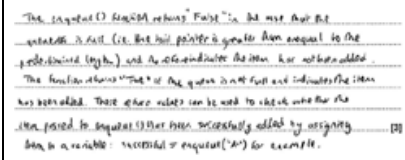


## Mark scheme

Question			Answer/Indicative content	Marks	Guidance
1		i	<code>item</code>	1	<p>Allow <code>queue</code> can be circular or linear, <code>stack</code> can only be linear</p> <p><b><u>Examiner's Comments</u></b></p> <p>The vast majority of candidates identified that <code>item</code> was the paramete</p>
		ii	<code>tailPointer</code>	1	<p>Allow <code>queue</code></p> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidates identified that <code>tailPointer</code> was an example of a global variable in the code.</p>
		iii	<p>1 mark for benefit and 1 mark for expansion: e.g.</p> <ul style="list-style-type: none"> <li>• Simpler to program ...</li> <li>• ... because values do not need to be passed/renamed/moved between different subroutines</li> <li>• Do not need to worry about returning values / do not need to decide between <code>byval/byref</code> ...</li> <li>• ... all parts of the program can access the (same) value (in the same way)</li> </ul> <p>1 mark for drawback and 1 mark for expansion:</p>	4	<p>2 marks max for benefit 2 marks max for drawback</p> <p><b><u>Examiner's Comments</u></b></p> <p>The majority of candidates struggled to describe the implications of using global variables, and this highlighted a lack of appreciation of parameter passing and variable scope. At AS Level candidates should become more adept at using parameters and develop an understanding of why this is beneficial – both in theoretical terms and</p>

			<ul style="list-style-type: none"> <li>• Uses more memory ...</li> <li>• ... because the memory space is declared when the program starts and remains in use throughout</li> <li>• Makes testing / debugging more difficult ...</li> <li>• .... as it's difficult to test an individual block of code</li> <li>• Reduces data accuracy / integrity ...</li> <li>• .... changing a global variable may have an impact on another module</li> </ul>		<p>within their personal programming practice. There were few responses that were able to give complete descriptions.</p>
		iv	<p>1 mark per bullet to max 3</p> <ul style="list-style-type: none"> <li>• False is returned if the queue is full ...</li> <li>• ... the main will not attempt to add another item</li> <li>• True is returned if the item is successfully added ...</li> <li>• ... so the main program can try to add another item</li> </ul>	3	<p><b><u>Examiner's Comments</u></b></p> <p>Some candidates thought that if <code>enqueue()</code> returned <code>True</code> it meant that there would be space to add additional values. This was not the case; it meant that the item had been enqueued successfully, but it could have been the last item that then filled the queue to capacity.</p> <p>Sometimes candidates did explain the reasons why <code>True</code> and <code>False</code> would be returned but were then unable to go on to say how they could be used. Good responses were seen such as 'the main program can give an error message if the operation didn't work if the queue was full'.</p> <p>Exemplar 1</p>

					
	v	1 mark for error <b>and</b> correction to max 3 <ul style="list-style-type: none"><li>• <b>Line 01</b> There is no need to pass in a parameter</li><li>• <b>Line 02</b> has != instead of ==</li><li>• <b>Line 04</b> has elseif and no condition / replace with <b>else</b></li><li>• <b>Line 06</b> should be headpointer = headpointer + 1 / Swap lines 06 and 07</li><li>• <b>Line 07</b> should be return value</li></ul>	3	<p>Do not award marks for the line number. The error must be related to the line number stated.</p> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidates identified at least one logical error, but few were able to identify three errors. The most common error identified was the error in line 02.</p>	
	vi	1 mark per bullet to max 3 <ul style="list-style-type: none"><li>• Use of loop to call dequeue()....</li><li>• ...to output the return value...</li><li>• ...until "EMPTY" is returned</li></ul> <p>e.g.</p> <pre>elementsLeft = true while elementsLeft == true value = dequeue() if not(value = "EMPTY") then print(value) else elementsLeft = false</pre>	3	<p><b><u>Examiner's Comments</u></b></p> <p>Some candidates mistakenly gave a definition for the dequeue() function instead of answering the question set. Candidates need to be mindful to read the question carefully.</p> <p>Some candidates used a count-controlled loop whereas a conditional loop was required. Those who made a good attempt at a response often did not assign the result of dequeue() to a variable,</p>	

			endif endwhile		so could not effectively use the return value from the function as well as print the value returned.
			<b>Total</b>	<b>15</b>	
2	a		<p>1 mark each to max 3. Max 2 for generic answers with no relation to scenario. e.g.</p> <ul style="list-style-type: none"> <li>• Has a set/fixed number of values</li> <li>• ...and the number of spaces in the road will not change</li> <li>• Stores data of one type</li> <li>• ... as the array is only made up of prize objects</li> <li>• Stores data linearly</li> <li>• ... match the linear nature of the road</li> <li>• Array contents are mutable</li> <li>• ... so prizes can be added/removed from the road</li> <li>• A single identifier is used to directly index</li> <li>• ... any position in the road</li> <li>• Can be iterated by index</li> <li>• ... to perform an operation on all road positions</li> </ul>	3	<p><b><u>Examiner's Comments</u></b></p> <p>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.</p>
	b	i	<p>1 mark each</p> <ul style="list-style-type: none"> <li>• Function/subroutine with identifier <code>getName</code> taking <b>no</b> parameters</li> <li>• Returning <code>name</code></li> </ul> <p>e.g.</p>	2	<p>BP1 Do not award procedure or method</p> <p>BP1 Allow self as an additional parameter if Python is used.</p> <p>BP1 If an access modifier is given for the method, it must be public and not private.</p> <p>BP2 Do not allow any modified name attribute to be returned.</p>

		<pre> public function getName() return name endfunction public getName() { return name } </pre>	<pre> def getName(self): return self.__name  function getName() { return this.name } </pre>		<p><b><u>Examiner's Comments</u></b></p> <p>While many candidates had little difficulty giving code for a <code>getter()</code> there were a number of common errors. Some candidates used a private access modifier when a <code>getter()</code> needs to be public. There was often erroneous use of 'procedure' whereas a <code>getter()</code> is a function that must return a value.</p> <p>Some candidates tried to set values within the <code>getter()</code> function when it should only have returned the class attribute value.</p>
	ii	<p>1 mark each</p> <ul style="list-style-type: none"> <li>• New instance of <code>prize</code> ...</li> <li>• ... with "Box", "money" and 25 as parameters</li> <li>• Assigned to <code>allPrizes</code> index 3</li> </ul> <p>e.g.</p> <pre> allPrizes[3] = new prize("Box", "money", 25) allPrizes[3] = prize.new("Box", "money", 25) allPrizes[3] = prize("Box", "money", 25) </pre>		3	<p>MP2 allow any order of parameters</p> <p>"Box" and "Money" must be strings and 25 must be an integer</p> <p>Allow <code>prize.new()</code> as new is given as the constructor method in the class diagram</p> <p><b><u>Examiner's Comments</u></b></p> <p>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".</p>
	iii	<p>1 mark for each bullet to maximum 3 e.g.</p> <ul style="list-style-type: none"> <li>• <b>Decision</b> - check whether the space already has a prize allocated ...</li> </ul>		3	<p>Give:</p> <ul style="list-style-type: none"> <li>• 1 mark for stating a decision</li> <li>• 1 mark for the action required if true</li> </ul>

		<ul style="list-style-type: none"> <li>• <b>Action if true</b> - another space/number will need to be generated</li> <li>• <b>Action if false</b> - the prize will be stored here</li> </ul> <ul style="list-style-type: none"> <li>• <b>Decision</b> - check if all 10 prizes have been allocated ...</li> <li>• <b>Action if true</b> - the algorithm needs to stop generating numbers</li> <li>• <b>Action if false</b> - a new number/space needs to be generated and checked</li> </ul>		<ul style="list-style-type: none"> <li>• 1 mark for the action required if false</li> </ul> <p><b><u>Examiner's Comments</u></b></p> <p>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 the random spaces but did not highlight the program conditions/decisions as required.</p>
	c i	<p>1 mark each</p> <ul style="list-style-type: none"> <li>• Constructor header (any suitable name e.g. new, constructor, create, init)</li> <li>• ...taking <b>one</b> parameter only</li> <li>• Initialising name to the parameter</li> <li>• Initialising money to 5</li> <li>• Initialising experience to 0 and roadPosition to 0</li> </ul> <p>e.g.</p>	5	<p>Allow minor changes to identifiers as long as purpose is clear.</p> <p>Allow</p> <pre>procedure new(pName) this.name = pName ...</pre> <p>(or similar e.g. <i>self.name</i>)</p> <p>Allow two parameters if one is <i>self</i> and the response is clearly in Python.</p> <p>The parameter name should be different to the attribute name.</p>

		<p><b>Pseudocode example:</b></p> <pre> public procedure new(pName) name = pName experience = 0 roadPosition = 0 money = 5 endprocedure </pre> <p><b>Python Example:</b></p> <pre> def __init__(self, pName): self.__name = pName self.__experience = 0 self.__roadPosition = 0 self.__money = 5 </pre> <p><b>C# Example:</b></p> <pre> public Character(string pName) { string name = pName; int experience = 0; int roadPosition = 0; int money = 5; } </pre>		<p><b><u>Examiner's Comments</u></b></p> <p>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.</p>
	ii	<p>1 mark each</p> <ul style="list-style-type: none"> <li>• Procedure/method header ...</li> <li>• ... taking <b>two</b> parameters, type (or similar) followed by value (or similar) ...</li> <li>• ... compare type parameter with "money"</li> <li>• ... compare type parameter with "experience"</li> <li>• ... both attributes updated correctly and nothing else modified</li> </ul> <p>e.g.</p> <pre> public procedure updateValues(pType, pValue) if pType == "money" then </pre>	5	<p>Do not allow Function for BP1</p> <p>BP2 parameters must be given in the correct order to match the calls to <code>updateValues()</code> in the question.</p> <p>"money" and "experience" must be string values</p> <p><b><u>Examiner's Comments</u></b></p> <p>The <code>updateValues</code> 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</p>

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



	e	<p>1 mark each</p> <ul style="list-style-type: none"> <li>• (Line 02) <code>for x = 0 to 49</code></li> <li>• (Line 03) <code>print("Space", x)</code></li> <li>• (Line 06) <code>else / elseif road[x] &lt;&gt; null</code></li> <li>• (Line 07) <code>print(road[x].getName())</code></li> </ul>	4	<p>Line 07 allow <code>print(road[x].name)</code></p> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidates scored three or four marks but in 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.</p>
	f	<p><b>Mark Band 3 – High level (7-9 marks)</b> 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. <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><b>Mark Band 2 – Mid level (4-6 marks)</b> 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</p>	9	<p><b>AO1: Knowledge and Understanding</b> <b>Indicative content</b></p> <ul style="list-style-type: none"> <li>• Global variables are created when the program starts, all subroutines can access/update the contents</li> <li>• Local variables are created in the subroutine they are created in, they are not accessible directly from any other subroutine</li> <li>• Local variables are removed from memory when the subroutine ends.</li> <li>• Local variables can be passed as parameters to a function to be updated, and then returned to override the original local variable</li> <li>• Local variables can be passed by reference to a subroutine to allow the content of the variable to be updated</li> </ul>

