)`
			
			`then`
			`upperbound = mid - 1`
			`else`
			`return mid`
			`endif`
			`endif`
			`endwhile`
			`endfunction`
				6	**Accept**
					`mid` = **int**((`upperbound` + `lowerbound`)/ **2**)
					
					Examiner's Comments
					
					Binary search (in both iterative and recursive form) is a standard algorithm that candidates are expected to be able to code at AS Level. Candidates made a number of common errors including not using integer division (`DIV` and / were accepted) for the midpoint calculation.
					
					Some candidates returned the value being searched for at the array index rather than returning mid as the index as specified in the question.
					
					Another common error that was made was to update `mid` by the wrong value, swapping the +1 and -1 round.

	b	i	1 mark per bullet to max 5 • 20 becomes the sorted list / [20] is the sorted list • ... take 8 and compare against ... • ... it is less than 20 so keep it in place / [20 8] is now the sorted list • Take 33 and compare it to each value in the sorted list. It is greater than 20, so move 20 and 8 and insert 33 / 33 is compared and moved down and [33 20 8] is now the sorted list • Repeat for all other elements.	5	Max 4 if ascending and not descending Zero marks if not provided a description and just shown data values **Examiner's Comments** Insertion sort is one of the standard sorting algorithms that candidates need to be familiar with using. A pleasing number of candidates made a reasonable start to the

					response, but fewer could produce a comprehensive response. The question required candidates to describe and not just show how insertion sort worked, so responses that only gave diagrams showing numbers moving with no annotation did not gain credit. Some candidates mistakenly described bubble sort or merge sort. Quite often, where candidate scored four marks, they omitted the first pass of the sort where the first item (`20`) is set as the sorted list. Some potentially quite good responses were too vague as to how the next item at the start of the next pass filtered back through the sorted list to its correct position.
		ii	1 mark for each set of test data **and** purpose e.g. • **Test Data:** 7 6 5 4 3 2 1 **Purpose:** Testing that data is already in order • **Test Data:** 1 7 2 6 3 5 4 **Purpose:** Testing that mixed data is sorted • **Test Data:** Six One Seven Three Four Fix Two • **Purpose:** Testing that words are rejected/ Testing words are sorted (alphabetically) • **Test Data:** -1, 0, 2, -1, 3, -3 • **Purpose:** Testing the algorithm works with negative numbers.	2	Each purpose must be different. Accept viable alternatives The purpose must be relevant to the test data given. A mark cannot be award unless both the test data and purpose are given. **<u>Examiner's Comments</u>** Many candidates gained some credit for giving a valid reason for a set of test data that they presented. Clear responses included testing decimals, negatives, different data types, and testing on data

					that was not already in order.
			Total	**13**	
2			1 mark for similarity: • Both are data structures / store data • Both allow data to be added/removed • Both have identifier(s) / pointer values 1 mark for difference: • Stack is LIFO and queue is FIFO • Queue supports enqueue/dequeue operations whilst stack supports push/pop operations • Queue require two pointers whilst stacks only require one pointer	2	**Examiner's Comments** Most candidates demonstrated knowledge of the purpose of stacks and queues. For the similarity, a frequent response was that both stacks and queues are data structures used to store data. For the difference, most candidates identified that a stack was a LIFO structure while a queue was a FIFO structure.
			Total	**2**	
3			**Mark Band 3-High Level** **(7-9 marks)** The candidate demonstrates thorough knowledge and understanding of suitability of algorithms; 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. The candidate provides a thorough discussion which is well-balanced. Evaluative comments are consistently relevant and well-considered. 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 suitability of algorithms; 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.	9	**AO1: Knowledge and Understanding** e.g. • Kofi's algorithm reads data from the text file and writes these value to an array. It cannot be called by different programs, doesn't work with different text files and does not consider the number of items in the file. • Zac's program also reads data from the text file and writes these values to an array. It can be called by different programs, works with different file names and does consider the number of items in the file.

			Mark Band 1-Low Level **(1-3 marks)** The candidate demonstrates a basic knowledge of suitability of algorithms, with limited understanding shown; the material is basic and contains some inaccuracies. The candidate 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. Judgments if made are weak and unsubstantiated. The information is basic and communicated 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.		**AO2.1: Application** e.g. Kofi's algorithm: • Has additional/unnecessary variables • ...e.g. fileName, dataValue • Loops 1000 times which is inappropriate when it needs to be used with unknown quantity • Will not read any lines after the first 1000 Zac's algorithm: • Uses a function so that is can be used in different programs / is independent • Takes file name as parameter so that it can be called with different files • The array size is 100. Therefore, if the data being read is greater than 100 the program will crash. • Does not close the file **AO3.3: Evaluation** e.g. • Zac's is more memory efficient as it uses fewer variables • Kofi's is more suitable in that it closes the file so it does not remain open in memory • Zac's is more suitable to be used in different

					programs due to function and not hard coding the file name • Zac's is easier to alter for a different file size as fewer lines of code would need to be altered **Examiner's Comments** There were a large number of either blank or very brief responses where candidates displayed no ability to read and interpret code. The question source material was designed to help candidates to comment on a range of programming practices such as file opening and closing, conditional versus count-controlled loops and the benefits of the use of functions with parameters. Level 1 responses often just focused on the use of two different loop types. There were few Level 3 responses that gave insightful evaluative comments. Where candidates were able to do so they often focused on memory usage or the ability to cope with different data sets that could be read. Exemplar 2

					 This response showed that the candidate has thought about the issues related to the question and had carefully annotated the code when analysing it. The response was well-structured taking each of the algorithms in turn and displayed a clear level of evaluative insight.
			Total	**9**	
4		i	1 mark each • <u>Linear</u> • The time will (increase) in **direct proportion** to the number of items	2	**<u>Examiner's Comments</u>** Many candidates did not identify linear but gave a description for 'directly proportional' or vice-versa, so only gained one mark. Candidates needed to specify that the time required increased in direct proportion to the number of items. Proportion on its own was too vague and could have meant inverse proportion

					for example. Incorrect responses included 'quickest time to find / least operations' defining best case rather than answering the question regarding O(n).
		ii	1 mark each • Logarithmic • The additional memory space required grows at a decreasing rate as the number of items increases	2	**Examiner's Comments** Many candidates did not identify logarithmic but tried to give a description or vice-versa. Many candidates just said the memory space increased in proportion to the log of n without explaining what this meant. For the second mark candidates had to explain that as the number of items n increased the amount of additional memory required became progressively smaller.
		iii	Constant / O(1)	1	**Examiner's Comments** The vast majority of candidates gave the correct response of constant or O(1).
		iv	Exponential / $O(2^n)$ / $O(K^n)$	1	**Examiner's Comments** Half the candidates correctly identified exponential complexity, but common erroneous responses included $O(n^2)$ or O(2n) instead of $O(2^n)$. **Misconception** A number of candidates

					erroneously thought that n^2 or 2^n demonstrated exponential growth instead of 2^n. Candidates need to have the mathematical grounding to understand the difference between different Big O growth factors.
			Total	**6**	
5	a		1 mark each to max 2 for justification e.g. • Can store multiple items of data under **one identifier** / so all the data about a task can be accessed using the same identifier • Can store data of **different data types** and this task has string, real and integers	2	**Examiner's Comments** Record structures were poorly understood, and it was clear that many candidates had very limited experience of using records / structures within a programming language. Many candidates gave responses related to database records rather than record data structures.
	b	i	1 mark each e.g. • Each node can have 0, 1 or 2 child nodes / a maximum of 2 child nodes • Nodes are ordered (with left nodes less than the parent and right nodes greater) • The location to which a node is added depends on its order.	2	**Examiner's Comments** Some candidates gave generic properties of trees such as 'root' instead of specific characteristics of a binary search tree as required by the question. There was then some lack of precision when describing the number of child nodes each parent node could have (maximum two, not always two), or lack of clarity in defining 'ordered' without qualifying what this meant.
		ii	1 mark for advantage e.g. • Searching is faster (O(log n)) • Inserting new tasks is faster	1	**Examiner's Comments** A number of candidates erroneously talked about

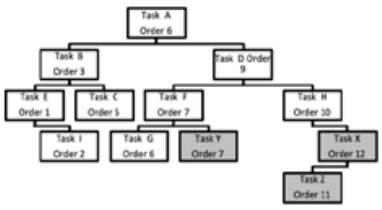
<table>
<tr><td></td><td></td><td></td><td>

- Do not need to sort the structure (each time a new task is inserted)

</td><td></td><td>binary tree traversal to traverse a 1D list. However, there were many clear and correct responses. Some unqualified ‘easier’ / ‘quicker’ type responses gained no credit. Candidates had to identify that it was quicker to search/insert into the tree.</td></tr>
<tr><td></td><td></td><td>iii</td><td>1 mark for each row

<table>
<tr><th>Statement</th><th>Depth-first (post-order)</th><th>Breadth-first</th><th>Neither of these two traversals</th></tr>
<tr><td>All nodes at the current depth are visited before moving to the next depth</td><td></td><td>✓</td><td></td></tr>
<tr><td>The algorithm traverses to the end of one branch before moving to another branch</td><td>✓</td><td></td><td></td></tr>
<tr><td>The algorithm will make use of backtracking</td><td>✓</td><td></td><td></td></tr>
<tr><td>The traversal can be used to output the contents of the tree in ascending order</td><td></td><td></td><td>✓</td></tr>
<tr><td>The algorithm will output the root node last</td><td>✓</td><td></td><td></td></tr>
</table>
</td><td>5</td><td><u>Examiner’s Comments</u>

Many candidates achieved at least partial credit if not full credit.</td></tr>
<tr><td></td><td></td><td>iv</td><td>1 mark for Task Y to right of Task F
1 mark for Task X to right of Task H …
… 1 mark for Task Z to left of Task X

 
</td><td>3</td><td>The direction of left/right child nodes must be clear and cannot just be a downward vertical line.

<u>Examiner’s Comments</u>

The binary search tree diagram generally was answered well with the majority of candidates gaining full marks.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>13</td><td></td></tr>
<tr><td>6</td><td>a</td><td></td><td>1 mark each to max 3. Max 2 for generic answers with no relation to scenario. e.g.

- Has a set/fixed number of values

</td><td>3</td><td><u>Examiner’s Comments</u></td></tr>
</table>

			• …and the number of spaces in the road will not change • Stores data of one type • … as the array is only made up of prize objects • Stores data linearly • … match the linear nature of the road • Array contents are mutable • … so prizes can be added/removed from the road • A single indentifier is used to directly index • … any position in the road • Can be iterated by index • … to perform an operation on all road positions		Many responses were too vague, showing little knowledge of the properties of arrays. Relatively few candidates appeared to be able to make explicit links to the scenario to achieve full marks.

	b	i	1 mark each
			• Function/subroutine with identifier `getName` taking **no** parameters
			• Returning `name`
			
			e.g.
			`public function getName()`
			`return name`
			`endfunction`
			
			`def getName(self):`
			`return self.__name`
			
			`public getName(){`
			`return name`
			`}`
			
			`function getName() {`
			`return this.name`
			`}`
				2	BP1 Do not award procedure or method
					
					BP1 Allow self as an additional parameter if Python is used.
					
