

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
1	a	<p>1 mark for definition</p> <ul style="list-style-type: none"> Removal of unnecessary detail // Simplification to allow development of a program more easily <p>1 mark to max 2 for application e.g.</p> <ul style="list-style-type: none"> The actual movements are represented by vertices/lines State of the move is represented by a letter/symbol rather than the actual move position Tree does not show details about what the moves are 	3 AO1.1 (1) AO2.1 (1) AO2.2 (1)	Allow other suitable examples that are relevant to the scenario in the question.
	b	<p>One node (node A) has more than 2 connections</p> <p>Nodes aren't ordered (e.g. F is C's left child)</p>	1 AO2.1 (1)	
	c	<p>1 mark for identification</p> <ul style="list-style-type: none"> Null pointers 	1 AO2.1 (1)	
	d	<p>1 mark per bullet</p> <ul style="list-style-type: none"> 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
	e	<p>i</p> <p>1 mark per bullet to max 3</p> <ul style="list-style-type: none"> 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)	
		<p>ii</p> <p>1 mark per bullet to max 3</p> <ul style="list-style-type: none"> 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 	3 AO1.2 (3)	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	f	<p>1 mark per similarity to max 2</p> <ul style="list-style-type: none"> • Both consists of nodes • Both are connected by edges/links • Both are non-linear data structures • Both are dynamic data structures <p>1 mark per difference to max 2</p> <ul style="list-style-type: none"> • 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 	4 AO1.1 (4)	
		Total	19	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
2	a	<p>1 mark per bullet to max 4 e.g.</p> <ul style="list-style-type: none">• Decomposition splits the problem into smaller sub problems• Repeated decomposition gives solvable parts• The division can lead to the development of subroutines/modules• The division can lead to a logical division between programmers/teams• ...e.g. one team works on one section and another concurrently on another	4 AO1.1 (2) AO1.2 (2)	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	b	<p>Mark Band 3 – High level (7-9 marks) The candidate demonstrates a thorough knowledge and understanding of concurrent processing; 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. <i>There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.</i></p> <p>Mark Band 2 – Mid level (4-6 marks) The candidate demonstrates reasonable knowledge and understanding of concurrent processing; 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. <i>There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.</i></p> <p>Mark Band 1 – Low Level (1-3 marks) The candidate demonstrates a basic knowledge of concurrent processing 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. Judgements if made are weak and unsubstantiated. <i>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.</i></p>	9 AO1.1 (2) AO1.2 (2) AO2.1 (2) AO3.3 (3)	<p>AO1: Knowledge and Understanding Indicative content</p> <ul style="list-style-type: none"> • Processes are happening at the same time/at overlapping times • One process may need to start before a second has finished • Individual processes are threads, each thread has a life line • One request will be sent to the server, this will have a thread <p>AO2: Application</p> <ul style="list-style-type: none"> • Multiple requests to the server can be made at the same time • Programming on server will need to allow multiple threads to manipulate a list of requests • Programming will need to restrict access to the database of seats/sales etc. • Will allow those reading and writing to manipulate at the same time • Record locking will need implementing – more complex programming • May be selling alongside other systems, therefore needs to communicate with external systems that will also use record locking to avoid two different external companies accessing and selling the same tickets. <p>AO3: Evaluation</p> <ul style="list-style-type: none"> • Will allow for multiple access to the website at the same time by different customers – as it would happen in real life • Will allow for multiple ticket sales for the same event without selling the same seat twice

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
		0 marks No attempt to answer the question or response is not worthy of credit.		
		Total	13	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance																									
3	a	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • 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 border="1" style="margin-top: 10px;"> <thead> <tr> <th>Function call</th> <th>num1</th> <th>num2</th> <th>num3</th> <th>result</th> </tr> </thead> <tbody> <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> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Function call	num1	num2	num3	result	<code>thisFunction(theArray, 0, 7, 35)</code>	0	7	35	3	<code>thisFunction(theArray, 4, 7, 35)</code>	4	7	35	5	<code>thisFunction(thisArray, 6, 7, 35)</code>	6	7	35	6						<p>5</p> <p>AO2.1 (3)</p> <p>AO2.2 (2)</p>	
Function call	num1	num2	num3	result																									
<code>thisFunction(theArray, 0, 7, 35)</code>	0	7	35	3																									
<code>thisFunction(theArray, 4, 7, 35)</code>	4	7	35	5																									
<code>thisFunction(thisArray, 6, 7, 35)</code>	6	7	35	6																									
	b	Binary search	<p>1</p> <p>AO2.1 (1)</p>																										
	c	<p>1 mark per bullet to max 4, e.g.</p> <ul style="list-style-type: none"> • 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 	<p>4</p> <p>AO1.1 (2)</p> <p>AO1.2 (2)</p>																										