		<p>which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed.</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><b>Mark Band 1 – Low Level (1-3 marks)</b>  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.</p> <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><b>0 mark</b>  No attempt to answer the question or response is not worthy of credit.</p>		<p><b>AO2: Application</b></p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> </ul> <p><b>AO3: Evaluation</b></p> <ul style="list-style-type: none"> <li>• As this is only a prototype, the use of global variables would be beneficial.</li> </ul>
--	--	---	--	---

					<ul style="list-style-type: none"> <li>• However, when the game expands, the use of global variables could create issues such as running out of memory, coupling, testing &amp; debugging problems and maintenance problems.</li> <li>• The programmer may be best to keep the variables as local and then pass them between the different subroutines as parameters byVal and byRef.</li> </ul> <p><b><u>Examiner's Comments</u></b></p> <p>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.</p>
			<b>Total</b>	<b>40</b>	
3	a		1 mark each <ul style="list-style-type: none"> <li>• Initialisation of counter variable before loop</li> <li>• Condition (e.g. while c &lt;= a)</li> <li>• c incremented in loop</li> </ul>	4	BP2 – Allow any suitable logic for the while loop condition that iterates between 1 and a.  Allow != for <> Allow += or equivalent for c = c + 1

		<div><ul style="list-style-type: none"><li>Completed loop will produce correct results 12, 24 ... 144</li></ul></div> <div>e.g.</div> <div><div><pre>c = 1 while c &lt;= a   print(c * a)   c = c + 1 endwhile</pre></div><div><pre>c = 0 while c &lt; a   c = c + 1   print(c * a) endwhile</pre></div><div><pre>c = 0 while c &lt;&gt; a   c = c + 1   print(c * a) endwhile</pre></div></div>	<div>Allow hard coded values for upper bound such as a = 12 or a = 13 depending on the relational operator used.</div> <div>No marks awarded if a conditional loop has not been used.</div> <div>Max 3 if solution does not completely work.</div> <div><b><u>Examiner's Comments</u></b></div> <div>This question was generally well attempted with marks relatively evenly distributed throughout the range of marks available. There were some off-by-one errors, with the loop counter or the position of the output line being incorrectly positioned before/after the counter increment. Incorrect loops such as count-controlled for loops were rejected. If responses did not produce a fully correct output, marks given were limited to 3 marks.</div> <div>Exemplar 1</div> <div><pre>while c = c = 1 while c &lt; a:   print(c * a)   c = c + 1</pre></div> <div>This response showed a typical off-by-one error where the candidate had not thought through the logic of the entire response. The condition</div>
--	--	--	--

					operator should have been $\leq$ rather than $<$ to iterate through all values of $c$ from 1 to 12.
	b	i	<p>1 mark each</p> <ul style="list-style-type: none"> <li>• A memory location / named storage location</li> <li>• Stores/holds data / a value</li> <li>• That can be changed</li> </ul>	2	<p>Element/identifier on its own is not enough for BP1</p> <p><b><u>Examiner's Comments</u></b></p> <p>Most candidates gained some credit for identifying that a variable holds a value, and this was the most popular response. Element/identifier on its own was insufficient for the first mark point as there had to be a clear indication that the identifier referred to a memory/storage location.</p>
		ii	<p>1 mark for all variables: a b c</p>	1	<p>Must have all three for the mark to be awarded.</p> <p><b><u>Examiner's Comments</u></b></p> <p>The majority of candidates had little difficulty correctly identifying the variables given in the code.</p>
	c		<p>1 mark for do loop and 1 mark for while loop up to a maximum of 2 marks.</p> <ul style="list-style-type: none"> <li>• While loop will check the condition at the <b>start</b> of the loop / pre-condition loop</li> <li>• Do loop will check the condition at the <b>end</b> of the loop / post-condition</li> <li>• The code in a while loop may never run (if the condition is already met)</li> </ul>	2	<p>Answer must cover both do loop and while loop for 2 marks to be awarded</p> <p>BP1 and BP2 must be specific as to the location that the condition is placed</p> <p><b><u>Examiner's Comments</u></b></p> <p>Many responses were vague and did not accurately identify or distinguish that a while loop</p>

		<ul style="list-style-type: none"> <li>The code in a do loop will always run at least one.</li> </ul>		<p>is a pre-condition loop while a do loop is a post-condition loop. Responses had to be clear as to the relative position of the condition in the loop. Accepted responses included a while loop is a pre-condition loop while a do loop is a post-condition loop. Fewer candidates identified that the body of a while loop may not be executed while the body of a do loop will always be executed at least once.</p>
		<b>Total</b>	<b>9</b>	
4	a	<p>1 mark for each completed statement</p> <pre>public procedure new()   for row = 0 to 9     for column = 0 to 19       <b>grid</b>[row, column] = new Treasure(-1,"")     next <b>column</b>   next row endprocedure</pre>	5	<p><b><u>Examiner's Comments</u></b></p> <p>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.</p>
	b	<p>1 mark each to max 7</p> <ul style="list-style-type: none"> <li>Procedure declaration taking parameter</li> <li>Taking two inputs for row and column from the user</li> <li>Accessing item at grid position...</li> <li>...using correct get methods <code>getGridItem</code></li> <li>Checking (treasure) object's level/value...</li> <li>...using correct get method <code>getLevel</code> <code>getValue</code></li> <li>...outputting "No treasure" if empty</li> <li>...otherwise outputting value and level</li> </ul>	7	<p>Note candidates may attempt to access private attributes directly <code>gameboard.grid(x,y)</code> for example, instead of <code>gameboard.getGridItem(x,y)</code>.</p> <p>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.</p> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidate responses gave very little</p>

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

```

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