					BP1 If an access modifier is given for the method, it must be public and not private.
					
					BP2 Do not allow any modified name attribute to be returned.
					
					Examiner's Comments
					
					While many candidates had little difficulty giving code for a `getter()` there were a number of common errors. Some candidates used a private access modifier when a `getter()` needs to be public. There was often erroneous use of 'procedure' whereas a `getter()` is a function that must return a value. Some candidates tried to set values within the `getter()` function when it should only have returned the class attribute value.

		ii	1 mark each • New instance of `prize` …	3	MP2 allow any order of parameters

			• … with “Box”, “money” and 25 as parameters
			• Assigned to `allPrizes` index 3
			
			e.g.
			`allPrizes[3] = new prize("Box", "money", 25)`
			`allPrizes[3] = prize.new("Box", "money", 25)`
			`allPrizes[3] = prize("Box", "money", 25)`
					"`Box`" and "`Money`" must be strings and `25` must be an integer
					
					Allow `prize.new()` as new is given as the constructor method in the class diagram
					
					Examiner’s Comments
					
					Many candidates struggled with the instantiation of an object. Where candidates made an attempt to instantiate some did not use a string for “box” and “money” or did not give 25 as an integer but instead gave the string “25”.

		iii	1 mark for each bullet to maximum 3 e.g. • **Decision** - check whether the space already has a prize allocated … • **Action if true** - another space/number will need to be generated • **Action if false** - the prize will be stored here • **Decision** - check if all 10 prizes have been allocated … • **Action if true** - the algorithm needs to stop generating numbers • **Action if false** - a new number/space needs to be generated and checked	3	Give: • 1 mark for stating a decision • 1 mark for the action required if true • 1 mark for the action required if false **Examiner’s Comments** There were only two reasonable decisions that could be given from the scenario details. Candidates needed to make it clear that a decision with a Boolean output was present that would dictate two potential outcomes. Some candidates quoted actions such as ‘randomly assign space for prize’ which did not represent a decision. Many responses described the mechanics of setting up the game and

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>the random spaces but did not highlight the program conditions/decisions as required.</td></tr>
<tr><td></td><td>c</td><td>i</td><td>1 mark each

• Constructor header (any suitable name e.g. new, constructor, create, init)
• …taking one parameter only
• Initialising name to the parameter
• Initialising money to 5
• Initialising experience to 0 and roadPosition to 0

e.g.

Pseudocode example:

Python Example:

C# Example:

<pre>public procedure new(pName)
name = pName
experience = 0
roadPosition = 0
money = 5
endprocedure</pre>
<pre>def __init__(self, pName):
self.__name = pName
self.__experience = 0
self.__roadPosition = 0
self.__money = 5</pre>
<pre>public Character(string pName)
{
string name = pName;
int experience = 0;
int roadPosition = 0;
int money = 5;
}</pre></td><td>5</td><td>Allow minor changes to identifiers as long as purpose is clear.

Allow
<pre>procedure
new(pName)
this.name = pName
…</pre>(or similar e.g. self.name)

Allow two parameters if one is self and the response is clearly in Python.

The parameter name should be different to the attribute name.

<u>Examiner’s Comments</u>

It was clear that those candidates with limited OOP programming knowledge found the writing of a relatively simple constructor method difficult. Those with relevant programming experience often found this to be a very straightforward question. Common errors included passing additional values to set the <code>experience</code>, <code>roadPosition</code> and <code>money</code> attributes rather than setting them to the constant values indicated in the question.</td></tr>
<tr><td></td><td></td><td>ii</td><td>1 mark each

• Procedure/method header …
• … taking two parameters, type (or similar) followed by value (or similar) …
• … compare type parameter with "<code>money</code>"
• … compare type parameter with "<code>experience</code>"</td><td>5</td><td>Do not allow Function for BP1

BP2 parameters must be given in the correct order to match the calls to <code>updateValues()</code> in the question.</td></tr>
</table>

			• … both attributes updated correctly and nothing else modified
			
			e.g.
			`public procedure updateValues(pType, pValue)`
			`if pType == "money" then`
			`money = money + pValue`
			`elseif pType == "experience"`
			`experience = experience + pValue`
			`endif`
			`endprocedure`
			
			`def updateValues(self, pType, pValue):`
			`if pType == "money":`
			`money += pValue`
			`elif pType == "experience":`
			`experience += pValue`
					"`money`" and "`experience`" must be string values
					
					Examiner's Comments
					
					The `updateValues` procedure again proved problematic for candidates with limited OOP experience. No marks were given for the first mark point if a function was declared as there was no return value. Parameter names needed to be fit for purpose, understandable, and had to match the order given in the question scenario to work for the given example calls.

	d		1 mark for each completed space
			
			`character1 = new Character("Jamal")`
			`newPosition = 0`
			`while newPosition < 50`
			`move = random(1, 4)`
			`character1.changePosition(move)`
			`newPosition = character1.getRoadPosition()`
			`if newPosition < 50 and road[newPosition] != null`
			`then`
			`prizeType = road[newPosition].getType()`
			`valueAmount = road[newPosition].getValue()`
			`character1.updateValues(prizeType, valueAmount)`
			`print("Congratulations you are in position", newPosition, "and found", road[newPosition].getName())`
			`print("Money", character1.getMoney(), "and experience", character1.getExperience())`
			`endif`
			`endwhile`
			`print("You reached the end of the road")`
				6	Allow `road.length / len(road)` instead of `50`
					
					Allow `<=49` instead of `< 50`
					
					Examiner's Comments
					
					Nearly all candidates achieved some marks, and a majority scored five or six marks.

	e		1 mark each • (Line 02) `for x = 0 to 49` • (Line 03) `print("Space", x)` • (Line 06) `else / elseif road[x] <> null`	4	Line `07` allow `print(road[x].name)` **Examiner's Comments** Many candidates scored three or four marks but in

			• (Line 07) `print(road[x].getName())`		general candidates found it harder to identify errors in the code than to complete code in the previous question. Some candidates didn't give the line number but rewrote the incorrect line before giving the corrected line, which was acceptable, although not ideal given the scaffolding.
	f		**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of global variables and the alternatives; 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 global variables and the alternatives; 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 global variables and the alternatives 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.*	9	**AO1: Knowledge and Understanding** **Indicative content** • Global variables are created when the program starts, all subroutines can access/update the contents • Local variables are created in the subroutine they are created in, they are not accessible directly from any other subroutine • Local variables are removed from memory when the subroutine ends. • Local variables can be passed as parameters to a function to be updated, and then returned to override the original local variable • Local variables can be passed by reference to a subroutine to allow the content of the variable to be updated **AO2: Application**

			0 mark No attempt to answer the question or response is not worthy of credit.		• The variables will be stored in memory throughout the whole code execution. However, the amount of data they are storing is relatively low so would not use a lot of memory. • When the game is expanded, the amount of data may increase so it could be memory intensive, especially if graphics are used in the game. • Both arrays are needed throughout the whole game so keeping them as global will make writing the code easier as the programmer will not need to keep passing them as parameters and setting return values. • Only one part of the game is being created initially and therefore the use of global variables would not affect the efficiency greatly. However, when the program expands, it could cause accuracy / testing / debugging and maintenance problems. **AO3: Evaluation**

					• As this is only a prototype, the use of global variables would be beneficial. • However, when the game expands, the use of global variables could create issues such as running out of memory, coupling, testing & debugging problems and maintenance problems. • The programmer may be best to keep the variables as local and then pass them between the different subroutines as parameters byVal and byRef. **Examiner's Comments** Most responses were Level 2 for definitions and some expansion to passing parameters. Very few candidates were able to go into depth about alternatives to global variables such as passing by value and passing by reference in detail or extending to issues such as scalability within a larger more extended game. Few candidates picked up on the fact that this was a more limited prototype that was likely to be expanded on which would require more consideration to be given to variable scope.

<table>
<tr><td></td><td></td><td></td><td>Total</td><td>40</td><td></td></tr>
<tr><td>7</td><td>a</td><td></td><td>
<table>
<tr><th>Node</th><th>Distance travelled</th><th>Heuristic</th><th>Distance travelled + Heuristic</th><th>Previous node</th><th>Marking Guidance</th></tr>
<tr><td>A</td><td>0</td><td>90</td><td>90</td><td>N/A</td><td>1 Mark</td></tr>
<tr><td>B</td><td>20</td><td>80</td><td>100</td><td>A</td><td rowspan="2">1 Mark</td></tr>
<tr><td>C</td><td>44</td><td>43</td><td>87</td><td>A</td></tr>
<tr><td>D</td><td>128</td><td>70</td><td>198</td><td>E</td><td>1 Mark</td></tr>
<tr><td>E</td><td>66</td><td>20</td><td>86</td><td>C</td><td>1 Mark</td></tr>
<tr><td>F</td><td>81</td><td>8</td><td>89</td><td>E</td><td>1 Mark</td></tr>
<tr><td>G</td><td>90</td><td>0</td><td>90</td><td>F</td><td>1 Mark</td></tr>
</table>

Path: A→ C →E → F → G Distance: 90 (1 Mark)</td><td>7</td><td>For Row A allow N/A, None, Null, - or a blank cell / equivalent.

<u>Examiner’s Comments</u>

Many candidates demonstrated a good understanding of the A* algorithm and most candidates achieved at least some of the marks available, with the most commonly given being the final path (often through inspection) and the first row of the table.

A few candidates treated the algorithm as if it were Dijkstra’s and explored all possible paths/routes rather than stopping as soon as the goal node was located.</td></tr>
<tr><td></td><td>b</td><td></td><td>1 mark for each difference up to a maximum of 4 marks:
e.g.

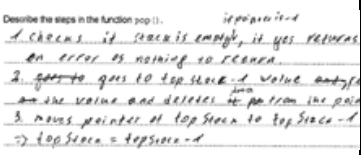
• Trees have one root node / graphs do not have a root node (1)
• Trees do not allow cycles/loops / graphs do allow cycles / loops (1)
• Trees store hierarchy / graphs have no hierarchy (1)
• Trees are always undirected / graphs can be directed (1)
• Trees are always connected / graphs can be connected or disconnected (1)</td><td>4</td><td>Do not allow responses related to weighted / unweighted.

