GCST

2.2.1 PROGRAMMING TECHNIQUES

TOPIC WISE EXAM QUESTIONS

ANSWERS

A-LEVEL OCR

GCSECOMPUTERSCIENCETUTOR.COM

<table>
<tr><td>4</td><td>(a)</td><td>(i)</td><td>1 mark for:<ul><li><code>isInteger</code></li><li><code>number</code></li><li><code>result</code></li><li><code>count</code></li><li><code>asciiValue</code></li></ul></td><td>1</td><td>Penalise excessive spaces in identifiers such as
<i>ascii   Value</i>
instead of <i>asciiValue</i></td></tr>
<tr><td>4</td><td>(a)</td><td>(ii)</td><td>(0)5</td><td>1</td><td></td></tr>
<tr><td>4</td><td>(a)</td><td>(iii)</td><td>(0)3</td><td>1</td><td></td></tr>
<tr><td>4</td><td>(b)</td><td></td><td>1 mark each

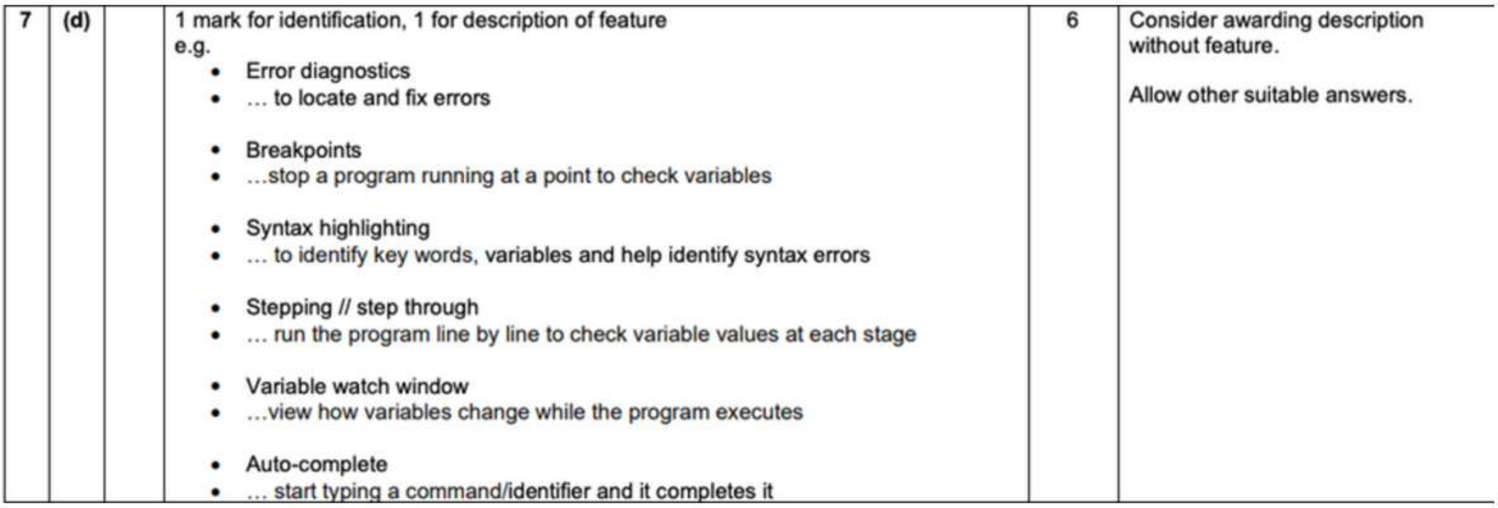
03<ul><li>Loop through each of the <b>characters/digits</b> in the <code>number</code> string (parameter)</li></ul>04<ul><li>Find the ASCII value of the current <b>character/digit</b></li></ul>09<ul><li>Return true if the value is an integer and false otherwise</li></ul></td><td>3</td><td></td></tr>
<tr><td>5</td><td>(a)</td><td></td><td>1 mark each to max 3<ul><li>The function calls itself….</li><li>…..such as line <code>05</code> / <code>07</code></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 <code>02</code></li><li>There may be more than one base case</li></ul></td><td>3</td><td>Allow answers in context as long as they are clear what the features are.</td></tr>
<tr><td>5</td><td>(b)</td><td></td><td>1 mark for final return value 29 (award in working or answer space)
1 mark each for working<ul><li>First call with 10 and second call with 7</li><li>Remainder of calls 6, 3, 2</li><li>Final call value -1</li><li>Adding/showing return values (1 + 2 + 3 + 6 + 7 + 10)</li></ul>e.g.