```


def guessGrid(Board):
def guessGrid(Board):
    x = int(input("Enter x"))
    y = int(input("Enter y"))
    place = Board.getGridItem(x,y)
    if place.getLevel() == "":
        print("No treasure")
    else:
        print("Treasure found! Value is", place.getValue(),
            " it is a", place.getLevel())

```

					<p>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.</p>
	C	<p><b>Mark Band 3 – High level (7–9 marks)</b></p> <p>The candidate demonstrates a thorough knowledge and understanding of parameters and local/global variables; the material is generally accurate and detailed.</p> <p>The candidate is able to apply their knowledge and understanding directly and consistently to the context provided.</p> <p>Evidence/examples will be explicitly relevant to the explanation.</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><b>Mark Band 2 – Mid level (4–6 marks)</b></p> <p>The candidate demonstrates reasonable knowledge and understanding of parameters and local/global variables; 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. Evidence/examples are for the most part implicitly relevant to the explanation.</p> <p>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</p>	9	<p><b>AO1: Knowledge and Understanding</b></p> <p><b>Indicative content</b></p> <ul style="list-style-type: none"><li>• Local variable can only be accessed within sub-program/main program it is declared within</li><li>• Global variable can be accessed by all sub-programs</li><li>• Parameters are items passed to a subproblem</li><li>• Passing by reference sends a pointer to the original value, so this will be changed when control is returned</li><li>• Passing by value sends the a copy of the value, so the original will not be changed when control is returned</li></ul> <p><b>AO2: Application</b></p> <ul style="list-style-type: none"><li>• If board is local it can only be accessed in the main program</li><li>• This will need to be passed to any sub-programs that need to use it</li></ul>	



		<p>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><b>Mark Band 1 – Low Level (1–3 marks)</b></p> <p>The candidate demonstrates a basic knowledge of parameters and local/global variables with limited understanding shown; the material is basic and contains some inaccuracies. The candidate makes a limited attempt to apply acquired knowledge and understanding to the context provided.</p> <p>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> <p><b>0 mark</b></p> <p>No attempt to answer the question or response is not worthy of credit.</p>	<ul style="list-style-type: none"> <li>• If the board needs to be changed it will need passing by reference, so that the board is updated</li> <li>• If it only needs to be accessed and not changed it can be passed by value</li> </ul> <p><b>AO3: Evaluation</b></p> <ul style="list-style-type: none"> <li>• If global then this would be present in memory throughout hence using more memory</li> <li>• ...however the board will be required throughout the program so may be as efficient as passing it through parameters</li> <li>• ...if global then the programming may be more straight forward, and less likely to have errors with passing the board incorrectly to subprograms, i.e. it may not be updated when it needs to be</li> <li>• Using local means that the board can be manipulated by subprograms without affecting the actual board if needed, for example to simulate potential changes.</li> </ul> <p><b><u>Examiner's Comments</u></b></p> <p>A good range of Level 2 responses were observed with competent descriptive definitions of local versus global and byVal/byRef parameter passing.</p> <p>There were far fewer high level AO3 evaluations</p>
--	--	--	--

				<p>of relative memory use within the context of the scenario when passing byVal/byRef for a relatively large grid object. Good, contextualised responses took the elements of the scenario in section B of the paper and talked about the grid object and passing parameters like x and y coordinates. Responses such as this were relatively rare.</p> <div><b>Assessment for learning</b></div> <p>Candidates would benefit from producing solutions centred around the scenario presented in Section B of the paper. Having extensive OOP programming experience in a relevant high level language will help candidates to successfully tackle questions where algorithms have to be presented. Sample Python code based on Section B is presented below for consideration.</p> <b>Assessment for learning</b> <pre># 2023 Section B  class Treasure:      def __init__(self, pValue: int, pLevel: str):         self.__value = pValue</pre>
--	--	--	--	--

					<pre>        self.__level = pLevel      def getValue(self):         return self.__value      def setValue(self,pValue: int):         return self.__pValue      def getLevel(self):         return self.__level      def setLevel(self,pLevel: str):         return self.__pLevel  class Board():      def __init__(self):         self.__grid = [[Treasure(-1,"") for _ in range(20)] for _ in range(10)]      def getGridItem(self,x: int, y: int):         return self.__grid[x][y]      def setGridItem(self,x: int,y: int, pTreasure: Treasure):         self.__grid[x][y] = pTreasure def guessGrid(b: Board):     row = int(input("Row: "))     col = int(input("Col: "))     t = b.getGridItem(row, col)</pre>
--	--	--	--	--	---

					<pre>         if t.getLevel() == "":             print("No treasure")         else:             print(f"Treasure found at {row} {col}!")             print(f"Level {t.getLevel()} Value {t.getValue()}")  # Test code - Treasure @ 2,2 all other locations no treasure island = Board() prize = Treasure(5,"Prize!") island.setGridItem(2,2,prize) guessGrid(island) </pre>
			<b>Total</b>	<b>21</b>	
5		i	1 mark for: <ul style="list-style-type: none"> <li>• isInteger</li> <li>• number</li> <li>• result</li> <li>• count</li> <li>• asciiValue</li> </ul>	1	Penalise excessive spaces in identifiers such as <i>ascii Value</i> instead of <i>asciiValue</i>  <b><u>Examiner's Comments</u></b>  This question was generally well done (although slightly less well done than parts (a) (ii) and (a) (iii)), but many candidates did very well. The most common erroneous responses were giving the names of predefined functions/properties or giving relational operators.
		ii	(0)5	1	<b><u>Examiner's Comments</u></b>  This question required an exact answer only and

					was answered correctly by the majority of candidates.
		iii	(0)3	1	<p><b><u>Examiner's Comments</u></b></p> <p>This question also required an exact answer only and was answered correctly by the majority of candidates.</p>
			<b>Total</b>	<b>3</b>	
6			<p>1 mark each to max 3</p> <ul style="list-style-type: none"> <li>• The function calls itself....</li> <li>• .....such as line 05 / 07</li> <li>• Each recursive call will create a new copy of the values in the function....</li> <li>• ....and add all of the values of the copy the call is being made from to a stack</li> <li>• There is a base case / condition that stops the recursive calls...</li> <li>• ...condition in line 02</li> <li>• There may be more than one base case</li> </ul>	3	<p>Allow answers in context as long as they are clear what the features are.</p> <p><b><u>Examiner's Comments</u></b></p> <p>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.</p>
			<b>Total</b>	<b>3</b>	
7			<p>1 mark for identification, 1 for description of feature e.g.</p> <ul style="list-style-type: none"> <li>• Error diagnostics</li> <li>• ... to locate and fix errors</li> </ul>	6	<p>Consider awarding description without feature.</p> <p>Allow other suitable answers.</p> <p><b><u>Examiner's Comments</u></b></p>

		<ul style="list-style-type: none"> <li>• Breakpoints</li> <li>• ...stop a program running at a point to check variables</li> <li>• Syntax highlighting</li> <li>• ... to identify key words, variables and help identify syntax errors</li> <li>• Stepping / step through</li> <li>• ... run the program line by line to check variable values at each stage</li> <li>• Variable watch window</li> <li>• ...view how variables change while the program executes</li> <li>• Auto-complete</li> <li>• ... start typing a command/identifier and it completes it</li> </ul>		<p>Most candidates scored well on this question which lent itself to a wide variety of potential answers. Some candidates quoted 'autocorrect' rather than auto complete or suggested predictive code. IDEs can have integrated run-time environments and translation software built-in or attached, but some responses were rather vague or included repetition and did not make it clear that they were referring to an integrated system. Sometimes the descriptive expansion or the initial identification of the point was vague and required the identification plus the description boxes to be taken in combination to give the mark. This was particularly evident when candidates gave a very vague 'debugger/debugging' identification.</p>
		<b>Total</b>	<b>6</b>	
8	i	it can only be accessed within the subroutine/block in which it is declared	1	<p><b><u>Examiner's Comments</u></b></p> <p>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.</p>
	ii	<p>1 mark for benefit e.g.</p> <ul style="list-style-type: none"> <li>• Increases data integrity</li> </ul>	2	<p><b><u>Examiner's Comments</u></b></p> <p>Some candidates confused the terms local variable and global variable and gave de finitions</p>


		<ul style="list-style-type: none"> <li>• More efficient memory usage</li> <li>• Stops other subroutines accidentally altering variable</li> </ul> <p>1 mark for drawback e.g.</p> <ul style="list-style-type: none"> <li>• Cannot be accessed directly by other subroutines</li> <li>• It has to be passed into a subroutine as a parameter</li> </ul>		the wrong way round. A significant number of responses demonstrated conceptual misconceptions. The contents of <code>dataArray</code> 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.
		<b>Total</b>	<b>3</b>	
9		<p>1 mark each</p> <p>03</p> <ul style="list-style-type: none"> <li>• Loop through each of the <b>characters/digits</b> in the <code>number</code> string (parameter)</li> </ul> <p>04</p> <ul style="list-style-type: none"> <li>• Find the ASCII value of the current <b>character/digit</b></li> </ul> <p>09</p> <ul style="list-style-type: none"> <li>• Return true if the value is an integer and false otherwise</li> </ul>	3	<p><b><u>Examiner's Comments</u></b></p> <p>This question required the <i>purpose</i> of the lines of code to be described, but many candidates just described the functionality of the lines of code such as 'line 03 is a counter controlled loop from 0 to <code>number.length - 1</code>'. The expected purpose of this line of code was to set up a loop to iterate through each character in the input string parameter.</p> <p>While many candidates could describe the purpose of at least one of the lines of code given, few could clearly describe the purpose of all three lines.</p>
		<b>Total</b>	<b>3</b>	
10		<p><b>Mark Band 3–High Level (7–9 marks)</b></p> <p>The candidate demonstrates thorough knowledge and</p>	9	<p><b>AO1: Knowledge and Understanding</b></p> <p>e.g. IDE:</p>

		<p>understanding of IDEs; the material is generally accurate and detailed.</p> <p>The candidate is able to apply their knowledge and understanding directly and consistently to the context provided.</p> <p>Evidence/examples will be explicitly relevant to the explanation.</p> <p>The candidate provides a thorough discussion which is well-balanced. Evaluative comments are consistently relevant and well-considered.</p> <p>There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.</p> <p><b>Mark Band 2-Mid Level (4–6 marks)</b></p> <p>The candidate demonstrates reasonable knowledge and understanding of IDEs; 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. Evidence/examples are for the most part implicitly relevant to the explanation.</p> <p>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.</p> <p>There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.</p> <p><b>Mark Band 1-Low Level (1–3 marks)</b></p> <p>The candidate demonstrates a basic knowledge of IDEs, with</p>		<ul style="list-style-type: none"> <li>• pretty print / syntax highlighting</li> <li>• auto-complete</li> <li>• auto-correction</li> <li>• breakpoints</li> <li>• stepping</li> </ul> <p>Editor:</p> <ul style="list-style-type: none"> <li>• no helpful writing/debugging features</li> <li>• no excess features/interface</li> </ul> <p><b>AO2.1: Application</b> e.g. IDE</p> <ul style="list-style-type: none"> <li>• identify syntax errors as writing</li> <li>• ...saves trying to find them</li> <li>• easier debugging as can step through a program</li> <li>• auto-indenting avoids errors from incorrect indentation</li> <li>• May have built in unit testing to automate testing and avoid new errors being introduced.</li> </ul> <p>Editor</p> <ul style="list-style-type: none"> <li>• does not offer suggestions on code corrections</li> </ul>
--	--	---	--	---