<u>Examiner’s Comments</u>

Most candidates struggled to score more than one or two marks for this question. Responses had to be clearly mappable to technical terms, but there was evidence of vague language in many cases. Candidates were expected to be able to talk about root nodes, cycles, hierarchy, directed/undirected edges and connected/disconnected nodes. Clear use of technical terms is expected at this level. A common incorrect response was</td></tr>
</table>

					weighted/unweighted. A minimum spanning tree is a subset of a graph that can have weighted edges.
			Total	**11**	
8	a		1 mark each • `headPointer` to identify the **first item/element** in the queue / identify which item to dequeue/remove next • `tailpointer` to identify the **next free** space in the queue / identify where the next item/element will be enqueued/added	2	**Examiner's Comments** This question was generally well answered by many candidates. There were some vague responses such as 'start of queue', which was not precise enough to indicate the first item, or the index of the first item, as distinct to the indexes of the actual data structure itself.
	b		1 mark for queue elements 1 mark for both pointers 	2	Allow 20 and 15 in place but crossed out OR allow 20 and 15 in place only if `headPointer` and `tailPointer` are correct **Examiner's Comments** Most candidates either tended to score full marks or no marks. Many candidates missed labelling the pointers in their diagrams and just gave updated queue values. Some candidates kept the values 20 and 15 on the diagram and correctly moved the `headPointer` to point to index 3 and the `tailPointer` to the next free space available. A considerable number of candidates erroneously shifted all items forward, showing a lack of understanding as to how a queue is efficiently implemented.

	c		1 mark each to max 2 e.g. • A queue is a FIFO structure / elements processed in the order entered • A queue will not allow new data inserted at the front / only allows new data to be enqueued at the rear • The queue contents cannot be resequenced/sorted without rewriting	2	**Examiner's Comments** The first part of many responses was well attempted, with most candidates identifying the First In First Out (FIFO) property of a queue. Far fewer candidates were able to successfully expand on this for the second mark. A linked explanation was required such as a higher priority item cannot be inserted at the front/in the middle of the queue because only the first element can be accessed/dequeued. Some candidates just reiterated what FIFO meant instead of giving the linked explanation to a priority queue which was insufficient to gain the second mark.
			Total	**6**	
9	a		1 mark each • Check if the stack is empty / check `topStack` is equal to 0 • …and if so return a suitable value (e.g. -1/ null) / do nothing /give warning • (If not empty) decrement `topStack` • Return the value in element `topStack` from the array numbers	4	Do not award BP3 if a value has been returned from the function for BP4 first. **Examiner's Comments** There were many weak responses that did not outline the discrete steps required in the `pop()` function and there was often no reference to the `topStack` pointer. A number of candidates confused a stack with a queue and talked about head/rear pointers. Some candidates confused 'space for 100 elements' in the question with the top of the stack and then

					erroneously talked about removing item 100. Some candidates erroneously stated that the value at `topStack - 1` would be returned before saying that `topStack` would be decremented. The order of the steps was important for the function to operate correctly, and many candidates lost a mark due to this. Exemplar 2  This response shows the candidate returning the value from `topStack - 1` in step 2. As soon as the function returns a value no further actions are performed within the function, so decrementing `topStack` in point 3 was not given.

	b		1 mark for each completed statement
			
			`function push (`**`dataValue`**`)`
			`if` **`topStack`** `!= 100 then`
			`numbers[`**`topStack`**`] = dataValue`
			`topStack = topStack +` **`1`**
			`return true`
			`else`
			`return false`
			`endif`
			`end function`
				4	**Examiner's Comments**
					
					The majority of candidates scored three or more marks. Some candidates erroneously used `numbers.length` or `len(numbers)` in the second space instead of `topStack`.

	c		1 mark each • Calling push() with parameter 15 • …storing/using return value in selection … • …comparing true/false (may be implicit e.g. `if push(15) then`) … • …outputting a suitable message if false and if true	4	True/False comparisons must be Boolean values and not strings, but allow FT after that. If `push()` is called twice BP4 cannot be awarded.

<table>
<tr><td></td><td></td><td></td><td>e.g.

<pre>
added = push(15)
if added = false then
  print("Not added")
else
  print("Added")
endif
</pre>
<pre>
if push(15) then
  print("Added")
else
  print("Not Added")
endif
</pre></td><td></td><td><u>Examiner's Comments</u>

Some candidates erroneously tried to make a call such as <code>Function Push(15)</code> instead of calling and using/storing the result of <code>Push(15)</code>. A number of responses incorrectly used string values "<code>True</code>" / "<code>False</code>" instead of Boolean <code>True / False</code>. Many candidates tried to directly access and use the <code>topStack</code> pointer instead of using the function return value as required in the question.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>12</td><td></td></tr>
<tr><td>10</td><td>a</td><td></td><td>1 mark each

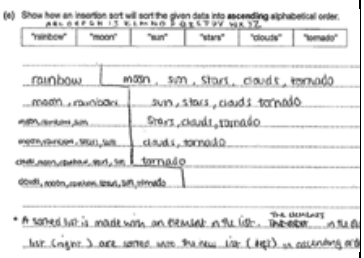
• Compare the first element (rainbow) to search item / clouds
• If it is equal to the search item return index / found
• If it is not equal move to the next element
• Repeat until either search item / clouds is equal / or the end of the list has been reached</td><td>3</td><td>Allow answers by example from the given dataset

<u>Examiner's Comments</u>

Most candidates scored the majority of the marks available and demonstrated a clear understanding of a linear search. Many candidates answered by example with values from the given list.</td></tr>
<tr><td></td><td>b</td><td></td><td>1 mark for: the data is not in order/sorted</td><td>1</td><td><u>Examiner's Comments</u>

Most candidates correctly identified the requirement for data to be sorted/ordered for a binary search to work.
'Organised' was too vague and was not accepted.</td></tr>
<tr><td></td><td>c</td><td></td><td>
<table>
<tr><td>rainbow</td><td>moon</td><td>sun</td><td>stars</td><td>clouds</td><td>tornado</td><td>Marking Guidance</td><td>Marking Guidance</td></tr>
<tr><td>rainbow</td><td>moon</td><td>sun</td><td>stars</td><td>clouds</td><td>tornado</td><td></td><td></td></tr>
</table>
</td><td>5</td><td>If candidate has given <u>descending</u> order, max 4.

MP1, MP3 and MP4 are lines that show a change of values during a pass.</td></tr>
</table>

					MP2 and MP5 do not have to be explicitly given in full if there is a comment to identify no change occur during the pass. Award no marks if not an insertion sort. **Examiner's Comments** Nearly half the candidates achieved full marks and clearly demonstrated the steps involved in an insertion sort for the given data. Some candidates confused insertion sort with either bubble, merge or selection sort, and so scored no marks for not answering the question. Exemplar 3  This exemplar demonstrated the state of the dataset after each pass of the insertion sort. The candidate has separated the sorted and unsorted parts of the list to make the response much clearer.

moon	rainbow	sun	stars	clouds	tornado	Values change	1 Mark
moon	rainbow	sun	stars	clouds	tornado		1 Mark
moon	rainbow	stars	sun	clouds	tornado	Values change	1 Mark
clouds	moon	rainbow	stars	sun	tornado	Values change	1 Mark
clouds	moon	rainbow	stars	sun	tornado		1 Mark

			Total	**9**	

11	a	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
				5	Allow use of this/self or equivalent dependent on language
					
					`public procedure new(value, level)`
					`  this.value = value`
					`  this.level =`

			e.g.
			`class Treasure`
			
			`    private value`
			`    private level`
			
			`    public procedure new(valueP, levelP)`
			`      value = valueP`
			`      level = levelP`
			`    endprocedure`
			`endclass`
					`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
					
					Examiner's Comments
					
					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.

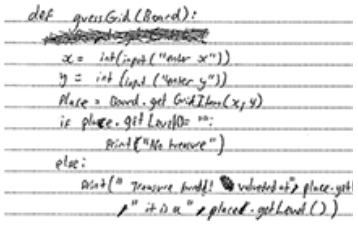
		ii	1 mark each
			• get level method header with no parameter
			• Returning `level` attribute
			
			e.g.
			`public function getLevel()`
			`  return level`
			`endfunction`
				2	Note Python *self* will appear, but no other parameters
					`def getLevel(self):`
					
					Examiner's Comments
					
					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.

	b		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.

	c		1 mark each to max 7	7	Note candidates may attempt to access private

			• 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")`
			`  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`
					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 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  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.
			Total	**21**	
12			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	**2**	

<table>
<tr>
<td>13</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.

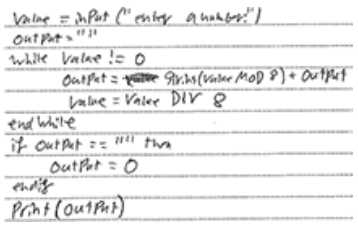
<table>
<tr><th>Function call</th><th>value</th><th>return</th></tr>
<tr><td>recursiveAlgorithm(10)</td><td>10</td><td>29</td></tr>
<tr><td>recursiveAlgorithm(7)</td><td>7</td><td>19</td></tr>
<tr><td>recursiveAlgorithm(6)</td><td>6</td><td>12</td></tr>
<tr><td>recursiveAlgorithm(3)</td><td>3</td><td>6</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>
<tr>
<td>14</td><td></td><td></td>
<td>1 mark each to max 6

• Taking number as input</td>
<td>6</td>
<td>Note candidates can reverse the string before output if they don't</td>
</tr>
</table>

			• 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`
					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.
					
					Examiner's Comments
					
					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

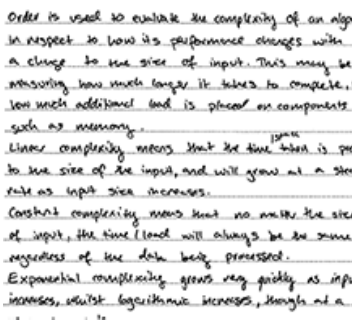
					such as '`denaryNum MOD 8`' which was not given marks. The order required for appending the remainder was frequently incorrect in many of the solutions that were presented, although some candidates did reverse the string at the end of the process if they did '`outString += remainder`' style solutions, 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
					
					 
					Value = inPut ("enter a number")
					outPut = ""
					while Value != 0
					outPut = ~~value~~ String(value MOD 8) + outPut
					Value = Value DIV 8
					end while
					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**	
15	a		**Mark Band 3 – High level** **(7–8 marks)** The candidate demonstrates a thorough knowledge and	9	**AO1: Knowledge and Understanding** **Indicative content**

			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 clear.* **0 mark** No attempt to answer the question or response is not worthy of credit.		• 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

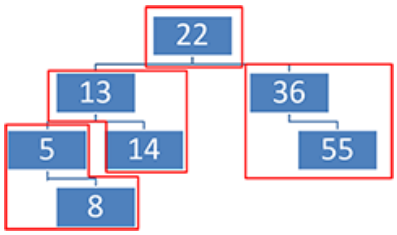
					fast as the data set grows relative to the other algorithms. The space required will also increase, but not insurmountably. This is therefore an efficient algorithm with large data sets compared to algorithm 1 and 2 overall. **AO3: Evaluation** • Number of elements is unknown. Exponential is least appropriate 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. **Examiner's Comments** 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   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.
	b	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 **<u>Examiner's Comments</u>** 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.
		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	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

			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)		be sortonses were phrased in terms of Big O complexities or the numbers of
			Total	**16**	
16	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	**Examiner's Comments** 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	4	Do not allow nodes to be drawn downwards. **Examiner's Comments**

			• 5 and 8 in correct order • 36 and 55 in correct order 		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.
		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.
		vi	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	2	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. **Examiner's Comments** 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

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>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>
<tr><td></td><td></td><td></td><td>Total</td><td>24</td><td></td></tr>
</table>

17	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.

