

Mark scheme

Question			Answer/Indicative content	Mark	Guidance
1			<p>1 mark for each input to max 2 e.g.</p> <ul style="list-style-type: none"> • Entering a name • Selecting a vehicle • Pressing arrow key to move forward • Pressing arrow key to move backward • Pressing arrow key to move left • Pressing arrow key to move right <p>1 mark for each output to max 2 e.g.</p> <ul style="list-style-type: none"> • Images of vehicles to choose from • Background of area • Image of other vehicles • Image of controls and description of what they do 	4	<p>Allow any feasible input/output for scenario</p> <p><u>Examiner's Comments</u></p> <p>While many candidates scored full marks for this question for identifying suitable inputs and outputs within the context of the problem a number of candidates scored no marks for erroneously giving input or output devices. Sometimes responses were unqualified such as 'left arrow key' rather than 'left arrow to turn left' where candidates did not identify an input within the context of the scenario as an input to the game.</p>
			Total	4	
2			<p>1 mark for each input to max 2</p> <ul style="list-style-type: none"> • Username • Password <p>1 mark for output e.g.</p> <ul style="list-style-type: none"> • Message to request input • Message to state login successful • Message to say login unsuccessful 	3	<p><u>Examiner's Comments</u></p> <p>This question was generally well answered by most candidates, but answers had to be given in the context of the scenario. Some less successful responses mistakenly identified input/output devices rather than specific examples of inputs and outputs.</p>
			Total	3	
3	a	i	<p>1 mark for:</p> <ul style="list-style-type: none"> • <code>isInteger</code> • <code>number</code> • <code>result</code> • <code>count</code> 	1	<p>Penalise excessive spaces in identifiers such as <i>ascii Value</i> instead of <i>asciiValue</i></p> <p><u>Examiner's Comments</u></p>

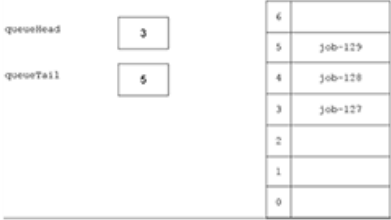
			<ul style="list-style-type: none"> • <code>asciiValue</code> 		<p>This question was generally well done (although slightly less well done than parts (a) (ii) and (a) (iii)), but many candidates did very well. The most common erroneous responses were giving the names of predefined functions/properties or giving relational operators.</p>
		ii	(0)5	1	<p><u>Examiner's Comments</u></p> <p>This question required an exact answer only and was answered correctly by the majority of candidates.</p>
		iii	(0)3	1	<p><u>Examiner's Comments</u></p> <p>This question also required an exact answer only and was answered correctly by the majority of candidates.</p>
		b	<p>1 mark each to max 2:</p> <ul style="list-style-type: none"> • 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	<p><u>Examiner's Comments</u></p> <p>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'.</p> <p>Pre-written or pre-tested, and saving development time due to already being written, were the most popular answers.</p>
			Total	5	
4		i	<p>1 mark per bullet e.g.</p> <ul style="list-style-type: none"> • stage (e.g. stage 1, stage 2, stage 3) • city name (e.g. London) • speed (e.g. slow, normal, fast) 	2	<p><u>Examiner's Comments</u></p> <p>Most candidates successfully identified the relevant potential inputs of city/stage/speed from the given scenario. Some</p>

					candidates suggested alternative inputs such as 'magnitude of earthquake' which were also given marks as they were valid inputs for a simulation relating to earthquakes.
		ii	<p>1 mark per bullet to max 2, e.g.</p> <ul style="list-style-type: none"> Does the build-up stage need to be shown? Does the earthquake taking place needs to be shown? Does the aftershock stage needs to be shown? 	2	<p>Allow other suitable examples</p> <p><u>Examiner's Comments</u></p> <p>Few candidates were given full marks for this question. There was some repetition of checking user input values, which was given in the stem of the question. There were also many statements of possible calculations rather than clearly expressed conditions or questions. More successful responses included decisions such as 'have buildings of a certain type survived the earthquake?'.</p>
			Total	4	
5		i	<p>1 mark per bullet</p> <ul style="list-style-type: none"> 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	<p><u>Examiner's Comments</u></p> <p>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.</p> <p>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.</p>

		ii	A loop that repeats a fixed / set number of times	1	<p><u>Examiner's Comments</u></p> <p>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.</p>
		iii	<ul style="list-style-type: none"> 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	<p><u>Examiner's Comments</u></p> <p>Many candidates found it difficult to explain the purpose of the <code>holdValue</code> variable in context. Where candidates achieved some of the marks, they most frequently identified <code>holdValue</code> 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 <code>dataArray[x]</code> and <code>dataArray[x+1]</code> to be swapped.</p> <p>Exemplar 1</p> <p><i>(It acts as a temporary variable to hold the value of numbers [x] while before number is overwritten so the original value is in both and is instead copied to numbers[x+1])</i></p> <p>This exemplar very clearly states exactly how and why the variable <code>holdValue</code> is required.</p>
		iv	<ul style="list-style-type: none"> 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	<p><u>Examiner's Comments</u></p> <p>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</p>

					who were able to expand on this to explain that this was required to repeat the process for the required number of passes, or until no swaps had occurred during a pass.																				
			Total	9																					
6			1 mark for the purpose and 1 mark for matching appropriate name (4 marks total), e.g: <ul style="list-style-type: none">• Pointer to the first element in the queue• firstElement / any other meaningful name• Pointer to the last element in the queue / Pointer to the first free element in the queue• lastElement / any other meaningful name	4AO1.2 (4)	Must cover purpose and name for 2 marks for each pointer.																				
			Total	4																					
7	a		1 mark per bullet <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><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>	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	5 AO2.1 (3) AO2.2 (2)	
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	1 AO2.1 (1)																					
	c		1 mark per bullet to max 4, e.g. <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	4 AO1.1 (2) AO1.2 (2)																					

			<ul style="list-style-type: none"> 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 		
	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>	<p>6 AO2.2 (3) AO3.1 (3)</p>	
			Total	16	
8			<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 AO1.2 (1) AO2.2 (2)</p>	

			Total	3	
9	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 	<p>2 AO1.2 (2)</p>	
	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) 	<p>5 AO2.1 (2) AO2.2 (3)</p>	<p>The underlying implementation of the queue has not been specified, so allow alternative valid answers. e.g. queueHead = 0 queueTail = 2 Location 2: 129 Location 1: 128 Location 0: 127</p>
	c	i	<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>	<p>5 AO2.2 (2) AO3.3 (3)</p>	<p>Note: Accept alternative valid underlying implementation answers e.g. Shifting all elements in queue forward.</p>
		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) 	<p>6 AO2.2 (3) AO3.3 (3)</p>	

			<ul style="list-style-type: none"> 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>		
		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>	<p>8 AO2.2 (2) AO3.3 (6)</p>	<p>Allow equivalent checks / logic</p>
		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) 	<p>3 AO2.1 (1) AO2.2 (2)</p>	<p>Credit equivalent modulo arithmetic solution</p>

	e		1 mark per bullet to max 3, e.g. <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	
10	a		<ul style="list-style-type: none"> • 10 	1 A03.2 (1)	
	b		<ul style="list-style-type: none"> • 30 	1 A03.2 (1)	
	c		<ul style="list-style-type: none"> • 10 	1 A03.2 (1)	
			Total	3	