		<p>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 a limited discussion which is narrow in focus. Judgments if made are weak and unsubstantiated.</p> <p>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.</p> <p><b>0 mark</b></p> <p>No attempt to answer the question or response is not worthy of credit.</p>	<ul style="list-style-type: none"> <li>• Has a lower footprint on memory/CPU which may be suited to quick alterations or working on lowed spec'd systems.</li> <li>• May be better when learning to program as it forces the user to type everything in full / doesn't give suggestions, helping things stick in memory.</li> </ul> <p><b>AO3.3: Evaluation</b></p> <p>e.g.</p> <ul style="list-style-type: none"> <li>• IDE is helpful in reducing original errors</li> <li>• IDE is helpful in finding and correcting errors</li> <li>• Editor is a simpler system to use e.g. less memory needed to run it, does not try and auto-correct incorrectly or introduce errors that the programmer has not made.</li> </ul> <p><b><u>Examiner's Comments</u></b></p> <p>A pleasing number of candidates could identify and describe a range of features of an IDE, with most being able to demonstrate sufficient knowledge to achieve mark band Level 2. Few candidates displayed a level of evaluative reasoning of sufficient depth to achieve mark band Level 3. More successful responses clearly explained how relevant debugging tools in an IDE led to much greater productivity than a text editor</p>
--	--	---	---

					<p>because of the way that they could assist with finding logical errors in code through the use of stepping and tracing. An acknowledgement of the amount of system resources and the complexity of some IDE environments was less often evaluated.</p> <p>Candidates need to focus on making clear logical arguments with evidence of evaluation (AO3) to be able to achieve mark band 3.</p>
			<b>Total</b>	<b>9</b>	
11	a		<p>1 mark per bullet to max 6</p> <ul style="list-style-type: none"><li>• function header taking parameter</li><li>• looping appropriately e.g. until value is 0</li><li>• dividing by 2 and finding remainder e.g. MOD</li><li>• adding 1 or 0 correctly</li><li>• ...appending to a value to be returned / final string reversed</li><li>• reducing value to use within loop</li><li>• returning calculated value</li></ul> <p>e.g.</p> <pre>function toBinary(denary)   binaryValue=""   while denary &gt; 0     temp = denary MOD 2     if temp == 1 then       binaryValue = "1" + binaryValue     else       binaryValue = "0" + binaryValue     endif</pre>	6	<p>Award a recursive algorithm as equivalent</p> <p><b><u>Examiner's Comments</u></b></p> <p>Very few candidates were able to produce a working function, but many gained some marks.</p> <p>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.</p> <p>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.</p>

		<pre> denary = denary DIV 2 endwhile return binaryValue endfunction </pre>		<p>More successful responses demonstrated an ability to problem solve, think logically, and present clear working functions.</p>
	b	<p>1 mark per bullet to max 4</p> <ul style="list-style-type: none"> <li>• taking value as input</li> <li>• looping until valid between 1 and 255</li> <li>• calling function with correct parameter</li> <li>• outputting return value</li> </ul> <pre> denary = -1 while denary &lt; 1 or denary &gt; 255     denary = input("Enter denary value between 1 and 255") endwhile print(toBinary(denary)) </pre>	4	<p>Allow other checks for a valid number. For example</p> <pre> denary.isInteger == False </pre> <p><b><u>Examiner's Comments</u></b></p> <p>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 <i>and</i>, and vice-versa. When candidates called <code>toBinary(inputVal)</code>, the result was not always stored for later use.</p> <p> <b>Assessment for learning</b></p> <p>When preparing candidates for the examination, they will benefit from a wide range of programming experience.</p>

					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.
			<b>Total</b>	<b>10</b>	
12	a	i	sequence	1	<b><u>Examiner's Comments</u></b> Nearly all candidates correctly identified 'sequence' as the correct construct.
		ii	selection / branching	1	<b><u>Examiner's Comments</u></b> Nearly all candidates correctly identified 'selection' as the correct construct.
	b		1 mark each to max 2 <ul style="list-style-type: none"> <li>total</li> <li>smallest</li> <li>largest</li> <li>x</li> <li>dataArray</li> </ul>	2	<b><u>Examiner's Comments</u></b> Most candidates correctly identified two variables from the given code.
	c		1 mark for each line and correction <ul style="list-style-type: none"> <li>Line 01 total = 0</li> <li>Line 02 smallest = dataArray[0]</li> <li>Line 04 for x = 0 to 19 (accept 20)</li> </ul>	4	Do not award a mark for the line number alone without correction. <b><u>Examiner's Comments</u></b> Most candidates were given some marks, but fewer achieved full marks.

			<ul style="list-style-type: none"> <li>Line 07 <code>if dataArray[x] &gt; largest then</code></li> <li>Line 14 <code>print("Average = " + total / 20)</code></li> </ul>		The context of the question indicated that the numbers input were positive integers, but candidates did not always appreciate this.
			<b>Total</b>	<b>8</b>	
13	a	i	It returns a value	1	<p><b><u>Examiner's Comments</u></b></p> <p>Most candidate correctly identified that <code>checkValid()</code> was a function because it returned a value. Some candidates erroneously stated 'because it has an output' which did not differentiate a function from a procedure since both can print an output as a side effect.</p>
		ii	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>If the players card is the same suit return true</li> <li>if the players card is the same number return true</li> <li>if neither is true, return false</li> </ul>	3	<p>Allow 1/0, 'T'/'F', "Yes"/"No" or any sensible alternative as return values.</p> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidates were able to think computationally to identify the points in a solution where a decision had to be taken. Occasionally some candidates did not specify the specific result that would be returned from the function after determining if a card was valid or not. There were also occurrences of incorrect logic such as returning valid using an AND condition on the same number and same suit clauses, rather than executing them in sequence.</p>

		b	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>• identifier <code>cards...</code></li> <li>• ...with 2 dimensions</li> </ul>	2	<p><b><u>Examiner's Comments</u></b></p> <p>Many candidates gained a mark by initialising the identifier <code>cards</code>, 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.</p>
			<b>Total</b>	<b>6</b>	
14	a	i	Line number 5	1	<p><b><u>Examiner's Comments</u></b></p> <p>Nearly all candidates gave the correct answer line 5.</p>
		ii	<p>1 mark per feature</p> <ul style="list-style-type: none"> <li>• A function that calls itself / a function that is defined in terms of itself</li> <li>• ...has a base case (that terminates the recursion)</li> </ul>	2	<p><b><u>Examiner's Comments</u></b></p> <p>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</p>

					just wrote ‘stopping condition’ it was too vague as it was unqualified since they could have been talking about any conditional loop.																								
	b		<table><thead><tr><th>Function call</th><th>number</th><th>return</th><th>Marking Guidance</th></tr></thead><tbody><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></tbody></table>	Function call	number	return	Marking Guidance	calculate(5)	5	15	1 Mark	calculate(4)	4	10	1 Mark	calculate(3)	3	6	1 Mark	calculate(2)	2	3	1 Mark	calculate(1)	1	1	1 Mark	5	<p><b><u>Examiner’s Comments</u></b></p> <p>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.</p>
Function call	number	return	Marking Guidance																										
calculate(5)	5	15	1 Mark																										
calculate(4)	4	10	1 Mark																										
calculate(3)	3	6	1 Mark																										
calculate(2)	2	3	1 Mark																										
calculate(1)	1	1	1 Mark																										
	c		calculate(10)	1	<p><b><u>Examiner’s Comments</u></b></p> <p>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.</p>																								
			Total	9																									
15	a	i	1 mark for correct data 1 mark for correc top of stack pointer	2	<p><b><u>Examiner’s Comments</u></b></p> <p>Most candidates scored full marks for this question, but the most common error was the omission of the pointerValue or the setting of</p>																								

			<div><div>pointerValue6</div><div><div><div>Index</div><div>Data</div></div><table><tr><td>8</td><td></td></tr><tr><td>7</td><td></td></tr><tr><td>6</td><td></td></tr><tr><td>5</td><td>7</td></tr><tr><td>4</td><td>6</td></tr><tr><td>3</td><td>3</td></tr><tr><td>2</td><td>6</td></tr><tr><td>1</td><td>5</td></tr><tr><td>0</td><td>10</td></tr></table></div></div>	8		7		6		5	7	4	6	3	3	2	6	1	5	0	10		the <code>pointerValue</code> to 5 to point to the last element in the stack instead of the next free location at the top of the stack.
8																							
7																							
6																							
5	7																						
4	6																						
3	3																						
2	6																						
1	5																						
0	10																						
		ii	1 mark per bullet <ul style="list-style-type: none"><li>Point to the next free space in the array</li><li>Points to the top of the stack</li></ul>	1	<b><u>Examiner’s Comments</u></b>  Many candidates answered the question clearly specifying a pointer to the next free index in the stack, but some candidates gave vague responses such as ‘location of next item’.																		
	b	i	1 mark per correctly completed statement  e.g. <pre>public function pop()   if pointerValue == 0 then     return -1   else     pointerValue = pointerValue -1     returnValue = stackArray[pointerValue]     return returnValue   endif endfunction</pre>	5	<b><u>Examiner’s Comments</u></b>  Most candidates gained some marks with many gaining full marks. The most common errors were writing identifiers for <code>pointerValue</code> and <code>returnValue</code> with spaces in, returning values as strings, or returning True/False instead of -1 as required in the question.																		
		ii	1 mark per bullet to max 6 <ul style="list-style-type: none"><li>function header</li><li>..taking parameter (ignore byval/byref)</li><li>checking if stack is full (<code>pointerValue</code> at 100)...</li><li>...and returning false</li><li>(otherwise) adding value to top of stack</li></ul>	6	Ignore additional parameters in function definition  Do not accept the return of string values  FT following a reasonable attempt to check if the stack is full																		



		<ul style="list-style-type: none"> <li>• ...incrementing top of stack pointer</li> <li>• return true</li> </ul> <p>e.g.</p> <pre>function push(value)   if pointerValue &lt; 100 then     stackArray[pointerValue] = value     pointerValue = pointerValue + 1     return true   else     return false   endif endfunction</pre>	<p><b><u>Examiner's Comments</u></b></p> <p>Candidates produced a higher standard of pseudocode this session and many scored most if not full marks for a standard stack push routine.</p> <p>Common errors included omitting a parameter and/or getting a user input as the value to place into the stack, returning strings or printing the return values, and off-by-one errors when testing to see if the <code>stackPointer</code> was at the top of the stack to determine if the stack was full.</p> <p>It was noticeable that a number of students who were only familiar with Python gave list append type solutions rather than using the array and pointers as per the implementation given.</p> <p>Another common error was incrementing the value of the <code>stackPointer</code> before the parameter was assigned to <code>stackArray</code> at the <code>stackPointer</code> index, which was frequently seen when candidates did not know that the <code>stackPointer</code> actually pointed to the index of the next free space in the stack.</p> <p><b>Exemplar 3</b></p>
--	--	--	--

					<pre> function push(num)   if pointerValue == stackArray.length then     return false   else     stackArray[pointerValue] = num     pointerValue++     return true   endif endfunction </pre> <p>Candidates are encouraged to present pseudocode solutions with clear indentation to aid readability. No specific language/syntax is expected, but the logic of the solution must be clear.</p>
		iii	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>instantiation of new object of type stack</li> <li>assigned to variable mathsStack</li> </ul> <pre> mathsStack = new stack() </pre>	2	<p><b>Accept</b>  <code>mathsStack = stack()</code></p> <p>allow missing brackets this time only e.g.  <code>mathsStack = stack</code></p> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidates struggled to answer this question because they lacked practical experience of OOP that showed a lack of familiarity in terms of creating instances of a class. Some candidates tried to declare <code>mathsStack</code> as a procedure or class, and many got it the wrong way round and actually declared a new identifier called <code>stack</code> as an instance of the class <code>mathStack</code>.</p>
		iv	1 mark for each completed statement	4	Accept equivalent for print e.g. output

		<pre> returnValue = true while returnValue == true     returnValue = mathsStack.push(input("Enter Number"))     if returnValue == false then         print("Stack full")     endif </pre>		<p><b><u>Examiner's Comments</u></b></p> <p>The majority of candidates scored some marks but relatively few gained full marks. Return was seen instead of print/output when the question did, on this occasion ask for the message to be output (as the code was specified as being in the main program rather than a subroutine).</p>