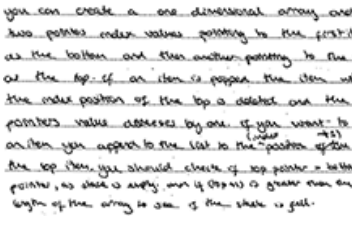
	d		1 mark each to max 4 Check space available in the free list • Check to make sure `freeListPointer` is not Null Add new data item to first free space in free list • Insert new data item at index `freeListPointer` (index 4) Append e.g. • Traverse to / locate the end of the list (index 3 'orange') • 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. **Examiner's Comments** 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.

	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**	
18		i	it can only be accessed within the subroutine/block in which it is declared	1	**Examiner's Comments** Few candidates were able to clearly define the term 'local variable'. The concept of scope of variable s appeared to be poorly understood, with few able to define that a local variable's scope was that of the function/procedure in which it was declared.
		ii	1 mark for benefit e.g. • Increases data integrity • More efficient memory usage • Stops other subroutines accidently altering variable 1 mark for drawback e.g. • Cannot be accessed directly by other subroutines • It has to be passed into a subroutine as a parameter	2	**Examiner's Comments** Some candidates confused the terms local variable and global variable and gave de finitions the wrong way round. A significant number of responses demonstrated conceptual misconceptions. The contents of `dataArray` could be used in other parts of the program if it was passed as a

					parameter to a function/procedure, but could not be referenced directly. Responses such as giving 'can't be used anywhere else in the program' as a disadvantage were, therefore, incorrect.
			Total	**3**	
19	a	i	1 mark per bullet • Start with the first element • Compare it to the number input • If it is equal, return the index / True • If not equal, move to the next element and repeat • Repeat until it is found, or the end of the array is reached • If found, return the index where the data was found • If the end of the list was reached, return -1 / False / "not found" message	5	**Examiner's Comments** A pleasingly high number of candidates demonstrated a clear understanding of how a linear search operated and used clear and accurate language. More candidates were clearer in the way that they articulated the steps of the search than in previous series. However, there were still a number of candidates who gave vague responses such as 'check each valu e until the value input is found' that did not articulate the required steps. Some candidates gave advantages or disadvantages of using a linear search but such responses did not answer the question.
		ii	1 mark per bullet • If the data is not in any order / binary search requires the data to be in order • When the number of items to search is small	1	**Examiner's Comments** Many candidates identified the fact that a linear search can operate on an unordered array whereas a binary search cannot. Some candidates alternatively cited the fact that a linear search could be used when the number of items to search was small, which was valid when qualified by the nu mber of items.

	b		1 mark per bullet to max 4 • (Stack) Pointer points to the last element added to the stack / top of the stack • New data is added to the pointer position / pointer+1… • …check for overflow condition • …pointer is then incremented • Data is removed from pointer/pointer-1 position… • …check for underflow condition • …pointer is then decremented • Elements can be accessed through Push() and Pop() methods that are implemented	4	Note: Answers must relate to an array implementation which means that a stack pointer must be implemented. **Examiner's Comments** Candidates need to be mindful of the fact that an array is a static data structure with a predetermined size. This meant that responses that explained how lists could be used to append and remove items were not given marks. This is an area where candidates who only have programming experience of using lists in Python often struggle. Implementing a stack in an array requires a stack pointer. Those candidates who appreciated this and who clearly had practical experience of modelling push()/pop() operations on a stack were able to articulate how the stack would access array locations at the stack pointer. Exemplar 3  This exemplar was one of the few responses to show a clear insight as to how an array can be used to store and access a stack.

					It makes it clear that a stack pointer is required and goes on to explain how the stack pointer is manipulated when push()/pop() operations are performed. **Misconception** Many candidates were not clear about the difference between an array and a list: • An array is a static structure whose size cannot be changed. • Lists are dynamic and support items being removed or appended.
			Total	**10**	
20		i	355	1	**Examiner's Comments** Many candidates did not appreciate that a bubble sort will require a maximum of n -1 passes in the worst case since the first item in the list will be in position after that number of passes and so would not require an additional pass. The most common incorrect responses were 356, and 3562 which confused the worst case time complexity $O(n^2)$ with the number of passes.
		ii	Insertion sort	1	Accept any valid sorting algorithm e.g. Merge sort, Quick sort

					Examiner's Comments Nearly all candidates could identify an additional sorting algorithm with the most common response being 'Insertion sort'.
			Total	**2**	

2 1	a		1 mark per bullet to max 6
			
			• function header taking parameter
			• looping appropriately e.g. until value is 0
			• dividing by 2 and finding remainder e.g. MOD
			• adding 1 or 0 correctly
			• …appending to a value to be returned / final string reversed
			• reducing value to use within loop
			• returning calculated value
			
			e.g.
			`function toBinary(denary)`
			`  binaryValue=""`
			`  while denary > 0`
			`  temp = denary MOD 2`
			`  if temp == 1 then`
			`  binaryValue = "1" + binaryValue`
			`  else`
			`  binaryValue = "0" + binaryValue`
			`  endif`
			`  denary = denary DIV 2`
			`  endwhile`
			`  return binaryValue`
			`endfunction`
				6	Award a recursive algorithm as equivalent
					
					Examiner's Comments
					
					Very few candidates were able to produce a working function, but many gained some marks.
					
					Some candidates had little idea of the concept of a function and struggled to define one. Many omitted the definition statement or omitted the required parameter and then asked for user input instead.
					
					The standard of pseudocode/code was quite weak. Indentation of constructs was often missing or hard to follow. Mid-range marks were achieved when candidates effectively utilised MOD to determine if the remainder on division by two was odd or even, and then using DIV to find the next term in the sequence.
					
					More successful responses demonstrated an ability to problem solve, think logically, and present clear working functions.

	b		1 mark per bullet to max 4 • taking value as input • looping until valid between 1 and 255 • calling function with correct parameter	4	Allow other checks for a valid number. For example `denary.isInteger ==`

			• outputting return value
			
			`denary = -1`
			`while denary < 1 or denary > 255`
			`  denary = input("Enter denary value between 1 and 255")`
			`endwhile`
			`print(toBinary(denary))`
					`False`
					
					Examiner's Comments
					
					Candidates found Question 4 (b) easier to approach than Question 4 (a). Most could write pseudocode to accept a user input. When validating the input to be a value between 1 and 255, there was incorrect use of relational operators with off -by-one errors on occasion. There was also incorrect use of logical operators where or was used instead of *and*, and vice-versa. When candidates called `toBinary(inputVal)`, the result was not always stored for later use.
					
					Assessment for learning
					
					When preparing candidates for the examination, they will benefit from a wide range of programming experience.
					
					Questions such as 4 (a) and 4 (b) present an ideal opportunity for developing coded solutions to test and discuss before looking at how the algorithms could be presented as pseudocode.

			Total	**10**	
22	a	i	sequence	1	**Examiner's Comments** Nearly all candidates correctly identified 'sequence' as the correct construct.

<table>
<tr><td></td><td></td><td>ii</td><td>selection / branching</td><td>1</td><td><u>Examiner's Comments</u>

Nearly all candidates correctly identified 'selection' as the correct construct.</td></tr>
<tr><td></td><td>b</td><td></td><td>1 mark each to max 2

• total
• smallest
• largest
• x
• dataArray</td><td>2</td><td><u>Examiner's Comments</u>

Most candidates correctly identified two variables from the given code.</td></tr>
<tr><td></td><td>c</td><td></td><td>1 mark for each line and correction

• Line 01
<code>total = 0</code>
• Line 02
• <code>smallest = dataArray[0]</code>
• Line 04
<code>for x = 0 to 19</code> (accept 20)
• Line 07
<code>if dataArray[x] > largest then</code>
• Line 14
<code>print("Average = " + total / 20)</code></td><td>4</td><td>Do not award a mark for the line number alone without correction.

<u>Examiner's Comments</u>

Most candidates were given some marks, but fewer achieved full marks.

The context of the question indicated that the numbers input were positive integers, but candidates did not always appreciate this.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>8</td><td></td></tr>
<tr><td>2
3</td><td>a</td><td></td><td>1 mark per bullet

• 1st swap of 5 and 3
• Remainder of first pass
• Pass 2
• Pass 3

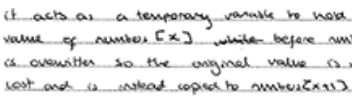
<table><tr><td>1</td><td>5</td><td>3</td><td>9</td><td>2</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>9</td><td>2</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>2</td><td>9</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>2</td><td>7</td><td>9</td></tr><tr><td>1</td><td>3</td><td>2</td><td>5</td><td>7</td><td>9</td></tr></table>End of pass 1</td><td>4</td><td>Candidates do not need to show each swap, so if the candidate has clearly shown the end of pass 1, they have met the first two marking points.

Marks can be awarded for correctly showing the results of each pass.