<table>
<tr><th>Function call</th><th><code>value</code></th><th><code>return</code></th></tr>
<tr><td><code>recursiveAlgorithm(10)</code></td><td><code>10</code></td><td><code>29</code></td></tr>
<tr><td><code>recursiveAlgorithm(7)</code></td><td><code>7</code></td><td><code>19</code></td></tr>
<tr><td><code>recursiveAlgorithm(6)</code></td><td><code>6</code></td><td><code>12</code></td></tr>
<tr><td><code>recursiveAlgorithm(3)</code></td><td><code>3</code></td><td><code>6</code></td></tr>
<tr><td><code>recursiveAlgorithm(2)</code></td><td><code>2</code></td><td><code>3</code></td></tr>
<tr><td><code>recursiveAlgorithm(-1)</code></td><td><code>-1</code></td><td><code>1</code></td></tr>
<tr><td></td><td></td><td></td></tr>
</table></td><td>5</td><td>The table is given as guidance, but actual process may be presented in different ways.</td></tr>
<tr><td>6</td><td></td><td></td><td>1 mark each to max 6<ul><li>Taking number as input</li><li>Calculating remainder after division by 8</li><li>Calculating integer after division by 8</li><li>Correct loop until 0 is reached (or equivalent method)</li><li>Concatenating each remainder // storing each remainder in an array/list</li><li>Outputting the correct result</li></ul>e.g. pseudocode<pre>
number = input("Enter a number")

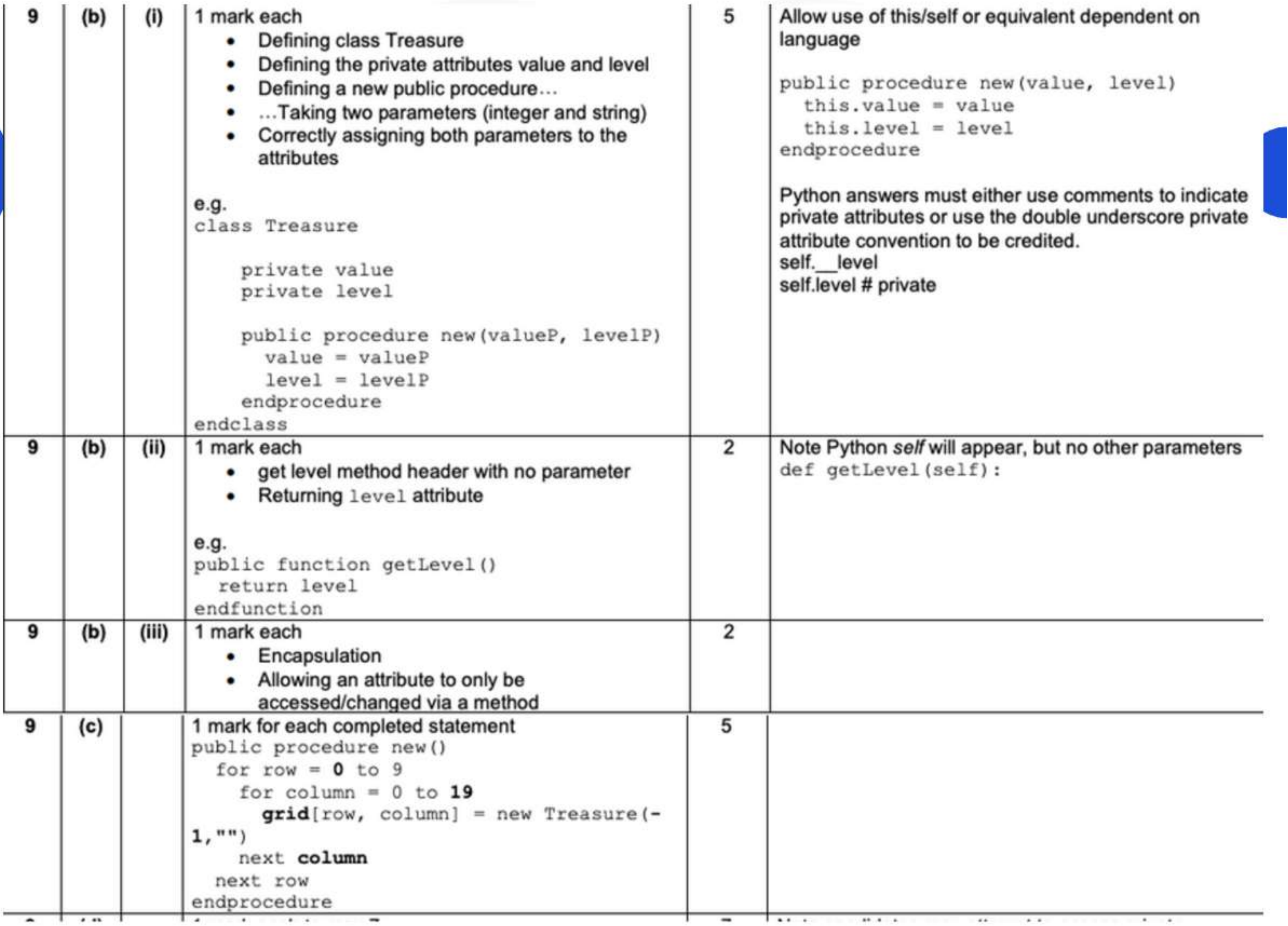
endResult = ""

while number != 0
  remainder = number MOD 8
  number = number DIV 8
  endResult = str(remainder) + str(endResult)
endwhile
print endResult
</pre></td><td>6</td><td>Note candidates can reverse the string before output if they don't concatenate in the order given in the example.
E.g.
<code>endResult = str(endResult) + str(remainder)</code>