	V	<p>mark per bullet to max 8</p> <ul style="list-style-type: none"> <li>• initialise a total to 0 outside of loop</li> <li>• looping</li> <li>• removing an item from the stack using the method pop</li> <li>• check if stack is empty</li> <li>• (if not) add value returned to total</li> <li>• ...outputting total</li> <li>• counting how many values are returned</li> <li>• stopping loop when either 20 items removed or no items left</li> </ul> <pre> total = 0 quantity = 0 returnValue = 0  while quantity&lt;20 and retunValue!=-1     returnValue = mathsStack.pop()     if(returnValue != -1) then         quantity = quantity + 1         total = total + returnValue         print(total)     endif endwhile </pre>	8	<p><b><u>Examiner's Comments</u></b></p> <p>The standard of pseudocode / programming code was better than in previous sessions and most candidates made a reasonable attempt to pop 20 values from the stack. Many candidates found it difficult to reference the class methods correctly or made assumptions about the existence of other methods that were not provided within the scenario (e.g. <code>.full()</code>, <code>.empty()</code>, <code>.length()</code>, <code>.remove()</code>) that should not be presumed to exist.</p> <p>Those who had little understanding of encapsulation often tried to access class attributes such as <code>stackPointer</code> directly to manipulate <code>mathsStack</code>, rather than using methods to interact with the instance of the stack.</p> <p>When candidates did use the <code>.pop()</code> method to retrieve a value from <code>mathsStack</code> they frequently did not store the result for later use to check whether then stack was empty.</p>

			<b>Total</b>	<b>28</b>	
16			<p>1 mark per bullet e.g.</p> <ul style="list-style-type: none"> <li>• Store data that has been used in cache/RAM in case needed again</li> <li>• e.g. store design of the weather/a cloud/external environment</li> </ul>	2	<p>Allow 2 valid examples for 2 marks</p> <p><b><u>Examiner's Comments</u></b></p> <p>Responses to caching were better than those seen in recent papers and many candidates managed to either describe the concept of caching or give an example of something that would realistically be cached. Far fewer managed to do both in detail.</p>
			<b>Total</b>	<b>2</b>	
17	a	i	<p>1 mark for each variable</p> <ul style="list-style-type: none"> <li>• <code>contents</code></li> <li>• <code>count</code></li> <li>• <code>numberOfWords</code></li> <li>• <code>words / words[]</code></li> </ul>	2	<p>Accept exact spelling only</p> <p>Do not award <code>numberOfWords</code> if there are obvious spaces in 'number of Words'. It must be a valid identifier.</p> <p><b><u>Examiner's Comments</u></b></p> <p>The majority of candidates answered correctly, with the most popular answers being <code>count</code> and <code>contents</code>. A few candidates incorrectly gave data values from the array rather than identifying variables in the function.</p>
		ii	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>• By reference the function receives the memory location of the data</li> </ul>	2	<p>Must cover <code>byVal</code> and <code>byRef</code> for 2 marks to be awarded.</p> <p>Must be clear that <code>byVal</code> <u>is a copy</u> of the original</p>

		<ul style="list-style-type: none"> <li>• By value the function receives a copy of the variable</li> <li>• By reference will make changes to the original variable</li> <li>• By value will make changes to the copy of the variable</li> <li>• By reference will overwrite data in the original variable</li> <li>• By value will not overwrite the data in the original variable</li> <li>• By reference will keep the changes after the function ends</li> <li>• By value will not keep the changes after the function ends</li> </ul>		<p>value.</p> <p><b><u>Examiner's Comments</u></b></p> <p>It was pleasing to see an improvement in responses to this topic this session. There were still some answers that were too vague that did not specify that by <i>value</i> uses a <u>copy</u> of the parameter and that by <i>reference</i> passes the <u>memory address</u>. Other examples of vagueness that were not given marks included answers such as 'by value can't change the value while by reference can' that did not qualify the scope within which changes can be made. A variable passed by value can clearly be changed in a function, but it is the local copy that is changed and then disregarded when the function finishes.</p>
	iii	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>• initialising a counter</li> <li>• looping between 0 and numberOfWords -1</li> <li>• incrementing counter inside loop</li> <li>• remainder of algorithm correct (initialisation, concatenation and return)</li> </ul> <p>e.g.</p> <pre> contents = "" count = 0 while count &lt; numberOfWords     contents = contents + words[count] + " "     count = count + 1 </pre>	4	<p>Accept:</p> <pre>while count &lt;= numberOfWords - 1</pre> <p>Accept other combinations for example counting from 1 and then subtracting 1 for the array element (but do not credit off by one errors)</p> <p>Accept:</p> <pre>len(words) for numberOfWords</pre> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidates struggled with this question. Some candidates rewrote the <code>for</code> loop putting the word <i>while</i> in place of <i>for</i> showing little</p>

		<pre>endwhile return contents</pre>		<p>understanding of the difference between a counter-controlled and a conditional loop.</p> <p>Common errors included not initialising the <code>count</code> variable before using it within the body of the while loop, off-by-one errors, and forgetting to increment the <code>count</code> variable within the loop. Poor indentation was often a problem, and a number of candidates erroneously placed the return statement inside the body of the loop.</p> <p><b>Off-by-One errors</b> There were many off-by-one errors observed, e.g. <code>while count &lt;= numberOfWords</code> rather than <code>while count &lt; numberOfWords</code>.</p> <p>Candidates need to give code that is logically accurate, and, in this instance, it required the loop to run the correct number of times so that all the words in the array were processed.</p>
	b	<p>1 mark for benefit, 1 mark for drawback e.g. Benefits:</p> <ul style="list-style-type: none"> <li>• Variable doesn't need passing as a parameter (byref)</li> <li>• You don't need to return a value</li> <li>• Can be accessed from any function / anywhere in the program</li> </ul> <p>Drawback:</p>	2	<p><b><u>Examiner's Comments</u></b></p> <p>Many candidates found it easier to describe a benefit than to give a drawback. The most commonly identified benefit was that the array would have global scope (and would therefore not need to be passed as a parameter), but often the descriptions given were too vague, e.g. 'can be accessed anywhere'. The correct technical vocabulary is required.</p> <p>Drawbacks were poorly described. Potential side</p>

			<ul style="list-style-type: none"> <li>Increases memory usage (as it is used until full program execution is over)</li> <li>Alterations within the function may have unwanted side effects elsewhere in the program.</li> </ul>		<p>effects and resultant complexity debugging were frequently alluded to as 'accidental change' but not fully developed into complete qualified points.</p> <p>There was also a frequent misconception that you cannot have multiple variables with the same name, which is not true. When a local variable is declared with the same name as a global variable that already exists, it takes precedence within the local scope.</p>
	c		<p>1 mark per identification 1 mark for expansion, max 2 each. Write: e.g.</p> <ul style="list-style-type: none"> <li>Auto-complete</li> <li>Start typing an identifier/command and it fills in the rest</li> <li>Auto-indent</li> <li>Indents code automatically within structures to avoid errors</li> <li>Coloured command words / pretty printing / syntax highlighting</li> <li>Shows which commands are correct / help identify key elements</li> </ul> <p>Test e.g.</p> <ul style="list-style-type: none"> <li>Breakpoints</li> <li>Stop the program running at a set point to check variables</li> <li>Variable watch window</li> <li>Display the values of the variables while the program is run</li> <li>Stepping</li> </ul>	4	<p><b><u>Examiner's Comments</u></b></p> <p>Most candidates achieved at least 2 marks by identifying two suitable features, but often the expansions to describe each feature were less clear. Developments in modern IDEs meant that there were many valid features, but auto-completion and breakpoints were particularly common answers.</p>

		<ul style="list-style-type: none"> <li>Run one line at a time and check variables</li> </ul> <p>Unit Testing</p> <ul style="list-style-type: none"> <li>Automated tests to be run to check changes ensure changes haven't introduced errors.</li> </ul>		
d		<p>1 mark per bullet to max 2</p> <ul style="list-style-type: none"> <li>Saves time from having to write the same algorithm repeatedly</li> <li>Reduced testing requirements</li> <li>Can be taken and used in different programs as well as the program they are written in / can be used in a program library</li> </ul>	2	<p>Allow other suitable answers</p> <p><b><u>Examiner's Comments</u></b></p> <p>Many candidates scored 1 mark, but fewer gave two clear benefits. Unqualified statements that were too vague were often given. A frequent response was 'saves time', but as an unqualified point it did not state how time was saved, e.g. 'saves time as you don't have to retype the same code/routine again'.</p> <p>Candidates need to be reminded that at this level responses of a 'quicker' or 'easier' nature will not gain marks unless qualified.</p> <p><b>Exemplar 1</b></p> <p><i>1. Saves time, as code has already been tested and written so does not need to be tested and written again (time saving).</i>  <i>2. Easier to debug and write, as each component can be tested individually. (Shows you are well-structured and thorough).</i></p> <p>Candidates are expected to fully qualify the points that they make. This response shows clear identification of points followed by qualifying statements.</p>



			<b>Total</b>	<b>16</b>	
18			<ul style="list-style-type: none"> <li>2,4,6,8</li> </ul>	1 (AO3.3) (1)	<b><u>Examiner's Comments</u></b> In many cases candidates gave the response 2 and did not iterate through the loop until the initial loop condition was false.
			<b>Total</b>	<b>1</b>	
19			A procedure does not return a value / a function will return a value.	1 (AO2.2) (1)	<b><u>Examiner's Comments</u></b> Nearly all candidates successfully identified the factual knowledge that a function returns a value.
			<b>Total</b>	<b>1</b>	
20	a		1 mark per bullet up to a maximum of 2 marks, e.g.: <ul style="list-style-type: none"> <li>Line 04 is calling the procedure</li> <li>Line 06 is defining the procedure</li> </ul>	2 (AO3.3) (2)	<b><u>Examiner's Comments</u></b> Many candidates responded correctly and could identify the difference between a procedure call and a procedure declaration.
	b		1 mark per bullet up to a maximum of 2 marks, e.g.: <ul style="list-style-type: none"> <li>the variable <code>change</code> is global (set on line 03)</li> <li>the value is printed after it has been changed to 0 by the procedure</li> </ul>	2 (AO3.3) (2)	<b><u>Examiner's Comments</u></b> Many candidates identified that the procedure would reduce the <code>change</code> to 0 but few identified that it was because <code>change</code> was a global variable which meant that the initial value was being decreased to zero or that line 22 would need to be moved to before line 10 to resolve the error.
	c		1 mark per bullet up to a maximum of 3 marks, e.g.: <ul style="list-style-type: none"> <li>to convert/cast the values</li> <li>...from an integer to a string</li> </ul>	3 (AO2.2) (3)	Allow "To typecast the variables"  <b><u>Examiner's Comments</u></b> Most candidates identified that the values needed to be converted from integer to string otherwise

		<ul style="list-style-type: none"> <li>to allow the values/labels to be joined/concatenated / to allow them to be printed together</li> </ul>		an error would be generated because they could not be concatenated. Fewer identified the process as type casting.
		<b>Total</b>	<b>7</b>	
21	a	<p>1 mark for any of the following bullet points:</p> <ul style="list-style-type: none"> <li>To write programming code.</li> <li>To debug programming code.</li> <li>To compile/interpret code.</li> </ul>	<p>1 (AO1.1) (1)</p>	<p><b><u>Examiner's Comments</u></b> Most candidates showed good recall of knowledge stating that an IDE was used to develop code.</p>
	b	<p>1 mark per bullet up to a maximum of 2 marks for each construct (4 marks in total), e.g.:</p> <ul style="list-style-type: none"> <li>Sequence... <ul style="list-style-type: none"> <li>...e.g. display payment details once a room has been selected</li> </ul> </li> <li>...e.g. send confirmation email after successfully entering payment details</li> <li>Selection... <ul style="list-style-type: none"> <li>...e.g. if payment details are successful then send confirmation email</li> </ul> </li> <li>...e.g. if a valid date has been entered then display a list of available rooms</li> <li>Iteration.... <ul style="list-style-type: none"> <li>...e.g. repeat code until a date has been entered</li> <li>...e.g. repeat code until a room has been selected</li> </ul> </li> </ul>	<p>4 (AO1.2) (2) (AO2.1) (2)</p>	<p>Award 1 mark for stating a construct and then 1 mark for a suitable example that is relevant to the context.</p> <p>Award a maximum of two marks for each programming construct.</p> <p><b><u>Examiner's Comments</u></b> Most candidates had factual knowledge of the specification point that lists sequence, iteration and branching as types of programming construct. However, many candidates found it more difficult to demonstrate computational thinking skills to logically break down the problem given in the stem of the question to give relevant examples.</p> <p>Several candidates did confuse programming constructs with either computational thinking concepts such as abstraction and decomposition</p>