<u>Examiner's Comments</u>

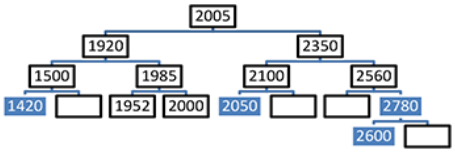
Most candidates clearly showed the steps that would take place in a bubble sort for the date given, with most achieving full marks. Some candidates did not</td></tr>
</table>

			<table><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td>2</td><td>3</td><td>5</td><td>7</td><td>9</td></tr></table> End of pass 2 End of pass 3		explicitly label each pass as required in the question, but marks were given where the passes could be implied. Few candidates described the principles of a bubble sort instead of applying it to the data given.
	b	i	1 mark per bullet • by reference will reorder the contents of the array • …so the new order can be accessed by the main program / so will be saved when the procedure ends • by value will change the array only in this procedure • … and so would need to return the array.	3	**Examiner's Comments** Many candidates struggled to go beyond recall of definitions for calling by reference and calling by value and struggled to apply it to the code given and to provide a detailed explanation. The bubble sort was defined as a procedure and not a function, so if numbers had been passed by value, a copy of the array would have been passed, and any changes made would not have been kept after the procedure had completed execution.
		ii	A loop that repeats a fixed / set number of times	1	**Examiner's Comments** Most candidates could accurately define a 'count controlled loop' as one that repeated a predefined number of times, although some candidates gave an ambiguous response that was equally applicable to a conditional loop.
		iii	• To temporarily hold a value (for numbers[x])… • …while it is being transferred from one position to another… • ….in the array numbers • To stop values over writing each other	3	**Examiner's Comments** Many candidates found it difficult to explain the purpose of the `holdValue` variable in context. Where candidates achieved some of the marks, they most

					frequently identified `holdValue` as a temporary store that was required to prevent accidental overwriting of data during the swap process. Relatively few were able to accurately describe how the variable allowed the contents of `dataArray[x]` and `dataArray[x+1]` to be swapped. Exemplar 1  This exemplar very clearly states exactly how and why the variable `holdValue` is required.
		iv	• Add a (second outer) loop • That will repeat for each pass / repeat until the flag is set to true at the end of a pass	2	**Examiner's Comments** Many candidates found it challenging to apply knowledge of a bubble sort to the code given. While a pleasing number identified the need to have an outer loop, there were far fewer who were able to expand on this to explain that this was required to repeat the process for the requi red number of passes, or until no swaps had occurred during a pass.
			Total	**13**	
24			1 mark per bullet • identifier `cards`… • …with 2 dimensions	2	**Examiner's Comments** Many candidates gained a mark by initialising the identifier `cards`, but fewer gained the second mark for correctly setting it to be a two-dimensional

					structure. Many obscure forms of syntax were observed, but marks was given if it was clear that the structure was two-dimensional, however, for many responses, it was clear that a one-dimensional list had been initialised.
			Total	**2**	
25	a	i	Line number `5`	1	**Examiner's Comments** Nearly all candidates gave the correct answer line `5`.
		ii	1 mark per feature • A function that calls itself / a function that is defined in terms of itself • …has a base case (that terminates the recursion)	2	**Examiner's Comments** Most candidates knew that a recursive algorithm is self-referential and calls itself, but some candidates were too vague specifying that it 'calls a function'. Candidates often found it harder to gain the second mark and some gave answers not related to the question such as explanations of how recursion uses stack frames during execution. Technical vocabulary is important, and some candidates did not make it clear that recursion has a base case. Those that stated a stopping/terminating condition needed to qualify their response to say that these conditions stopped/halted the recursion. Where candidates just wrote 'stopping condition' it was too vague as it was unqualified since they could have been talking about any conditional loop.
	b			5	**Examiner's Comments**

			<table><tr><th>Function call</th><th>number</th><th>return</th><th>Marking Guidance</th></tr><tr><td>calculate(5)</td><td>5</td><td>15</td><td>1 Mark</td></tr><tr><td>calculate(4)</td><td>4</td><td>10</td><td>1 Mark</td></tr><tr><td>calculate(3)</td><td>3</td><td>6</td><td>1 Mark</td></tr><tr><td>calculate(2)</td><td>2</td><td>3</td><td>1 Mark</td></tr><tr><td>calculate(1)</td><td>1</td><td>1</td><td>1 Mark</td></tr></table>		Many candidates continue to struggle with recursion as a concept and so had little idea how to trace and unwind a call to a recursive function. Some got to the last call and the base case and could return 1 or unwind one step further to gain the first 2 marks. Fewer achieved all 5 marks.
	c		calculate(10)	1	**Examiner's Comments** Those candidates who scored full marks in 7b had little difficulty giving the correct call calculate(10). Some candidates just wrote 10, which did not answer the question. Those who showed little understanding of recursion in 7b rarely gave the correct answer in 7c.
			Total	**9**	
26	a		2005	1	**Examiner's Comments** Most candidates correctly identified the root node as 2005.
	b		1 mark for each to max 2 • 1500 • 1952 • 2000 • 2100 • 2560	2	**Examiner's Comments** Most candidates correctly identified valid leaf nodes, but a significant number erroneously gave 1920 and 2350 as the child nodes of the root instead of identifying leaf nodes.
	c		1 mark for each in the correct place • 1420 • 2050 • 2780 • 2600	4	**Examiner's Comments** Many candidates scored at least 2 marks, but many erroneously inserted 2600 before 2780, or just put 2600 as the left child node of 2560. A common mistake was to use straight lines for new child

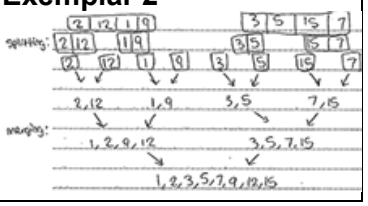
					nodes rather than clearly indicating whether the new child node was a left/right child of the parent node.
	d		**Mark Band 3 – High level** **(7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of search traversals; the material is generally accurate and detailed. The candidate 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 search traversals; 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 search traversals 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 marks** No attempt to answer the question or response is not worthy of credit.	9AO1.1 (2)AO1.2 (2)AO2.1 (2)AO3.3 (3)	**AO1: Knowledge and Understanding** **Indicative content** • Breadth first takes first value then visits all nodes connected to it. It then takes all nodes connected to those nodes. • Depth first goes to the left node, this becomes a new tree. It continues going to the left until a leaf. It then returns this, then goes right and repeats from the start. Follow left, follow right, take root. **AO2: Application** • Breadth will return 2005 1920 2350 1500 1985 2100 2560 (1420) 1952 2000 (2050) (2780) (2600) • Post-order / Depth will return (1420) 1500 1952 2000 1985 1920 (2050) 2100 (2600) (2780) 2560 2350 2005 **AO3: Evaluation** **Evaluations may vary and include one or more of the following points:** • Breadth is more efficient when the data searched for is closer to the root.

					• Depth is more efficient when data to be search for is further down. • Depth memory requirement is linear • Depth can be written recursively to aid understanding. • Breadth in general is better time complexity • In large trees depth may never return a value Candidates are not expected to know the complexities for the search traversals, however credit should be awarded if candidates choose to include these. Limit to band 2 if there is no evaluation of BFS/DFS **Examiner's Comments** Many candidates achieved some marks, but a few did confuse Breadth-first search (BFS) and Depth-first search (DFS), getting them the wrong way round. Most had a much clearer understanding of BFS than DFS and were able to score marks in Level 1 for showing how a BFS would be carried out. Fewer managed to accurately show how a DFS would work. While there was an understanding that DFS would traverse leftward to the lowest leftmost node, descriptions of back tracking were not given as many marks. A common

					mistake made with DFS was to output the nodes in the order they were first visited rather than the order in which they are output. Few could describe the mechanics of the algorithms with BFS using a queue and DFS using a stack. Those who did achieved the top of Level 2. Very few candidates could give exemplar uses of BFS or DFS (e.g. deleting nodes in a tree) or compare cases where they might be used (e.g. distance of target node(s) from root), with very little evaluation. This meant there were very few Level 3 responses seen.
			Total	**16**	

27	a		1 mark for each completed statement
			
			`arrayLength = 50 / numberArray.length`
			`tempValue =0`
			`do`
			`  flag = false`
			`  for y = 0 to arrayLength - 2`
			`  if numberArray[y] > numberArray[y + 1] then`
			`  tempValue = numberArray[y]`
			`  numberArray[y] = numberArray[y + 1]`
			`  numberArray[y + 1] = tempValue`
			`  flag = true`
			`  endif`
			`  next y`
			`until flag == false`
				5	Note if numberArray - 1 / 49 used, then for loop for y will need to be 0 to arrayLength - 1
					
					Allow other suitable valid identifier in place of tempValue e.g. temp
					
					Examiner's Comments
					
					Many candidates scored at least 1 mark for correctly initialising the `arrayLength`, although some erroneous `length.arrayLength` rather than `arrayLength.length` answers were seen. If candidates are assuming the existence of inbuilt methods they should reference them in the correct OOP way.

					Very few candidates achieved the second marking point for the loop. A very common off-by-one error was seen with the value 1 given. If `arrayLength` was set to 50 this would cause a run time error for an out-of-bounds reference when the loop ran. Some candidates tried to swap the array values directly in a Pythonic style that was not suitable for a pseudocode solution in context, and some used incorrectly formatted variable identifiers. **Variable identifiers** Valid identifiers must be single words. In a number of instances, it was clear that `tempValue` was given as `temp Value`. Where candidates give an answer that clearly has spaces within an intended identifier name no marks will be given.
	b		1 mark for each stage shown • Splitting into individual items 2 12 1 9 3 5 15 • Combining in pairs 2 12 1 9 3 5 7 • Merge pairs	4	Do not award a mark for the final stage, unless candidate has shown the previous sorting stage(s). **<u>Examiner's Comments</u>** Many candidates presented very clear solutions that used the values in `numberArray` to construct clearly annotated diagrams. Textual prose responses tended to either not refer at all to the given data set, or mentioned it only partially, so lost marks. There was occasional confusion with other

			1 2 9 12 3 5 7 15 Merge for final 1 2 3 5 7 9 12 15		sorting algorithms, but this was seen relatively infrequently, Where errors were made candidates did not split the data to the atomised level. Many did not fully understand how a merge sort works by merging two separate lists of ordered values together but performed in-place sorts in sub-lists. **Misconception** The merge phase in a merge sort takes two separate lists that are already ordered. Values in those two separate lists are taken and merged in sequence to form a new list. E.g. List 1 [1, 9] and List 2 [2, 10] are taken and merged 1 from List 1, 2 from List 2, 9 from List 1 and then 10 from List 2, into a new List. Values in lists [1, 9] and [2, 10] are not placed in a list [1, 9, 2, 10] and then sorted in-situ. A significant number of candidates thought that a merge sort split lists up until values were in pairs and showed [15, 7] going directly to [7, 15] (i.e. an in-situ sort) without first being atomised into two separate lists of [15] and [7] and then being merged to give [7, 15]. **Exemplar 2** 

					Candidate responses to questions that require 'showing' how data sets are ordered by sorting algorithms are best tackled by using clearly annotated diagrams such as this exemplar response.
	c		**Mark Band 3 – High level** **(9-12 marks)** The candidate demonstrates a thorough knowledge and understanding of sorting algorithms; 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** **(5-8 marks)** The candidate demonstrates reasonable knowledge and understanding of sorting algorithms; 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** **(1-4 marks)** The candidate demonstrates a basic knowledge of sorting algorithms 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*.	12 AO1.1 (3) AO1.2 (3) AO2.1 (3) AO3.3 (3)	**AO1: Knowledge and Understanding** **Indicative content** • Merge sort splits data into individual lists and merges • Insertion makes first value sorted list, then inserts each item into the sorted list • Bubble sort looks through each item in turn, number of items times **AO2: Application** • Merge uses more memory as new lists are needed. Insertion and Bubble need constant memory. • Bubble and Insertion have the best best-times, both O(n) because they run through the data once. merge sort requires a minimum number of stages so best case is longer (O(n log(n)) • Merge average is the same as best. Insertion and Bubble has average o(n^2).