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	d	<p>1 mark per bullet to max 6</p> <ul style="list-style-type: none"> • 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 <p>e.g.</p> <pre>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</pre>	6 AO2.2 (3) AO3.1 (3)	
		Total	16	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
4	a	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • By reference will change the actual contents of the array in the main program// when control returns to the main program the array will be sorted • By value would create a copy and not change the original // when control returns to the main program the array will not be sorted • By value the array is local to the function. • By reference will use less memory 	<p>2</p> <p>AO1.2 (1)</p> <p>AO2.2 (1)</p>	
	b	<p>1 mark per bullet to max 3</p> <ul style="list-style-type: none"> • Descending order • Line 07 (dataArray[tempos] < temp) has the comparison... • ...that checks if current position is less than item to insert and... • ...breaks out of loop when current position is less than or equal to item to insert 	<p>3</p> <p>AO1.2 (1)</p> <p>AO2.2 (2)</p>	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	c	<p>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 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. <i>There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.</i></p> <p>Mark Band 2 – Mid level (4-6 marks) The candidate demonstrates reasonable knowledge 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. <i>There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence</i></p> <p>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 candidate makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides nothing more than an unsupported assertion. <i>The information is basic and communicated</i></p>	9 AO1.1 (2) AO1.2 (2) AO2.1 (2) AO3.3 (3)	<p>AO1: Knowledge and Understanding Indicative content O(1)</p> <ul style="list-style-type: none"> Constant space, does not change <p>O(n)</p> <ul style="list-style-type: none"> Linear Same as number of elements As number of elements increases so does the time/space <p>O(n²)</p> <ul style="list-style-type: none"> polynomial As number of elements increases, time/space increases by *n <p>O(n log(n))</p> <ul style="list-style-type: none"> Linearithmic <p>AO2: Application</p> <ul style="list-style-type: none"> 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 <p>AO3: Evaluation e.g.</p> <ul style="list-style-type: none"> 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.

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
		<p><i>in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.</i></p> <p>0 marks</p> <p>No attempt to answer the question or response is not worthy of credit.</p>		
d		<p>1 mark per bullet for description to max 6</p> <ul style="list-style-type: none"> • 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	20	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance														
5	a	<p>1 mark per pointer</p> <ul style="list-style-type: none"> queueHead: Point to the first element in the queue // next element to remove queueTail: Point to the last element in the queue 	2 AO1.2 (2)															
	b	<p>1 mark per bullet up to max 5</p> <ul style="list-style-type: none"> first 3 jobs removed 128 and 129 added in positions 4 and 5 respectively no additional jobs queueHead being 3 (FT errors) queueTail being 5 (FT errors) <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="flex: 1;"> <p>queueHead</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">3</div> </div> <div style="flex: 1; text-align: center;"> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">6</td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">5</td><td style="padding: 2px;">job-129</td></tr> <tr><td style="padding: 2px;">4</td><td style="padding: 2px;">job-128</td></tr> <tr><td style="padding: 2px;">3</td><td style="padding: 2px;">job-127</td></tr> <tr><td style="padding: 2px;">2</td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">0</td><td style="padding: 2px;"></td></tr> </table> </div> <div style="flex: 1;"> <p>queueTail</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">5</div> </div> </div>	6		5	job-129	4	job-128	3	job-127	2		1		0		5 AO2.1 (2) AO2.2 (3)	<p>The underlying implementation of the queue has not been specified, so allow alternative valid answers.</p> <p>e.g.</p> <p>queueHead = 0 queueTail = 2 Location 2: 129 Location 1: 128 Location 0: 127</p>
6																		
5	job-129																	
4	job-128																	
3	job-127																	
2																		
1																		
0																		
	c	<p>1 mark per bullet to max 5</p> <ul style="list-style-type: none"> Function declaration Checking if queue is empty ...returning null (Otherwise) incrementing queueHead ...returning buffer[queueHead-1] <p>e.g.</p> <pre>function dequeue() if queueHead > queueTail then return null else queueHead = queueHead + 1 return buffer[queueHead-1] endif endfunction</pre>	5 AO2.2 (2) AO3.3 (3)	<p>Note: Accept alternative valid underlying implementation answers e.g. Shifting all elements in queue forward.</p>														