					or with other elements of programming such as procedures, functions or classes.
	c		<p>1 mark per bullet up to a maximum of 2 marks for each advantage (4 marks in total), e.g.:</p> <ul style="list-style-type: none"> <li>• Saves times (1) as code does not need to be rewritten (1)</li> <li>• Code may already be tested (1) which will therefore save development/debugging time (1)</li> <li>• More efficient code (1) which will use less memory/be easier to maintain (1)</li> <li>• May require less technical knowledge (1) as code can be used rather than written (1)</li> </ul>	<p>4 (AO1.1) (2) (AO1.2) (2)</p>	<p>Award a maximum of two marks for each advantage.</p> <p>Allow other suitable examples.</p> <p><b><u>Examiner's Comments</u></b> Many candidates did not give a clear explanation for a point made, and there were a number of 'cheaper' or 'quicker' type responses that had no qualification for what was being alluded to as 'cheaper' or 'quicker'.</p> <p><b><u>Exemplar 1</u></b></p> <p><i>1. Answer: you do not divide different elements in a program, as once they have been tested, you know they are error free. Can be used individually.</i></p> <p><i>2. Reduces the number of programming errors as you will be using other programmers' expertise with pre-made components. This also reduces the development time as not every component needs to be developed from scratch.</i></p> <p>A response that shows a clear point being made with a detailed explanatory expansion for both benefits.</p>
			<b>Total</b>	<b>9</b>	
22	a	i	<p>1 mark per bullet up to a maximum of 4 marks, e.g.:</p> <ul style="list-style-type: none"> <li>• makes use of a (while) loop...</li> <li>• ....which will continually check the telephone number....</li> </ul>	<p>4 (AO2.2) (2)</p>	<p><b><u>Examiner's Comments</u></b> Most candidates achieved some marks for this question, but few achieved complete responses. Many identified iteration within the first version as key to repeating the sequence until the validation</p>

		<ul style="list-style-type: none"> <li>• ...and repeatedly ask for a telephone number to be re-entered....</li> <li>• ....until the rules have been met</li> </ul>	(AO3.3) (2)	conditions were met. Fewer could explain how version two of the code would allow invalid data to be entered into the system.
	ii	<p>1 mark for stating a suitable procedure name and 1 mark for a suitable purpose (4 marks in total), e.g.:</p> <ul style="list-style-type: none"> <li>• <b>Procedure Name:</b> registerAccount</li> <li>• <b>Purpose:</b> To load up the registration page when the user wants to create a new account</li> <li>• <b>Procedure Name:</b> validateDetails</li> <li>• <b>Purpose:</b> To check the suitability of a new username and password entered</li> <li>• <b>Procedure Name:</b> checkLogin</li> <li>• <b>Purpose:</b> Checks a username and password matches those stored when a customer logs in</li> <li>• <b>Procedure Name:</b> checkStock</li> <li>• <b>Purpose:</b> Checks an item is in stock before the item can be added to a shopping basket</li> </ul>	4 (AO2.1) (4)	<p>The procedure name must be relevant to the context in the question.</p> <p>Both a procedure name and purpose must be stated for 2 marks.</p> <p>Allow other suitable examples that are relevant to the context.</p> <p><b><u>Examiner's Comments</u></b> Candidates needed to apply computational thinking and show procedural identification. Most scored full marks and identified suitable procedures such as checking login details or adding items to shopping basket with clear descriptions.</p>
	iii	<p>1 mark per bullet up to a maximum of 4 marks, e.g.:</p> <ul style="list-style-type: none"> <li>• by reference will receive the memory location of where the data/variable is stored</li> <li>• by value will receive a copy of the data/variable</li> </ul>	2 (AO1.2) (2)	<p>Must cover by reference and by value to be given full marks.</p> <p><b><u>Examiner's Comments</u></b> More candidates were able to produce accurate definitions of passing by value and passing by reference than in previous series.</p>

		<p><b>Mark Band 3 – High level (7-9 marks)</b>  The candidate demonstrates a thorough knowledge and understanding of modularity; 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 modularity within a team of programmers which results in a supported and realistic judgment as to whether it is suitable to use within the 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><b>Mark Band 2 – Mid level (4-6 marks)</b>  The candidate demonstrates reasonable knowledge and understanding of modularity; 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 modularity 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><b>Mark Band 1 – Low Level (1-3 marks)</b>  The candidate demonstrates a basic knowledge of modularity with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to</p>	<p>9  (AO1.1)  (2)  (AO1.2)  (2)  (AO2.1)  (2)  (AO3.3)  (3)</p>	<p><b>Knowledge and Understanding</b></p> <ul style="list-style-type: none"> <li>• A module is a set/block of instructions which is given a name which can then be called upon by the main body of code to complete a task(s).</li> <li>• The different modules within the program are then executed when the program is run in order to provide a solution to the problem.</li> <li>• A module can be created using a function which is a block of code that will return a value, or a procedure which is a block of code that will not return a value.</li> <li>• Alternatively, a module may contain a collection of different sub-programs (procedures/functions) that are combined together to complete a task.</li> </ul> <p><b>Application</b></p> <ul style="list-style-type: none"> <li>• The program is a large program and therefore the use of modules will allow the program to be split into different subtasks.</li> <li>• Different programmers can agree the procedure names and parameters needed for each module.</li> <li>• Each subtask can then be written and tested by different programmers simultaneously which will therefore reduce development time.</li> </ul>
--	--	--	--	--

		<p>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 in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.</i></p> <p><b>0 marks</b> No attempt to answer the question or response is not worthy of credit.</p>	<ul style="list-style-type: none"> <li>Each subtask can be self-contained which means that the work carried out by one programmer should not affect another.</li> <li>Each subtask can be programmed by specific programmers who have the skills and expertise in that area/language being used.</li> </ul> <p><b>Evaluation</b></p> <ul style="list-style-type: none"> <li>Modularity will reduce development time as the team of programmers can work together at the exact same time on the same program.</li> <li>Programmers can specialise in their area of expertise/language which therefore means blocks can be reused by other programmers who may not have expertise in some areas.</li> <li>Programmers may feel restricted if they have to follow the agreed house style. This could have a knock-on effect with other programmers if not followed.</li> </ul> <p><b><u>Examiner's Comments</u></b> Candidates continue to struggle with the banded response question on the paper and most responses did not have a clear structure.</p> <p>These questions always require candidates to</p>
--	--	---	--

					<p>show knowledge by defining the terms being used, applying knowledge to the specific context given in the question, and then weighing the merits or drawbacks in an evaluation.</p> <p>Few candidates could develop the concept of modularity to the level of interfaces and the need for agreed standards for function calls and return values. Most responses focused on the basics of the use of functions and procedures but did not demonstrate how different modules might be integrated.</p>
			<b>Total</b>	<b>19</b>	
23			<p>1 mark per box up to a maximum of 4 marks:</p> <pre> graph TD     FR[Function rooms] --&gt; CA[Check availability]     FR --&gt; CR[Choose room]     FR --&gt; MR[Make reservation]     FR --&gt; CP[Check payment]     CA --&gt; ED[Enter date]     CA --&gt; DARR[Display available rooms]     CP --&gt; CPD[Check payment details]     CP --&gt; SCE[Send confirmation email] </pre>	<p>4 (AO2.2) (4)</p>	<p><b><u>Examiner's Comments</u></b></p> <p>Candidates needed to apply computational thinking skills to identify the components of the problem presented in the stem of the question and to determine the order of the steps required. Many candidates responded in context and generally answered well, but a significant number either repeated the points they made or repeated steps already given in the diagram.</p>
			<b>Total</b>	<b>4</b>	
24	a	i	<p>1 mark per box up to a maximum of 3 marks.</p> <ul style="list-style-type: none"> <li>• <b>Select puzzle</b> and <b>display blank grid</b> (below new game)</li> <li>• <b>Select box</b> and <b>change colour of boxes</b> (below play game)</li> </ul>	<p>12AO1.1 (2)AO1.2 (2)AO2.1 (23AO3.3 (5)</p>	

		<ul style="list-style-type: none"> <li>• <b>Compare to answer</b> and <b>display correct/incorrect</b> (below check answer)</li> </ul> <p>e.g.</p>  <pre> graph TD     NewGame[New game] --&gt; NewGameSub[New game]     NewGame --&gt; PlayGame[Play game]     NewGame --&gt; CheckAnswer[Check answer]     NewGameSub --&gt; SelectPuzzle[Select puzzle]     NewGameSub --&gt; DisplayStartGrid[Display start grid]     PlayGame --&gt; SelectBox[Select box]     PlayGame --&gt; ChangeColour[Change colour of box]     CheckAnswer --&gt; CompareToAnswer[Compare to answer]     CheckAnswer --&gt; DisplayCorrectIncorrect[Display correct/incorrect] </pre>		
	ii	<p>1 mark per bullet up to a maximum of 2 marks, e.g: e.g.</p> <ul style="list-style-type: none"> <li>• Splits the problem into smaller chunks</li> <li>• Smaller problems are more manageable</li> <li>• Smaller problems are easier to solve</li> <li>• To see where code can be reused in the solution</li> <li>• To split tasks between different programmers</li> </ul>	<p>2AO1.1 (1)AO1.2 (1)</p>	
	iii	<p>1 mark for input, 1 for process 1 for output e.g. Input:</p> <ul style="list-style-type: none"> <li>• Clicking a box</li> </ul> <p>Process:</p> <ul style="list-style-type: none"> <li>• Generating new puzzle</li> <li>• Checking if block is black</li> <li>• Changing block to white</li> </ul> <p>Output:</p>	<p>3AO2.2 (3)</p>	



			<ul style="list-style-type: none"> <li>Grid with coloured squares</li> </ul>		
		b i	<p>1 mark for each correctly completed statement up to a maximum of 5 marks:</p> <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 06             then 07                 count = count + 1 08             elseif count &gt;= 1 then 09                 output = output + 10                     str(count) + " " 11                 count = 0 12             endif 13         next i 14         if count &gt;= 1 then 15             output=output+str(count) 16         elseif output == "" then 17             output = "0" 18         endif 19     return output 20 endfunction </pre>	<p>5AO2.2 (2)AO3.2 (3)</p>	<p>Accept</p> <p>for i = 0 to row.length-1</p> <p>for i = 0 to row.length</p> <p>for i=0 to 5</p>

		ii	<p>1 mark per bullet up to a maximum of 2 marks, e.g:</p> <ul style="list-style-type: none"> <li>• Initialise the variable output...</li> <li>• ...with a space</li> <li>• ...for use later on in the code...</li> <li>• ...So it can be used for concatenation later in the code ...</li> <li>• ...to avoid an error being generated</li> </ul>	<p>2AO1.2 (1)AO2.2 (1)</p>	
		iii	<p>1 mark per bullet up to a maximum of 3 marks, e.g:</p> <ul style="list-style-type: none"> <li>• check the value stored in each index</li> <li>• check whether it is at the end of a row</li> <li>• check whether each row has been given an output or not</li> </ul>	<p>3AO2.2 (3)</p>	
		iv	<p>1 mark per bullet up to a maximum of 6 marks:</p> <ul style="list-style-type: none"> <li>• Procedure heading for displayRowAnswer</li> <li>• ...taking puzzle as parameters</li> <li>• Nested loops through all array elements</li> <li>• ...outputting all rows</li> <li>• ... at the end of each row calling countRow ....</li> <li>• .....with parameters puzzle and the current loop counter</li> </ul> <p>e.g.</p> <pre> procedure displayRowAnswer(puzzle)   for i = 0 To 4     for j = 0 To 4       print(puzzle[i, j] + " ")     next j   print (" " + countRow     (puzzle, i)) </pre>	<p>6AO2.2 (3)AO3.2 (3)</p>	<p>Accept</p> <p>for i = 0 to <b>row.length-1</b></p> <p>for i = 0 to <b>row.length</b></p> <p>for i=0 to <b>5</b></p>

			<pre> next i endprocedure </pre>		