			0 marks No attempt to answer the question or response is not worthy of credit.		• Worst time merge has same has best and average because same number of stages are needed. Bubble sort and insertion all have worse O(n^2) **AO3: Evaluation** • There are a small number of elements (10) therefore a bubble sort of insertion would be better space wise because no further space is needed. • Merge would not need excessive amounts of more memory as there are only a small number of elements. • Time complexity, there is a small number of elements therefore Bubble and Insertion may be preferable. Differences are unlikely to be significant, so either would be more appropriate. **<u>Examiner's Comments</u>** Most candidates could describe the basic elements of each of the bubble, merge and insertion sort, although some had difficulty remembering insertion sort and confused it with quick sort. Many candidates

					struggled to show accurate knowledge of the Big O time complexity values for best/average/worst case for the three algorithms, which is a learnt response. Some did give very good examples of where bubble and insertion sort would give best/worst case times. Very few candidates could cite space complexity, and where they did a number thought bubble/insertion were space complexity O(n) because there are n elements rather than O(1) which means constant with no extra memory overhead. There was some evaluation of the number of items being sorted in most cases. Overall, many responses were clustered in Level 2, but a pleasing number of candidates achieved Level 3 with clear and detailed descriptions, accurate Big O values, and an evaluation of the size of the data set being sorted.
			Total	**21**	
28	a		1 mark e.g. • Symbols are used to represent the address • The edges represent possible connections between addresses not the actual physical routes	1	Allow other suitable answers that are in context of the problem **<u>Examiner's Comments</u>** Very few candidates were able to give suitable answers within the context of the problem. The question was asking why the graph in Fig 5 was a visualisation. Few

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>candidates identified that it was because the letters at the nodes represented delivery addresses, while the weights on the edges represented the road distances between the addresses. Most candidates gave descriptions of visualisation in general rather than answering in context.</td></tr>
<tr><td></td><td>b</td><td>i</td><td>
<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 / -</td><td>1 Mark</td></tr>
<tr><td>B</td><td>3</td><td>A</td><td rowspan="2">1 Mark</td></tr>
<tr><td>C</td><td>13</td><td>E</td></tr>
<tr><td>D</td><td>10</td><td>B</td><td rowspan="2">1 Mark</td></tr>
<tr><td>E</td><td>6</td><td>B</td></tr>
<tr><td>F</td><td>9</td><td>E</td><td rowspan="2">1 Mark</td></tr>
<tr><td>G</td><td>16</td><td>F</td></tr>
<tr><td>H</td><td>~~21~~ 19</td><td>~~D~~ G</td><td>1 Mark</td></tr>
</table>
Final Path = A,B,E,F,G,H, Distance = 19 (1 Mark)
</td><td>6</td><td>Order of previous nodes visited must be clear

Note that nodes in the table do not have to be given in alphabetical order by candidates

<u>Examiner’s Comments</u>

Most candidates gave the final path and the total distance correctly by inspection if nothing else. All nodes are initially set to infinity, so A is updated to 0 and has a distance 0 from A as the start node and many candidates missed this. Those who gave an answer by inspection wrote ABEFGH without knowing Dijkstra’s algorithm but gained some marks by giving the distances to BEFGH along the way, but distances to nodes C and D were omitted. Few candidates clearly showed that the initial calculation for the path distance to H (from D, distance 21) was later updated and overwritten with the more optimal path length from G with distance 19.</td></tr>
<tr><td></td><td></td><td>ii</td><td>1 mark per bullet
Similarity:</td><td>2</td><td>Must contain a similarity and a difference for both marks.</td></tr>
</table>

			• Both always find the shortest route • Both are pathfinding algorithms Differences: • A* is (usually) more efficient / dijkstra's is (usually) slower • A* uses heuristics to find a solution faster / Dijkstra's does not use heuristics		**Examiner's Comments** Those candidates who scored well in c(i) frequently gained full marks for describing the similarities and differences between Dijkstra's algorithm and the A* algorithm. The most common responses were that both give the shortest path and that A* uses heuristics.
			Total	**9**	
29		i	The array/data must be in order/sorted	1	**Examiner's Comments** Most candidates successfully identified that data must be ordered as a precondition for a binary search. A few candidates were too vague giving unqualified answers such as 'must be sorted', without specifying what had to be sorted.
		ii	1 mark per bullet • Compare the search item with the first value • ….then compare the search item with the next value • ….repeat the above process until either • ….the end of the array has been reached or • ….the search item is found and then stop • ….then return the array position / return -1 / False if not found	4	Not all mark points are dependent, but points awarded must following logically in sequence. **Examiner's Comments** Many candidates scored some marks for describing the steps involved in a linear search, but relatively few presented a comprehensive and detailed description for full marks. Many responses used vague language such 'checking if the value is found' without explaining that a comparison between the current term and the target search value must be performed. Other examples of vague

					language use included responses such as 'keep going until you find what you're looking for', which begged the question, how do you know when you've found what you're looking for? Common errors included candidates who mistakenly described a binary search, and those who did not answer the question. Examples of not answering the question included giving properties of a linear search such as its linear run time or the fact that it can be run on an unordered data set.
			Total	**5**	
30	a		1 mark per bullet up to a maximum of 3 marks, e.g.: • To start from the beginning / first booking slot • Search each slot in order / sequentially • Search until the first empty slot is found	3 (AO2.2) (3)	**Examiner's Comments** This question was generally not well answered. Candidates found it difficult to articulate a clear and concise set of steps that take place during a linear search. Many responses were vague and not directly related to a linear search being performed on the data presented in the table.

	b		1 mark per bullet up to a maximum of 7 marks, e.g.:
			
			• Defining the `findFirst` function correctly
			• Suitable logic for checking the first time slot
			• Suitable logic for checking the next time slot…
			• Suitable loop to check all time slots
			• Suitable logic for returning the available time slot
			• Suitable logic for returning -1 if no time slots available
			• Suitable use of variable names and indentation
				7 (AO3.1) (7)	**Example solution:**
					`function findFirst()`
					`  count = 0`
					`  do`
					`  found = false`
					`  if customerID[2, count] == "" then`
					`  found = true`
					`  else`
					`  count = count + 1`
					`  endif`

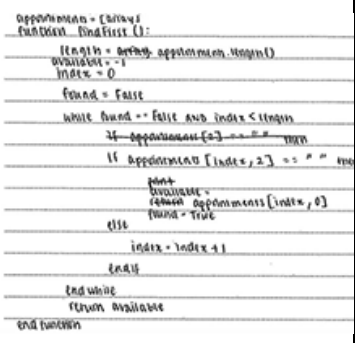
```
    until count ==
10 or found == true
    if found == True
        return
customerID[0,count]
    else
        return -1
    endif
endfunction
```

There are many different ways that this function could have been achieved. Therefore other alternative methods should be given credit.

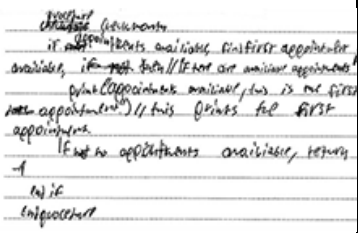
Examiner's Comments

A number of candidates started by defining a procedure rather than defining a function. Indentation was not always consistent with the constructs being used. Candidates seemed to have difficulty with referencing two dimensional structures. Algorithms that require two dimensional data frequently appear on this paper and candidates need to have extensive practical programming experience solving problems using these structures.

Exemplar 2



A well-structured pseudocode response that uses indentation and variable naming well. The

					logic of the loop to check each item in sequence for an empty entry is clear, as is the indexing into the 2-Dimensional table structure. **Exemplar 3**  To contrast with Exemplar 2, this exemplar shows a lack of pseudocode with a response written mostly in prose English, with inconsistent indentation and a lack of clear variable naming. **Assessment for learning** Candidates frequently benefit from having extensive practical programming experience when answering pseudocode questions. Past paper questions can provide a context for problems for implementation. For this question candidates could either be asked to code a two dimensional table structure for the data in **Fig 1** and to then implement the linear search, or they code be provided with a scaffolded partially complete solution with the table defined and the stem of a function given.

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>Routinely taking past paper questions that can be practically implemented is an effective way to make sure that candidates have relevant experience.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>10</td><td></td></tr>
<tr><td>3 1</td><td>a</td><td></td><td>1 mark for each correct row up to a maximum of 4 marks.
<table><tr><td>89</td><td>25</td><td>75</td><td>37</td><td>45</td><td></td></tr><tr><td>25</td><td>89</td><td>75</td><td>37</td><td>45</td><td>1 mark</td></tr><tr><td>25</td><td>75</td><td>89</td><td>37</td><td>45</td><td>1 mark</td></tr><tr><td>25</td><td>75</td><td>37</td><td>89</td><td>45</td><td>1 mark</td></tr><tr><td>25</td><td>75</td><td>37</td><td>45</td><td>89</td><td>1 mark</td></tr></table></td><td>4
(AO2.1)
(4)</td><td>Marks should be awarded for correct swapping of adjacent items that are out of order. Therefore if the previous step is incorrect but the candidate has followed through with the correct answer then marks should be awarded.