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	ii	<p>1 mark per bullet to max 6</p> <ul style="list-style-type: none"> • Function declaration taking parameter • Checking if queue is full • ...returning -1 • (Otherwise) incrementing queueTail • Adding newJob to buffer(queueTail) • Returning 1 <p>e.g.</p> <pre>function enqueue(newJob) if queueTail == 99 then return -1 else queueTail = queueTail + 1 buffer[queueTail] = newJob return 1 endif endfunction</pre>	6 AO2.2 (3) AO3.3 (3)	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	iii	<p>1 mark per bullet to max 8</p> <ul style="list-style-type: none"> • Inputting user choice • If enqueue chosen input job name • ...call enqueue with input value as parameter • ...check if return value is -1 and output full • ...otherwise output message that item is added • If dequeue chosen • ...call dequeue and save returned value • ...output returned value (jobname) if not null • ...or output queue is empty <p>e.g.</p> <pre>main() choice = input("Add or remove?") if choice == "ADD" then jobname = input("Enter job name") returnValue = enqueue(jobname) if returnValue == -1 then print("Queue full") else print("Job added") endif else returnValue = dequeue() if returnValue == null then print("Queue empty") else output returnValue endif endif endmain</pre>	8 AO2.2 (2) AO3.3 (6)	Allow equivalent checks / logic
	d	<p>1 mark per bullet to 3</p> <ul style="list-style-type: none"> • Check if either head or tail are incremented to above 99 • ... set to be 0 instead • When checking if array is full check if (queueTail == queueHead - 1) OR (queueTail==99 AND queueHead==0) 	3 AO2.1 (1) AO2.2 (2)	Credit equivalent modulo arithmetic solution

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	e	<p>1 mark per bullet to max 3, e.g.</p> <ul style="list-style-type: none"> • Use a different structure e.g. a linked list • ...items can be added at different points in the linked list depending on priority • ...by changing the pointers to items needing priority • Have different queues for different priorities • ...add the job to the queue relevant to its priority • ...print all the jobs in the highest priority queue first 	3 AO2.1 (2) AO2.1 (1)	Allow other suitable descriptions that show how the program could be amended.
		Total	32	