		v	<p>1 mark for clearly identifying each error and giving the correction.</p> <ul style="list-style-type: none"> <li>Line 01 needs <code>answerGrid</code> as parameter</li> <li>Line 04 <code>==</code> should be <code>!=</code></li> <li>Line 08 should be <code>next row</code></li> </ul>	3AO2.1 (3)	<p>Do not award marks for line numbers alone without stating the error.</p> <p>Consider 1 mark for not changing line 04 but changing 05 to true and 09 to False</p>
		c	<p><b>Mark Band 3 – High level (7-9 marks)</b>  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 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><b>Mark Band 2 – Mid level (4-6 marks)</b>  The candidate demonstrates reasonable knowledge 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</p>	9AO1.1 (2)AO1.2 (2)AO2.1 (2)AO3.3 (3)	<p><b>AO1: Knowledge and Understanding</b>  <b>Indicative content</b>  Local variables:</p> <ul style="list-style-type: none"> <li>Scope within the module defined within</li> <li>Cannot access externally unless passed as parameter, or returned from function</li> <li>When module is exited, memory of variable is freed</li> </ul> <p>Global variables:</p> <ul style="list-style-type: none"> <li>Scope within the entire program</li> <li>Can access from anywhere</li> <li>Retained in memory permanently</li> </ul> <p>ByRef Points to location of variable  ByVal Sends the value</p> <p><b>AO2: Application</b></p> <ul style="list-style-type: none"> <li>If global the arrays can be accessed from all modules by direct reference</li> </ul>

		<p>conclusion showing some recognition of influencing factors that would determine whether it is possible to use local and global variables 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><b>Mark Band 1 – Low Level (1-3 marks)</b></p> <p>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.</p> <p>The candidate provides nothing more than an unsupported assertion.</p> <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><b>0 marks</b></p> <p>No attempt to answer the question or response is not worthy of credit.</p>		<ul style="list-style-type: none"> <li>• If local to the main, the arrays will need to be passed as parameters byreference</li> <li>• Can send ByVal – but not always possible with arrays in some languages</li> <li>• 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</li> </ul> <p><b>AO3: Evaluation</b></p> <p>e.g.</p> <ul style="list-style-type: none"> <li>• +ve Local = memory efficient</li> <li>• +ve Global = easier programming, simpler to follow, easier to debug</li> <li>• -ve Global = memory inefficient, not good programming technique</li> <li>• -ve Local = more difficult to trace/debug/follow where the values are passed</li> <li>• 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</li> </ul>
	d	<p>1 mark per bullet to max 4</p> <p>e.g.</p> <ul style="list-style-type: none"> <li>• Make use of random numbers</li> <li>• Generate an x/horizontal size for the grid</li> </ul>	<p>4AO2.1 (2)AO2.2 (2)</p>	

			<ul style="list-style-type: none"> <li>• Generate a y/vertical size for the grid</li> <li>• Loop through each row/column</li> <li>• ...generate a number between 0 and the number of rows/columns (depending on MP4 answer)</li> <li>• Loop through each box</li> <li>• ...generate a 1 or 0 to store in it</li> </ul>		
			<b>Total</b>	<b>40</b>	
25	a		<p>1 mark for the purpose and 1 mark for matching appropriate name (4 marks total), e.g:</p> <ul style="list-style-type: none"> <li>• Pointer to the first element in the queue</li> <li>• firstElement / any other meaningful name</li> <li>• Pointer to the last element in the queue / Pointer to the first free element in the queue</li> <li>• lastElement / any other meaningful name</li> </ul>	4AO1.2 (4)	Must cover purpose and name for 2 marks for each pointer.
	b		<p>1 mark per bullet up to a maximum of 5 marks, e.g:</p> <ul style="list-style-type: none"> <li>• Check if the queue is full</li> <li>• ... if the firstElement pointer (+1) = lastElement / length variable == queue's capacity</li> <li>• ... if it is return False</li> <li>• Adds element at lastElement (+1) position / Adds element at startPosition+length</li> <li>• ... increments lastElement pointer</li> <li>• If lastElement is greater than last Index / pointer becomes pointer MOD array.size</li> <li>• ...reset to 0</li> </ul>	5AO1.2 (5)	Look out for variations in representing the queue

			Total	9	
26			<p><b>Mark Band 3 – High level (7-9 marks)</b>  The candidate demonstrates a thorough knowledge and understanding of IDEs; 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 IDEs 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><b>Mark Band 2 – Mid level (4-6 marks)</b>  The candidate demonstrates reasonable knowledge and understanding of IDEs; 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 IDEs 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><b>Mark Band 1 – Low Level (1-3 marks)</b></p>	<p>9AO1.1  (2)AO1.2  (2)AO2.1  (2)AO3.3  (3)</p>	<p><b>AO1: Knowledge and Understanding</b>  <b>Indicative content e.g.</b></p> <ul style="list-style-type: none"> <li>• IDE is software that includes an editor, compiler, run-time environment</li> </ul> <p>Creating</p> <ul style="list-style-type: none"> <li>• Autocorrect</li> <li>• Autocomplete</li> <li>• Pretty printing</li> </ul> <p>Testing</p> <ul style="list-style-type: none"> <li>• Breakpoints</li> <li>• Stepping</li> <li>• Variable watch window</li> </ul> <p><b>AO2: Application</b>  e.g.</p> <ul style="list-style-type: none"> <li>• Tell you when you make a syntax error</li> <li>• Allows you to write and run the code in one piece of software</li> <li>• Suggests code so you don't have to remember code, or autocorrect spelling mistakes</li> <li>• Helps you trace the program so you can see what happens when values change</li> </ul>

		<p>The candidate demonstrates a basic knowledge of IDEs 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> <p><i>The information is basic and communicated in an unstructured way.</i></p> <p><i>The information is supported by limited evidence and the relationship to the evidence may not be clear.</i></p> <p><b>0 marks</b></p> <p>No attempt to answer the question or response is not worthy of credit.</p>		<p>without having to manually insert print statements etc.</p> <ul style="list-style-type: none"> <li>Autogenerate boilerplate code.</li> </ul> <p><b>AO3: Evaluation</b></p> <p>Candidates will need to evaluate the benefits and drawbacks of using IDEs.</p> <p>e.g.</p> <ul style="list-style-type: none"> <li>Reduce errors through autocorrect and suggestions</li> <li>Reduce time to write the program because features help you spot errors before running the code, some errors will be corrected so you don't have to</li> <li>Write and test in one environment so you don't have to close and run elsewhere, then re-open etc.</li> </ul>
		<b>Total</b>	<b>9</b>	
27	i	<p><b>1 mark for each number/statement up to a maximum of 6 marks:</b></p> <pre>function binarySearch(dataArray:byref, upperbound, lowerbound, <b>searchValue</b>)     while true         middle = lowerbound +             ((upperbound - lowerbound) <b>DIV 2</b>)         if upperbound &lt; lowerbound</pre>	<p>6AO1.2 (2)AO3.3 (4)</p>	

			<pre> then     return <b>-1</b> else     if dataArray[middle] &lt;         searchValue then         lowerbound = <b>middle + 1</b>     elseif dataArray[middle] &gt;         searchValue then         upperbound = <b>middle - 1</b>     else         return <b>middle</b>     endif endif endwhile endfunction </pre>		
		ii	Do...until / repeat...until / post condition	1AO1.2 (1)	
			<b>Total</b>	<b>7</b>	
28	a		<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>• Calculation of result to 3</li> <li>• Call with <code>thisFunction(theArray, num1=4, num2=7, num3=35)</code></li> <li>• Result = 5</li> <li>• call with <code>thisFunction(theArray, num1=6, num2=7, num3=35)</code></li> <li>• (Result = 6) return of value 6</li> </ul>	<p>5 AO2.1 (3) AO2.2 (2)</p>	



			<table><tr><th>Function call</th><th>num1</th><th>num2</th><th>num3</th><th>result</th></tr><tr><td>thisFunction (theArray,0,7,35)</td><td>0</td><td>7</td><td>35</td><td>3</td></tr><tr><td>thisFunction (theArray,4,7,35)</td><td>4</td><td>7</td><td>35</td><td>5</td></tr><tr><td>thisFunction (thisArray,6,7,35)</td><td>6</td><td>7</td><td>35</td><td>6</td></tr></table>	Function call	num1	num2	num3	result	thisFunction (theArray,0,7,35)	0	7	35	3	thisFunction (theArray,4,7,35)	4	7	35	5	thisFunction (thisArray,6,7,35)	6	7	35	6		
Function call	num1	num2	num3	result																					
thisFunction (theArray,0,7,35)	0	7	35	3																					
thisFunction (theArray,4,7,35)	4	7	35	5																					
thisFunction (thisArray,6,7,35)	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"><li>• Recursion uses more memory...</li><li>• ...iteration uses less memory</li><li>• Recursion declares new variables /variables are put onto the stack each time...</li><li>• ...iteration reuses the same variables</li><li>• Recursive can run out of memory/stack space...</li><li>• ...while iteration cannot run out of memory</li><li>• Recursion can express a problem more elegantly / in fewer lines of code...</li><li>• ...while iteration can take more lines of code / be harder to understand</li><li>• Recursion will be self-referential / will call itself...</li><li>• ... whereas iteration does not</li></ul>	4 AO1.1 (2) AO1.2 (2)																					
	d		1 mark per bullet to max 6 <ul style="list-style-type: none"><li>• Retains function call</li></ul>	6 AO2.2 (3)																					

		<ul style="list-style-type: none"> <li>• Uses a loop</li> <li>• ...that will loop until all elements inspected or value found</li> <li>• Updates num1 appropriately</li> <li>• Updates num2 appropriately</li> <li>• Returns -1 in the correct place if the value has not been found</li> <li>• Returns the result in the correct place if the value has been found</li> </ul> <p>e.g.</p> <pre>function thisFunction(theArray, num1, num2, num3)  while (true)     result = num1 + ((num2 - num1) DIV 2)     if num2 &lt; num1 then         return -1     else         if theArray[result] &lt; num3 then             num1 = result + 1         elseif theArray[result] &gt; num3 then             num2 = result - 1         else             return result         endif     endif endwhile endfunction</pre>	AO3.1 (3)	
		<b>Total</b>	<b>16</b>	
29		1 mark per bullet	2 AO1.2 (1)	

			<ul style="list-style-type: none"> <li>• 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</li> <li>• By value would create a copy and not change the original / when control returns to the main program the array will <b>not</b> be sorted</li> <li>• By value the array is local to the function.</li> <li>• By reference will use less memory</li> </ul>	AO2.2 (1)	
			<b>Total</b>	<b>2</b>	
30			<p>1 mark per bullet up to a maximum of 4 marks:</p> <ul style="list-style-type: none"> <li>• The inner loop will compare all of the adjacent items....</li> <li>• ....in a single pass</li> <li>• The outer loop will repeat the process of checking adjacent items....</li> <li>• ...until all passes are complete / the items are sorted/no swaps have been made in a pass</li> </ul>	4 AO1.2 (4)	Allow other valid interpretations e.g. conditional while loop ... used to compare against swap flag after each pass; counter controlled for loop ... used to check adjacent items on each pass
			<b>Total</b>	<b>4</b>	
31	a	i	<p>1 mark for identifying error and correction (identification may be implicit)</p> <ul style="list-style-type: none"> <li>• Line 02 tempPointer should become headPointer, not -1 tempPointer = headPointer</li> <li>• Line 05 message should say it's empty not full print("List is empty")</li> </ul>	3 AO2.1 (2) AO2.2 (2)	

		<ul style="list-style-type: none"> <li>Line 07 pointer should be tempPointer while  <code>linkedList[tempPointer].getPointer() != -1</code></li> <li>Line 08 Incorrect call to node pointer <code>dataToPrint = dataToPrint + " " + linkedList[tempPointer].getData()</code></li> <li>Line 09 assignment is wrong way <code>tempPointer = linkedList[tempPointer].getPointer()</code></li> <li>Line 11 missing final parenthesis <code>print(dataToPrint + " " + linkedList[tempPointer].getData())</code></li> </ul>		Do not award marks for stating the line number without a valid correction.
	ii	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>Stepping Through The Code...</li> <li>...to run one line at a time to see where the error is</li> <li>Syntax Error Highlighting....</li> <li>....to distinguish syntax errors in the program code</li> <li>Setting breakpoints...</li> <li>....to debug individual sections of code at a time</li> <li>Variable watch window...</li> <li>...To check that the variables are being updated corrected</li> </ul>	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>
	b	<p><b>Mark Band 3 – High level (9-12 marks)</b></p> <p>The candidate demonstrates a thorough knowledge and understanding of the object oriented 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.</p> <p>Evidence/examples will be explicitly relevant to the explanation. The candidate is able to weigh up the use of all of the object oriented techniques which results in a supported and</p>	<p>12</p> <p>AO1.1 (3)</p> <p>AO1.2 (3)</p> <p>AO2.1 (3)</p> <p>AO3.3 (3)</p>	<p><b>AO1: Knowledge and Understanding</b></p> <p><b>Indicative content</b></p> <ul style="list-style-type: none"> <li>Classes, this a template. It will define what attributes and methods an object should have.</li> <li>Objects, when you create an instance of a class. Each object that is instantiated from</li> </ul>