<u>Examiner’s Comments</u>
There were a number of responses that included clear diagrams illustrating the first pass of a bubble sort on the given data. Very few candidates confused bubble sort with other types of sorting algorithm so most achieved full marks for this question.</td></tr>
<tr><td></td><td>b</td><td>i</td><td>• Line 5</td><td>1
(AO2.1)
(1)</td><td>Allow line 4 and 12

<u>Examiner’s Comments</u>
Most candidates had little difficulty identifying line <code>05</code> as the obvious response, but some candidates did choose the decision taking place at the end of the <code>do..until</code> loop in line <code>12</code>.</td></tr>
<tr><td></td><td></td><td>ii</td><td>numbers</td><td>1
(AO2.1)
(1)</td><td><u>Examiner’s Comments</u>
Nearly all candidates correctly identified <code>numbers</code> as the correct parameter name.</td></tr>
<tr><td></td><td></td><td>iii</td><td>1 mark per bullet up to a maximum of 2 marks, e.g.:

• To temporarily hold data…
• …To allow the contents of two variables to be swapped</td><td>2
(AO1.2)
(2)</td><td><u>Examiner’s Comments</u>
Candidates found it difficult to articulate a response that broke the purpose of the <code>temp</code></td></tr>
</table>

			• …to ensure that data is not overwritten		variable down in lines `06` to `08` into logical steps with reasons. Many candidates identified it as a temporary store, but few could explain that it allowed the contents of the two array positions to be swapped without erroneously overwriting either value.
		iv	1 mark per bullet up to a maximum of 2 marks, e.g.: • signifies whether or not any swaps have been made in a pass • if still set to true at the end of a pass, then the list is sorted	2 (AO1.2) (2)	**Examiner's Comments** Many responses to this question were generalised answers that stated that the `sorted` variable determined whether the data was sorted or not. This could be an indication that candidates did not have practical experience of implementing a bubble sort with a swap flag. Most candidates did not appreciate the function of the variable during each pass of the bubble sort. Candidates need to be well versed in the different ways of implementing a bubble sort.
			Total	**10**	
32	a	i	1 mark per box up to a maximum of 3 marks. • **Select puzzle** and **display blank grid** (below new game) • **Select box** and **change colour of boxes** (below play game) • **Compare to answer** and **display correct/incorrect** (below check answer) e.g. 	12AO1.1 (2)AO1.2 (2)AO2.1 (23AO3.3 (5)	
		ii	1 mark per bullet up to a maximum of 2 marks, e.g: e.g. • Splits the problem into smaller chunks • Smaller problems are more manageable	2AO1.1 (1)AO1.2 (1)	

<table>
<tr><td></td><td></td><td></td><td>

Smaller problems are easier to solve
To see where code can be reused in the solution
To split tasks between different programmers

</td><td></td><td></td></tr>
<tr><td></td><td></td><td>iii</td><td>1 mark for input, 1 for process 1 for output
e.g.
Input:

• Clicking a box

Process:

• Generating new puzzle
• Checking if block is black
• Changing block to white

Output:

• Grid with coloured squares</td><td>3AO2.2 (3)</td><td></td></tr>
<tr><td></td><td>b</td><td>i</td><td>1 mark for each correctly completed statement up to a maximum of 5 marks:

<pre>
01 function countRow(puzzle:byref,
   rowNum:byval)
02   count = 0
03   output = " "
04   for i = 0 To 4
05   if puzzle[rowNum, i] == 1
       then
06   count = count + 1
07   elseif count >= 1 then
08   output = output +
         str(count) + " "
09   count = 0
10   endif
11   next i
12   if count>= 1 then
13   output=ouput+str(count)
14   elseif output == "" then
15   output = "0"
16   endif
17   return output
18 endfunction
</pre>
</td><td>5AO2.2 (2)AO3.2 (3)</td><td>Accept

for i = 0 to row.length-1

for i = 0 to row.length

for i=0 to 5</td></tr>
<tr><td></td><td></td><td>ii</td><td>1 mark per bullet up to a maximum of 2 marks, e.g:

• Initialise the variable output…
• …with a space</td><td>2AO1.2 (1)AO2.2 (1)</td><td></td></tr>
</table>

			• …for use later on in the code… • …So it can be used for concatenation later in the code … • …to avoid an error being generated		
		iii	1 mark per bullet up to a maximum of 3 marks, e.g: • check the value stored in each index • check whether it is at the end of a row • check whether each row has been given an output or not	3AO2.2 (3)	

		iv	1 mark per bullet up to a maximum of 6 marks:
			
			• Procedure heading for displayRowAnswer
			• …taking puzzle as parameters
			• Nested loops through all array elements
			• …outputting all rows
			• … at the end of each row calling countRow ….
			• …..with parameters puzzle and the current loop counter
			
			e.g.
			`procedure displayRowAnswer(puzzle)`
			`  for i = 0 To 4`
			`  for j = 0 To 4`
			`  print(puzzle[i, j] + " ")`
			`  next j`
			`  print (" " + countRow (puzzle, i))`
			`  next i`
			`endprocedure`
				6AO2.2 (3)AO3.2 (3)	Accept
					
					for i = 0 to **row.length-1**
					
					for i = 0 to **row.length**
					
					for i=0 to **5**

		v	1 mark for clearly identifying each error and giving the correction. • Line 01 needs `answerGrid` as parameter • Line 04 == should be != • Line 08 should be `next row`	3AO2.1 (3)	Do not award marks for line numbers alone without stating the error. Consider 1 mark for not changing line 04 but changing 05 to true and 09 to False
	c		**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of local and 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. The candidate is able to weigh up the use of both local and global variables which results in a supported and realistic	9AO1.1 (2)AO1.2 (2)AO2.1 (2)AO3.3 (3)	**AO1: Knowledge and Understanding** **Indicative content** Local variables: • Scope within the module defined within • Cannot access externally unless passed as

			judgment as to whether it is possible to use them in this context. *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 knoledge and understanding of local and 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 makes a reasonable attempt to come to a conclusion showing some recognition of influencing factors that would determine whether it is possible to use local and global variables in this context. *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 local and 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 nothing more than an unsupported assertion. *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 marks** No attempt to answer the question or response is not worthy of credit.		parameter, or returned from function • When module is exited, memory of variable is freed Global variables: • Scope within the entire program • Can access from anywhere • Retained in memory permanently ByRef Points to location of variable ByVal Sends the value **AO2: Application** • If global the arrays can be accessed from all modules by direct reference • If local to the main, the arrays will need to be passed as parameters byreference • Can send ByVal – but not always possible with arrays in some languages • Modules are self contained and then can be reused in other programs he wants to create without needing to take the global variables with them **AO3: Evaluation** e.g. • +ve Local = memory efficient • +ve Global = easier programming,

					simpler to follow, easier to debug • -ve Global = memory inefficient, not good programming technique • -ve Local = more difficult to trace/debug/follow where the values are passed • Relatively small program – don't know about overall plan for it, it might not be memory intensive, unlikely anyone else is going to access/amend e.g. use as a library – therefore global would not waste significant resources
	d		1 mark per bullet to max 4 e.g. • Make use of random numbers • Generate an x/horizontal size for the grid • Generate a y/vertical size for the grid • Loop through each row/column • …generate a number between 0 and the number of rows/columns (depending on MP4 answer) • Loop through each box • …generate a 1 or 0 to store in it	4AO2.1 (2)AO2.2 (2)	
			Total	**40**	

33	a		1 mark for each completed statement up to a maximum of 5 marks:
			`private numberFloors`
			`private width`
			`private` **`height`**
			`public procedure new(pFloors, pWidth, pHeight)`
			`numberFloors =` **`pFloors`**
			`width = pWidth`
			`height = pHeight`
			`endprocedure`
				5AO2.2 (3)AO3.2 (2)	Accept other specific language conventions that would correctly achieve the same outcomes.

			`public function getNumberFloors()`
			`  return **numberFloors**`
			`endfunction`
			`public function setNumberFloors(pFloors)`
			`  if pFloors >= 1 then`
			`  numberFloors = **pFloors**`
			`  return true`
			`  else`
			`  return **false**`
			`  endif`
			`endfunction`
			`endclass`		

	b		1 mark per bullet up to a maximum of 6 marks:
			• Class declaration for house with inherits building
			• Declaring bedrooms and bathrooms as private
			• New declaration …
			• … with all **five** parameters
			• Calling super constructor / equivalent with floors, width and height set
			• Setting bedrooms and bathrooms
			e.g.
			`class house inherits building`
			`  private bedrooms, bathrooms`
			`  public procedure new(pFloors,`
			`  pWidth, pHeight, pBedrooms,`
			`  pBathrooms)`
			`  super.new(pFloors, pWidth,`
			`  pHeight)`
			`  bedrooms = pBedrooms`
			`  bathrooms = pBathrooms`
			`  endprocedure`
			`endclass`
				6AO2.1 (1)AO2.2 (2)AO3.2 (3)	

	c		1 mark per bullet up to a maximum of 4 marks:
			• Procedure header taking parameter
			• Adding parameter to array
			• …at position `numberBuildings`
			• Incrementing `numberBuilding`
			e.g.
			`procedure newbuilding(pBuilding)`
			`  buildings[numberBuildings] =`
			`  pBuilding`
			`  numberBuildings =`
			`  numberBuildings + 1`
			`endprocedure`
				4AO2.1 (1)AO3.2 (3)	

	d		1 mark per bullet up to a maximum of 4 marks: • Creating a new instance of house with identifier `houseOne`	4AO2.1 (1)AO3.2 (3)	

			• …with the correct parameters
			• Creating a new instance of houseRoad named `limeAvenue`
			• …sending `houseOne` as parameter to the constructor
			
			e.g.
			`houseOne = new house(2, 8, 10, 3, 2)`
			`limeAvenue = new houseRoad(houseOne)`		

			Total	**19**	
34			1 mark for any example e.g. • Data is not sorted • Item you are looking for is the first item in the list • Small number of items	1AO1.2 (1)	
			Total	**1**	
35	a		1 mark per bullet, each must be applied correctly to data • Choose a pivot / identify start and end pointers • Compare each element to the pivot… / compare start and end pointers • Put items < pivot in the left sublist • Put items > pivot in the right sublist • Choose a pivot in each sublist • If start pointer is larger than end pointer… • …then swap data items around • And repeat the process until each item becomes a pivot	5AO1.2 (2)AO2.2 (3)	
	b		1 mark per bullet to max 2 • decomposing data sets into smaller subsets • and then sorting each split subset • until each subset is sorted • and then combining the subsets to provide a solution	2AO1.1 (1)AO2.1 (1)	
			Total	**7**	
36			1 mark per bullet to max 6 • Visit root node **M** • Visit **E** and **S** • Visit **C** and **J** (from E) • …then **P** and **V** (from S) • Visit **G** and **K** (from J) • Visit **L** (from **K**)	6AO1.2 (1)AO2.1 (3)AO2.2 (2)	
			Total	**6**	

<table>
<tr>
<td>3
7</td>
<td>a</td>
<td>i</td>
<td>1 mark for each number/statement up to a maximum of 6 marks:

<pre>
function binarySearch(dataArray:byref,
upperbound, lowerbound, searchValue)
  while true
    middle = lowerbound +)
    ((upperbound - lowerbound))
    DIV 2)
    if upperbound < lowerbound
    then
      return -1
    else
      if dataArray[middle] <
      searchValue then
        lowerbound = middle + 1
      elseif dataArray[middle] >
      searchValue then
        upperbound = middle - 1
      else
        return middle
      endif
    endif
  endwhile
endfunction
</pre>
</td>
<td>6AO1.2 (2)AO3. 3 (4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ii</td>
<td>Do…until / repeat…until / post condition</td>
<td>1AO1.2 (1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td></td>
<td>1 mark for each tick up to a maximum of 6 marks
Worst-case space complexity:

<table>
<tr><th></th><th>Binary search</th><th>Linear search</th></tr>
<tr><td>O(log(n))</td><td></td><td></td></tr>
<tr><td>O(1)</td><td>✓</td><td>✓</td></tr>
<tr><td>O(n)</td><td></td><td></td></tr>
</table>