Mark Scheme

Question			Answer/Indicative content	Marks	Guidance																																																																																																														
6	a	i	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • Points to where the next/first free node is • To add data into the linked listed. 	2 AO1.2 (1) AO2.2 (1)																																																																																																															
		ii	Points to the first element in the linked list	1 AO1.2 (1)																																																																																																															
		iii	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • No change made to nodes/pointers unaffected by this removal • Index 0 points to 2 instead of 3 • Node 9 points to 3 instead of -1 // Node freeListPointer points to 3 instead of 4 • Node 3 points to 4 // -1 (must match MP3) <p>Solution:</p> <table style="margin-left: 200px;"> <tr> <td>headPointer</td> <td>0</td> <td>index</td> <td>data</td> <td>pointer</td> </tr> <tr> <td>freeListPointer</td> <td>4</td> <td>0</td> <td>2.6</td> <td>2</td> </tr> <tr> <td></td> <td></td> <td>1</td> <td>3.5</td> <td>-1</td> </tr> <tr> <td></td> <td></td> <td>2</td> <td>1.8</td> <td>1</td> </tr> <tr> <td></td> <td></td> <td>3</td> <td>6.9</td> <td>-1</td> </tr> <tr> <td></td> <td></td> <td>4</td> <td></td> <td>5</td> </tr> <tr> <td></td> <td></td> <td>5</td> <td></td> <td>6</td> </tr> <tr> <td></td> <td></td> <td>6</td> <td></td> <td>7</td> </tr> <tr> <td></td> <td></td> <td>7</td> <td></td> <td>8</td> </tr> <tr> <td></td> <td></td> <td>8</td> <td></td> <td>9</td> </tr> <tr> <td></td> <td></td> <td>9</td> <td></td> <td>3</td> </tr> </table> <p>Alternative Solution:</p> <table style="margin-left: 200px;"> <tr> <td>headPointer</td> <td>0</td> <td>index</td> <td>data</td> <td>pointer</td> </tr> <tr> <td>freeListPointer</td> <td>3</td> <td>0</td> <td>2.6</td> <td>2</td> </tr> <tr> <td></td> <td></td> <td>1</td> <td>3.5</td> <td>-1</td> </tr> <tr> <td></td> <td></td> <td>2</td> <td>1.8</td> <td>1</td> </tr> <tr> <td></td> <td></td> <td>3</td> <td>6.9</td> <td>4</td> </tr> <tr> <td></td> <td></td> <td>4</td> <td></td> <td>5</td> </tr> <tr> <td></td> <td></td> <td>5</td> <td></td> <td>6</td> </tr> <tr> <td></td> <td></td> <td>6</td> <td></td> <td>7</td> </tr> <tr> <td></td> <td></td> <td>7</td> <td></td> <td>8</td> </tr> <tr> <td></td> <td></td> <td>8</td> <td></td> <td>9</td> </tr> <tr> <td></td> <td></td> <td>9</td> <td></td> <td>-1</td> </tr> </table>	headPointer	0	index	data	pointer	freeListPointer	4	0	2.6	2			1	3.5	-1			2	1.8	1			3	6.9	-1			4		5			5		6			6		7			7		8			8		9			9		3	headPointer	0	index	data	pointer	freeListPointer	3	0	2.6	2			1	3.5	-1			2	1.8	1			3	6.9	4			4		5			5		6			6		7			7		8			8		9			9		-1	4 AO1.2 (1) AO2.2 (1)	<p>6.9 3 may or may not be written by candidates, both are acceptable.</p> <p>Candidates may add the node freed up (node 3) to the start or the end of the free storage. Award marks for both approaches.</p>
headPointer	0	index	data	pointer																																																																																																															
freeListPointer	4	0	2.6	2																																																																																																															
		1	3.5	-1																																																																																																															
		2	1.8	1																																																																																																															
		3	6.9	-1																																																																																																															
		4		5																																																																																																															
		5		6																																																																																																															
		6		7																																																																																																															
		7		8																																																																																																															
		8		9																																																																																																															
		9		3																																																																																																															
headPointer	0	index	data	pointer																																																																																																															
freeListPointer	3	0	2.6	2																																																																																																															
		1	3.5	-1																																																																																																															
		2	1.8	1																																																																																																															
		3	6.9	4																																																																																																															
		4		5																																																																																																															
		5		6																																																																																																															
		6		7																																																																																																															
		7		8																																																																																																															
		8		9																																																																																																															
		9		-1																																																																																																															

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	b	<p>i</p> <p>1 mark per bullet</p> <ul style="list-style-type: none"> • Class declaration and all code is nested within the class • Two private identifiers data and pointer (with suitable data types if given) • Public constructor heading as a procedure (public may be implied but cannot be private) taking both parameters as given in table • Assigns parameters to the attributes <p>e.g.</p> <pre>public class node private data as real private pointer as integer public procedure new(newData, newPointer) data = newData pointer = newPointer endprocedure endclass</pre>	<p>4</p> <p>AO2.2 (1)</p> <p>AO3.3 (3)</p>	<p>Accept</p> <p>public node(newData, newPointer) (may also have data types for parameters e.g. int newData)</p> <p>Accept:</p> <p>this.data = newData this.pointer = newPointer</p> <p>or similar</p>
		<p>ii</p> <p>1 mark per bullet to max 2</p> <ul style="list-style-type: none"> • A get method allows the attribute to be accessed / returned • A set method allows the attribute to be changed (with parameters) 	<p>2</p> <p>AO2.2 (2)</p>	
	c	<p>1 mark per bullet to max 6</p> <ul style="list-style-type: none"> • Initialise message string • Start with the headPointer • Check if the headPointer is null • ...return that the list is empty • Check the pointer of the node at headPointer • If it is not null/-1/the last element • loop through all the linkedList elements • ...concatenate the pointer to the message • ...replacing the pointer with the current node's pointer each time • ...if the data is found concatenate the pointer and "found" to the message and return it • ...if the loop ends and the data item is not found, concatenate "not found" to the message and return it 	<p>6</p> <p>AO1.2 (2)</p> <p>AO2.1 (2)</p> <p>AO2.2 (2)</p>	