		<p>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><b>Mark Band 2 – Mid level (5-8 marks)</b>  The candidate demonstrates reasonable knowledge and understanding of the object oriented 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.  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 oriented technique 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><b>Mark Band 1 – Low Level (1-4 marks)</b>  The candidate demonstrates a basic knowledge of the object oriented techniques 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.  <i>The information is basic and communicated in an unstructured way.</i></p>		<p>the same class will share the same attributes and methods.</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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</li> <li>• Encapsulation, this protects attributes of an object by making them private so that they can't be accessed or altered accidentally by other objects.</li> </ul> <p><b>AO2: Application</b></p> <ul style="list-style-type: none"> <li>• A class can be used to declare the attributes and methods for the linked list. These will initialise the nodes and join them.</li> <li>• 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.</li> <li>• 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.</li> </ul>
--	--	---	--	--

		<p><i>The information is supported by limited evidence and the relationship to the evidence may not be clear.</i></p> <p><b>0 marks</b> No attempt to answer the question or response is not worthy of credit.</p>		<ul style="list-style-type: none"> <li>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.</li> </ul> <p><b>AO3: Evaluation</b></p> <ul style="list-style-type: none"> <li>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.</li> <li>OOP also allows programs to be easier to modify and maintain.</li> </ul>
		<b>Total</b>	<b>21</b>	
32	a	<p>1 mark per pointer</p> <ul style="list-style-type: none"> <li>queueHead: Point to the first element in the queue / next element to remove</li> <li>queueTail: Point to the last element in the queue</li> </ul>	<p>2 AO1.2 (2)</p>	
	b	<p>1 mark per bullet up to max 5</p> <ul style="list-style-type: none"> <li>first 3 jobs removed</li> <li>128 and 129 added in positions 4 and 5 respectively</li> <li>no additional jobs</li> <li>queueHead being 3 (FT errors)</li> <li>queueTail being 5 (FT errors)</li> </ul>	<p>5 AO2.1 (2) AO2.2 (3)</p>	<p>The underlying implementation of the queue has</p>


					<p>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"> <li>• Function declaration</li> <li>• Checking if queue is empty</li> <li>• ...returning null</li> <li>• (Otherwise) incrementing queueHead</li> <li>• ...returning buffer[queueHead-1]</li> </ul> <p>e.g.</p> <pre>function dequeue()   if queueHead &gt; 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"> <li>• Function declaration taking parameter</li> <li>• Checking if queue is full</li> <li>• ...returning -1</li> <li>• (Otherwise) incrementing queueTail</li> </ul>	<p>6 AO2.2 (3) AO3.3 (3)</p>	

		<ul style="list-style-type: none"> <li>• Adding newJob to buffer(queueTail)</li> <li>• Returning 1</li> </ul> <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"> <li>• Inputting user choice</li> <li>• If enqueue chosen input job name</li> <li>• ...call enqueue with input value as parameter</li> <li>• ...check if return value is -1 and output full</li> <li>• ...otherwise output message that item is added</li> <li>• If dequeue chosen</li> <li>• ...call dequeue <b>and</b> save returned value</li> <li>• ...output returned value (jobname) if not null</li> <li>• ...or output queue is empty</li> </ul> <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</pre>	<p>8 AO2.2 (2) AO3.3 (6)</p>	<p>Allow equivalent checks / logic</p>



			<pre>         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>		
	d		<p>1 mark per bullet to 3</p> <ul style="list-style-type: none"> <li>• Check if either head or tail are incremented to above 99</li> <li>• ... set to be 0 instead</li> <li>• When checking if array is full check if (queueTail == queueHead – 1) OR (queueTail==99 AND queueHead==0)</li> </ul>	<p>3 AO2.1 (1) AO2.2 (2)</p>	Credit equivalent modulo arithmetic solution
	e		<p>1 mark per bullet to max 3, e.g.</p> <ul style="list-style-type: none"> <li>• Use a different structure e.g. a linked list</li> <li>• ...items can be added at different points in the linked list depending on priority</li> <li>• ...by changing the pointers to items needing priority</li> <li>• Have different queues for different priorities</li> <li>• ...add the job to the queue relevant to its priority</li> <li>• ...print all the jobs in the highest priority queue first</li> </ul>	<p>3 AO2.1 (2) AO2.1 (1)</p>	Allow other suitable descriptions that show how the program could be amended.

			Total	32																																		
33	a	i	1 mark per bullet <ul style="list-style-type: none"><li>Points to where the next/first free node is</li><li>To add data into the linked listed.</li></ul>	2 AO1.2 (1) AO2.2 (1)																																		
		ii	Points to the first element in the linked list	1 AO1.2 (1)																																		
		iii	1 mark per bullet <ul style="list-style-type: none"><li>No change made to nodes/pointers unaffected by this removal</li><li>Index 0 points to 2 instead of 3</li><li>Node 9 points to 3 instead of -1 / Node freeListPointer points to 3 instead of 4</li><li>Node 3 points to 4 / -1 (must match MP3</li></ul> <div>Solution:<div><div>headPointer 0 freeListPointer 4</div><table><thead><tr><th>index</th><th>data</th><th>pointer</th></tr></thead><tbody><tr><td>0</td><td>2.4</td><td>2</td></tr><tr><td>1</td><td>3.5</td><td>-1</td></tr><tr><td>2</td><td>1.8</td><td>1</td></tr><tr><td>3</td><td>6.9</td><td>-1</td></tr><tr><td>4</td><td></td><td>5</td></tr><tr><td>5</td><td></td><td>6</td></tr><tr><td>6</td><td></td><td>7</td></tr><tr><td>7</td><td></td><td>8</td></tr><tr><td>8</td><td></td><td>9</td></tr><tr><td>9</td><td></td><td>3</td></tr></tbody></table></div><div>Alternative Solution:</div></div>	index	data	pointer	0	2.4	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	4 AO1.2 (1) AO2.2 (1)	6.9 3 may or may not be written by candidates, both are acceptable.  Candidates may add the node freed up (node 3) to the start or the end of the free storage. Award marks for both approaches.
index	data	pointer																																				
0	2.4	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																																				

					
	b	i	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>Class declaration and all code is nested within the class</li> <li>Two private identifiers data and pointer (with suitable data types if given)</li> <li>Public constructor heading as a procedure (public may be implied but cannot be private) taking both parameters as given in table</li> <li>Assigns parameters to the attributes</li> </ul> <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 AO2.2 (1) AO3.3 (3)</p>	<p>Accept</p> <p>public node(newData, newPointer) <i>(may also have data stypes for parameters e.g. int newData)</i></p> <p>Accept:</p> <p>this.data = newData this.pointer = newPointer</p> <p>or similar</p>
		ii	<p>1 mark per bullet to max 2</p> <ul style="list-style-type: none"> <li>A get method allows the attribute to be accessed / returned</li> <li>A set method allows the attribute to be changed (with parameters)</li> </ul>	<p>2 AO2.2 (2)</p>	

		c	<p>1 mark per bullet to max 6</p> <ul style="list-style-type: none"> <li>• Initialise message string</li> <li>• Start with the headPointer</li> <li>• Check if the headPointer is null</li> <li>• ...return that the list is empty</li> <li>• Check the pointer of the node at headPointer</li> <li>• If it is not null/-1/the last element</li> <li>• loop through all the linkedList elements</li> <li>• ...concatenate the pointer to the message</li> <li>• ...replacing the pointer with the current node's pointer each time</li> <li>• ...if the data is found concatenate the pointer and "found" to the message <b>and</b> return it</li> <li>• ...if the loop ends and the data item is not found, concatenate "not found" to the message <b>and</b> return it</li> </ul>	<p>6 AO1.2 (2) AO2.1 (2) AO2.2 (2)</p>	
			<b>Total</b>	<b>19</b>	
34		i	<p>1 mark per bullet up to a maximum of 3 marks:</p> <ul style="list-style-type: none"> <li>• 5 (Jimmy)</li> <li>• 8 (Siad)</li> <li>• 9 (Tommy)</li> </ul>	<p>3 AO2.1 (3)</p>	
		ii	<p>1 mark per bullet up to a maximum of 2 marks, e.g:</p> <ul style="list-style-type: none"> <li>• Efficient</li> <li>• ...as does not need to search every single element/uses divide and conquer</li> </ul>	<p>2 AO1.2 (2)</p>	

		iii	<ul style="list-style-type: none"> <li>Linear Search / Serial Search</li> </ul>	1 A02.1 (1)	
		iv	<ul style="list-style-type: none"> <li>The items are in alphabetical order / the items are sorted</li> </ul>	1 A02.1 (1)	
			<b>Total</b>	<b>7</b>	
35			<p>1 mark per bullet up to a maximum of 9 marks:</p> <ul style="list-style-type: none"> <li>Defining the <code>createUsername</code> procedure correctly</li> <li>Suitable logic for inputting the first name</li> <li>Suitable logic for inputting the surname</li> <li>Suitable logic for using the first letter of the first name</li> <li>Suitable logic for joining the different sections of the username together</li> <li>Suitable logic to pass the username into the function <code>existingUsers</code> (eg as a parameter or global variable)</li> <li>Suitable logic for continually increasing the number by 1 .....</li> <li>...until the username is unique</li> <li>Sensible use of variable names and indentation throughout</li> </ul>	9 A03.1 (3) A03.2 (6)	<p><b>Example solution:</b></p> <pre> procedure createUsername()     firstname = input("Enter First Name")     surname = input("Enter Surname")     number = 0     while unique == false         number = number + 1         username = surname +         firstname.substring(0,1) +         str(number)         unique =         existingUsers(username)     endwhile     print("Username is unique") endprocedure </pre> <p>There are many different ways that this procedure could have been achieved. Therefore other alternative methods should be given credit</p>

					(candidates may use <code>substring</code> or <code>mid</code> to access first character of <code>firstname</code> ).
			<b>Total</b>	<b>9</b>	
36	a	i	<ul style="list-style-type: none"> <li>Selection/branching</li> </ul>	1  A03.3 (1)	
		ii	1 mark per bullet up to a maximum of 3 marks, e.g: <ul style="list-style-type: none"> <li>The number of user attempts is not known</li> <li>The code will need to continue until the password entered is correct</li> <li>A while loop will keep repeating until the correct password has been input / condition is met</li> <li>A for loop will only repeat a certain number of times</li> <li>A for loop may continually ask for password even though it's been entered correctly</li> </ul>	4  A02.1 (4)	
		iii	1 mark per bullet up to a maximum of 3 marks, e.g: <ul style="list-style-type: none"> <li>Correct use of <code>do</code> at the start of the loop.</li> <li>Correct use of <code>until</code> at the end of the loop.</li> <li>Correct logic for inputting password, checking the entered password and for setting <code>check</code> to <code>true</code>/checking the password within the condition of the loop.</li> </ul>	3  A03.2 (3)	<b>Example Solution</b> <pre>do     enteredPassword=input("Enter Password")     if enteredPassword == correctPassword then         check = true     endif until check == true</pre> <b>Alternative solution</b>

					<pre>do     enteredPassword=input("Enter Password") until enteredPassword==correctPassword</pre>
	b	i	<p>1 mark for identifying a feature and 1 mark for stating how it can be used up to a maximum of 2 marks for each IDE feature (6 marks maximum in total.)</p> <p>For example:</p> <ul style="list-style-type: none"> <li>Autocomplete (1) which will predict variable / built-in functions (1)</li> <li>Auto indent (1) to automatically indent code when selection / iterative statements are used (1)</li> <li>Colour coding (1) to be able to distinguish between the different parts of each statement/line (1)</li> </ul>	<p>6</p> <p>A01.1 (3)</p> <p>A01.2 (3)</p>	<p>Allow other suitable responses that are appropriate to <b>writing</b> programming code such as automatic syntax analysis, automatic cross-referencing, line numbers and code comments, automated refactoring, automated code generation (e.g. creating templates for common patterns).</p>
		ii	<p>1 mark per bullet up to a maximum of 2 marks for each benefit(4 marks maximum in total), e.g:</p> <ul style="list-style-type: none"> <li>Reduced development time (1) due to time boxing / each subtask being given a strict time limit (1)</li> <li>Increased user involvement (1) so issues can be identified and fixed early / more likely to meet client requirements (1)</li> <li>The requirements do not all need to be stated at the start (1) so therefore it is more flexible (1)</li> </ul>	<p>4</p> <p>A01.1 (2)</p> <p>A01.2 (2)</p>	<p>Allow other suitable responses.</p>
		iii	<p>1 mark per bullet up to a maximum of 2 marks, e.g:</p>	<p>2</p>	<p>Response must cover both black box and white box testing for 2 marks.</p>

			<ul style="list-style-type: none"><li>• Black box is when the internal structure/ design is not known (to the tester)</li><li>• Black box requires limited/no programming knowledge</li><li>• White box is when the internal structure/ design is known (to the tester)</li><li>• White box requires programming knowledge</li></ul>	A01.2 (2)	
			<b>Total</b>	<b>20</b>	