Best-case space complexity:

<table>
<tr><th></th><th>Binary search</th><th>Linear search</th></tr>
<tr><td>O(log(n))</td><td></td><td></td></tr>
<tr><td>O(1)</td><td>✓</td><td>✓</td></tr>
<tr><td>O(n)</td><td></td><td></td></tr>
</table>
</td>
<td>6AO1.1 (6)</td>
<td></td>
</tr>
</table>

Average time complexity:

	Binary search	Linear search
O(log(n))	✓	
O(1)		
O(n)		✓

			Total	**13**	
38		i	1 mark per bullet to max 2 e.g: • A rule of thumb / estimate / guess • That estimates the distance / cost from each node to the destination node • To speed up the process of finding a solution • …by identify which paths to follow first	2AO1.1 (1)AO1.2 (1)	
		ii		8AO1.2 (3)AO2.1 (3)AO2.2 (2)	

Node	Distance travelled	Heuristic	Distance travelled + Heuristic	Previous node	MARKING GUIDANCE
A (✓)	0	90	90		**1 MARK**
B (✓)	∞ 21	80	101	A	**1 MARK**
C (✓)	∞ 42	65	107	A	**1 MARK**
D (✓)	∞ 42+12=54	50	104	C	**1 MARK**
E	∞ 21+40=61	50	111	B	**1 MARK**
F (✓)	∞ 42+12+23=77	30	107	D	**1 MARK**
G	∞ 42+12+23+33=110	0	110	F	**1 MARK**

Final path = A,C,D,F,G and Distance = 110 (1 Mark)

			Total	**10**	
39	a		**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of big O and sorting algorithms; the material is	9 AO1.1 (2)	**AO1: Knowledge and Understanding**

			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. The candidate is able to weigh up the use of the sorting algorithms which results in a supported and realistic judgment as to whether it is possible to use them in this context. *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 knoledge and understanding of big O and sorting algorithms; 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 makes a reasonable attempt to come to a conclusion showing some recognition of influencing factors that would determine whether it is possible to use the sorting algorithms in this context. *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 and sorting algorithms 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 nothing more than an unsupported assertion. *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 marks** No attempt to answer the question or response is not worthy of credit.	AO1.2 (2) AO2.1 (2) AO3.3 (3)	**Indicative content** O(1) • Constant space, does not change O(n) • Linear • Same as number of elements • As number of elements increases so does the time/space O(n^2) • polynominal • As number of elements increases, time/space increases by *n O(n log(n)) • Linearithmic **AO2: Application** • Space: Merge sort will require more memory usage as the number of elements increases. Insertion will not require any more space than original. Quick will increase but not as much as merge. • Best time: Insertion increases at the same rate as the number of elements. Quick and merge increase at greater rate • Worst time: insertion and quick

					increase significantly by n for each additional item. Merge sort increases less per element. • Log more appropriate for large number of elements **AO3: Evaluation** e.g. • Small array – space is not important. Few number of elements. Look for consistency. • Large array therefore memory important – could remove merge as inappropriate. Logarithmic more efficient.
	b		1 mark per bullet for description to max 6 • Compare each pair of adjacent elements • If they are not in the correct order then swap the elements • If they are in the correct order then do no swap elements • When you read the end of the array return to the start • Repeat n elements time • Set a flag to be false whenever a swap is made • …repeat the loop if the flag is not false	6 AO1.1 (2) AO1.2 (4)	
			Total	**15**	
4 0	a	i	1 mark per bullet to max 3 • Search the tree to find the location of Node E / by example of search • Replace the content of node E with blank/null/equivalent • Make node A point to the node H • Add node E to the empty node list	3 AO1.2 (3)	

<table>
<tr><td></td><td></td><td>ii</td><td>1 mark per bullet to max 3

• Search the tree to find the location of node G / by example of search
• Create a new node with value K
• Add a pointer from node G to the new node
• Make node K point to null/equivalent</td><td>3
AO1.2
(3)</td><td></td></tr>
<tr><td></td><td>b</td><td></td><td>1 mark per similarity to max 2

• Both consists of nodes
• Both are connected by edges/links
• Both are non-linear data structures
• Both are dynamic data structures

1 mark per difference to max 2

• Tree is 1-directional whereas a graph is 2-directional
• Tree has a root node whereas a graph does not have a (clear) root node
• Tree will not have cycles whereas graphs can contain cycles
• Tree will not be weighted whereas edges in a graph can be weighted</td><td>4
AO1.1
(4)</td><td></td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>10</td><td></td></tr>
<tr><td>4
1</td><td>a</td><td></td><td>1 mark per bullet

• Calculation of result to 3
• Call with <code>thisFunction(theArray, num1=4, num2=7, num3=35)</code>
• Result = 5
• call with <code>thisFunction(theArray,num1=6,num2=7,num3=35)</code>
• (Result = 6) return of value 6

<table><tr><th>Function call</th><th>num1</th><th>num2</th><th>num3</th><th>result</th></tr><tr><td><code>thisFunction (theArray,0,7,35)</code></td><td>0</td><td>7</td><td>35</td><td>3</td></tr><tr><td><code>thisFunction (theArray,4,7,35)</code></td><td>4</td><td>7</td><td>35</td><td>5</td></tr><tr><td><code>thisFunction (thisArray,6,7,35)</code></td><td>6</td><td>7</td><td>35</td><td>6</td></tr></table></td><td>5
AO2.1
(3)
AO2.2
(2)</td><td></td></tr>
<tr><td></td><td>b</td><td></td><td>Binary search</td><td>1
AO2.1
(1)</td><td></td></tr>
</table>

	c		1 mark per bullet to max 4, e.g. • Recursion uses more memory… • …iteration uses less memory • Recursion declares new variables /variables are put onto the stack each time… • …iteration reuses the same variables • Recursive can run out of memory/stack space… • …while iteration cannot run out of memory • Recursion can express a problem more elegantly / in fewer lines of code… • …while iteration can take more lines of code / be harder to understand • Recursion will be self-referential / will call itself… • … whereas iteration does not	4 AO1.1 (2) AO1.2 (2)	

	d		1 mark per bullet to max 6
			
			• Retains function call
			• Uses a loop
			• …that will loop until all elements inspected or value found
			• Updates num1 appropriately
			• Updates num2 appropriately
			• Returns -1 in the correct place if the value has not been found
			• Returns the result in the correct place if the value has been found
			
			e.g.
			`function thisFunction(theArray, num1, num2, num3)`
			
			`while (true)`
			`  result = num1 + ((num2 - num1) DIV 2)`
			`  if num2 < num1 then`
			`  return -1`
			`  else`
			`  if theArray[result] < num3 then`
			`  num1 = result + 1`
			`  elseif theArray[result] > num3 then`
			`  num2 = result - 1`
			`  else`
			`  return result`
			`  endif`
			`  endif`
			`endwhile`
			`endfunction`
				6
				AO2.2
				(3)
				AO3.1
				(3)	

			Total	**16**	
4 2	a		One node (node A) has more than 2 connections Nodes aren't ordered (e.g. F is C's left child)	1 AO2.1 (1)	

	b		1 mark for identification • Null pointers	1 AO2.1 (1)	
	c		1 mark per bullet • Take **A** as starting node • Visit **B**, **C** and **E** • Visit **D, F, G** and **H** • Visit **I** and **J**	4 AO1.2 (2) AO2.2 (2)	Allow the reverse ordering from right to left e.g. A; E, C, B; H, G, F, D; J, I
			Total	**6**	
4 3	a		• Insertion sort	1 A02.1 (1)	Allow other sorting algorithms not listed in the specification (e.g. Merge Sort, Quick Sort etc)
	b		**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of bubble sorts; 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. The candidate is able to weigh up the use of bubble sorts within the context which results in a supported and realistic judgment as to whether it is suitable to use within the context.*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 bubble sorts; 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 makes a reasonable attempt to come to a conclusion showing some recognition of influencing factors that would determine whether it is possible to use bubble sorts in this context. *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 bubble sorts 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	9 AO1.1 (2) AO1.2 (2) A02.1 (2) A03.3 (3)	**Knowledge and Understanding** • All adjacent items are compared against each other. • The biggest number in the adjacent pair is swapped over with the smallest number. A temp variable is used to hold the data while it's being moved. • When a swap is made a flag is set. • This is repeated for all adjacent values, known as one pass. • At the end of one pass, the largest item should appear at the end of the list. • If at the end of the list the flag has been set the flag is unset and the algorithm starts from the beginning of the list again. • When the algorithm gets to the end of

			provides nothing more than an unsupported assertion.*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 marks** No attempt to answer the question or response is not worthy of credit.		the list and the flag is unset the list is sorted. **Application** • As there are 250,000 items a bubble sort would perform very slowly as a lot of passes will need to be made in order to sort the items. • Bubble sorts are better suited to data sets where the items are almost/partly sorted. However the smaller numbers are currently towards the end and the larger numbers are towards the start. • This will therefore increase the amount of comparisons / passes/swaps required which will therefore slow the performance of the sort down. **Evaluation** • The algorithm is easy to implement as the number of lines of code is less than other standard sorting algorithms. • Although a bubble sort will be able to sort the items into order, it will take longer than other sorting algorithms due to the number of items and the current order or

					items in the unsorted list.
	c		• 249,999	1 A02.2 (1)	
			Total	**11**	
4 4	a	i	1 mark per bullet up to a maximum of 4 marks: • A queue is First In First Out (FIFO) • Therefore bookings will be executed in the order they have received • A stack is Last In First Out (LIFO) • Therefore the bookings would be executed from the most recent booking	4 A02.2 (4)	
		ii	`custNumber`	1 A03.3 (1)	
		iii	1 mark per bullet up to a maximum of 2 marks: • Correct logic for incrementing the tail pointer by 1 (e.g. `tail = tail +1`) • Correct logic for adding `custNumber` to the tail pointer (e.g. `queue[tail]=custNumber`)	2 A03.2 (2)	

		iv	• If the tail is greater than 10 / `maxElements`	1
				A02.2
				(1)
					Accept:
					`not((tail + 1) > maxElements)`
					Or
					`(tail + 1) <= maxElements`
					Or equivalent

	b	i	1 mark per bullet up to a maximum of 3 marks: • 5 (Jimmy) • 8 (Siad) • 9 (Tommy)	3 A02.1 (3)	
		ii	1 mark per bullet up to a maximum of 2 marks, e.g: • Efficient • ...as does not need to search every single element/uses divide and conquer	2 A01.2 (2)	
		iii	• Linear Search / Serial Search	1	

				A02.1 (1)	
		iv	• The items are in alphabetical order / the items are sorted	1 A02.1 (1)	
			Total	**15**	