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	d	<p>1 mark for identifying error and correction (identification may be implicit)</p> <ul style="list-style-type: none"> • Line 02 tempPointer should become headPointer, not -1 tempPointer = headPointer • Line 05 message should say it's empty not full print("List is empty") • Line 07 pointer should be tempPointer while linkedList[tempPointer].getPointer() != -1 • Line 08 Incorrect call to node pointer dataToPrint = dataToPrint + " " + linkedList[tempPointer].getData() • Line 09 assignment is wrong way tempPointer = linkedList[tempPointer].getPointer() • Line 11 missing final parenthesis print(dataToPrint+ " " + linkedList[tempPointer].getData()) 	3 AO2.1 (2) AO2.2 (2)	<p>Do not award marks for stating the line number without a valid correction.</p>
	ii	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • Stepping Through The Code... • ...to run one line at a time to see where the error is • Syntax Error Highlighting.... •to distinguish syntax errors in the program code • Setting breakpoints... •to debug individual sections of code at a time • Variable watch window... • ...To check that the variables are being updated correctly 	6	<p>The features must relate to debugging code.</p> <p>Allow other suitable features appropriate to debugging code.</p> <p>1 Mark for identification and 1 mark for suitable expansion.</p>

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
	e	<p>Mark Band 3 – High level (9-12 marks) The candidate demonstrates a thorough knowledge and understanding of the object orientated techniques; 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 all of the object orientated techniques which results in a supported and realistic judgment as to whether it is possible to use them in this context.</p> <p><i>There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.</i></p> <p>Mark Band 2 – Mid level (5-8 marks) The candidate demonstrates reasonable knowledge and understanding of the object orientated techniques; the material is generally accurate but at times underdeveloped.</p> <p>The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed.</p> <p>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 each object orientated technique in this context.</p> <p><i>There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence</i></p> <p>Mark Band 1 – Low Level (1-4 marks) The candidate demonstrates a basic knowledge of the object orientated techniques 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.</p> <p>The candidate provides nothing more than an unsupported assertion.</p>	12 AO1.1 (3) AO1.2 (3) AO2.1 (3) AO3.3 (3)	<p>AO1: Knowledge and Understanding Indicative content</p> <ul style="list-style-type: none"> Classes, this is a template. It will define what attributes and methods an object should have. Objects, when you create an instance of a class. Each object that is instantiated from the same class will share the same attributes and methods. Inheritance, when a sub class takes on the attributes and methods from a superclass/parent class. It can also have its own extra attributes/methods. Overriding, when a method name is the same in a parent and sub class, then the method in the parent/super class will be overridden Encapsulation, this protects attributes of an object by making them private so that they can't be accessed or altered accidentally by other objects. <p>AO2: Application</p> <ul style="list-style-type: none"> A class can be used to declare the attributes and methods for the linked list. These will initialise the nodes and join them. Objects can then be used to instantiate the class each time a new linked list is needed. Each can be given a different identifier by the other programs. Further subclasses may be used by other programs. These can therefore take on the attributes and methods from the base class. These can also be changed or overridden depending on the purpose of the other programs. Encapsulation can be used by using set and get methods to ensure that the nodes in the linked list are changed in a way that is intended. <p>AO3: Evaluation</p> <ul style="list-style-type: none"> Use of OPP techniques will allow for code reusability. His linked list can be saved as library and then reused many times leading to less code. OOP also allows programs to be easier to modify and maintain.

Mark Scheme

Question		Answer/Indicative content	Marks	Guidance
		<p><i>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.</i></p> <p>0 marks No attempt to answer the question or response is not worthy of credit.</p>		
		Total	40	