The final markpoint can only be awarded where the correct output will be produced by the algorithm.</td></tr>
</table>



7	(d)		1 mark for identification, 1 for description of feature e.g. • Error diagnostics • … to locate and fix errors • Breakpoints • …stop a program running at a point to check variables • Syntax highlighting • … to identify key words, variables and help identify syntax errors • Stepping // step through • … run the program line by line to check variable values at each stage • Variable watch window • …view how variables change while the program executes • Auto-complete • … start typing a command/identifier and it completes it	6	Consider awarding description without feature. Allow other suitable answers.



9	(b)	(i)	1 mark each
			• Defining class Treasure
			• Defining the private attributes value and level
			• Defining a new public procedure…
			• …Taking two parameters (integer and string)
			• Correctly assigning both parameters to the attributes
			
			e.g.
			`class Treasure`
			
			`    private value`
			`    private level`
			
			`    public procedure new(valueP, levelP)`
			`      value = valueP`
			`      level = levelP`
			`    endprocedure`
			`endclass`
				5	Allow use of this/self or equivalent dependent on language
					
					`public procedure new(value, level)`
					`  this.value = value`
					`  this.level = level`
					`endprocedure`
					
					Python answers must either use comments to indicate private attributes or use the double underscore private attribute convention to be credited.
					self.__level
					self.level # private

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

9	(b)	(iii)	1 mark each • Encapsulation • Allowing an attribute to only be accessed/changed via a method	2	

9	(c)		1 mark for each completed statement
			`public procedure new()`
			`  for row = `**`0`**` to 9`
			`    for column = 0 to `**`19`**
			`      `**`grid`**`[row, column] = new Treasure(-`
			`1``,"")`
			`    next `**`column`**
			`  next row`
			`endprocedure`
				5	

<table>
<tr><td>9</td><td>(d)</td><td></td><td>1 mark each to max 7
• Procedure declaration taking parameter
• Taking two inputs for row and column from the user
• Accessing item at grid position…
• …using correct get methods <code>getGridItem</code>
• Checking (treasure) object's level/value…
• …using correct get method <code>getLevel</code> <code>getValue</code>
• …outputting "No treasure" if empty
• …otherwise outputting value and level

e.g.<pre>
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
</pre></td><td>7</td><td>Note candidates may attempt to access private attributes directly gameboard.grid(x,y) for example, instead of gameboard.getGridItem(x,y).

Credit cannot be given for the dependent second mark using appropriate get method if they do this, but FT marks can be awarded for later points if a reasonable attempt has been made.</td></tr>
<tr><td>9</td><td>(e)</td><td></td><td>1 mark each to max 4 e.g.
• Code can easily be reused…
• …classes can be used in other programs
• …inheritance can be to extend upon existing classes
• …as a class can be based on an existing class
• Easier to maintain….
• ….as classes can be modified or extended
• …debugging can be easier as encapsulation limits how attibutes are changed.
• Code can be more secure…
• … as access to attributes can be restricted to being via methods.
• Better for coding as part of a team…
• …as classes can be distributed between team members.</td><td>4</td><td>1 mark per benefit identified and 1 mark per expansion. Max 2 benefits and 1 expansion per benefit.</td></tr>
<tr><td>9</td><td>(f)</td><td></td><td><b>Mark Band 3 – High level</b>
<b>(7-9 marks)</b>
The candidate demonstrates a thorough knowledge and understanding of parameters and local/global variables; the material is generally accurate and detailed.
The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation.
<i>There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.</i>

<b>Mark Band 2 – Mid level</b>
<b>(4-6 marks)</b>
The candidate demonstrates reasonable knowledge and understanding of parameters and local/global variables; the material is generally accurate but at times underdeveloped.
The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed.
Evidence/examples are for the most part implicitly relevant to the explanation.
The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed.
<i>There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.</i>

<b>Mark Band 1 – Low Level</b>
<b>(1-3 marks)</b>
The candidate demonstrates a basic knowledge of parameters and local/global variables with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a</td><td>9</td><td><b>AO1: Knowledge and Understanding</b>
<b>Indicative content</b>
• Local variable can only be accessed within sub-program/main program it is declared within
• Global variable can be accessed by all sub-programs
• Parameters are items passed to a subproblem
• Passing by reference sends a pointer to the original value, so this will be changed when control is returned
• Passing by value sends the a copy of the value, so the original will not be changed when control is returned

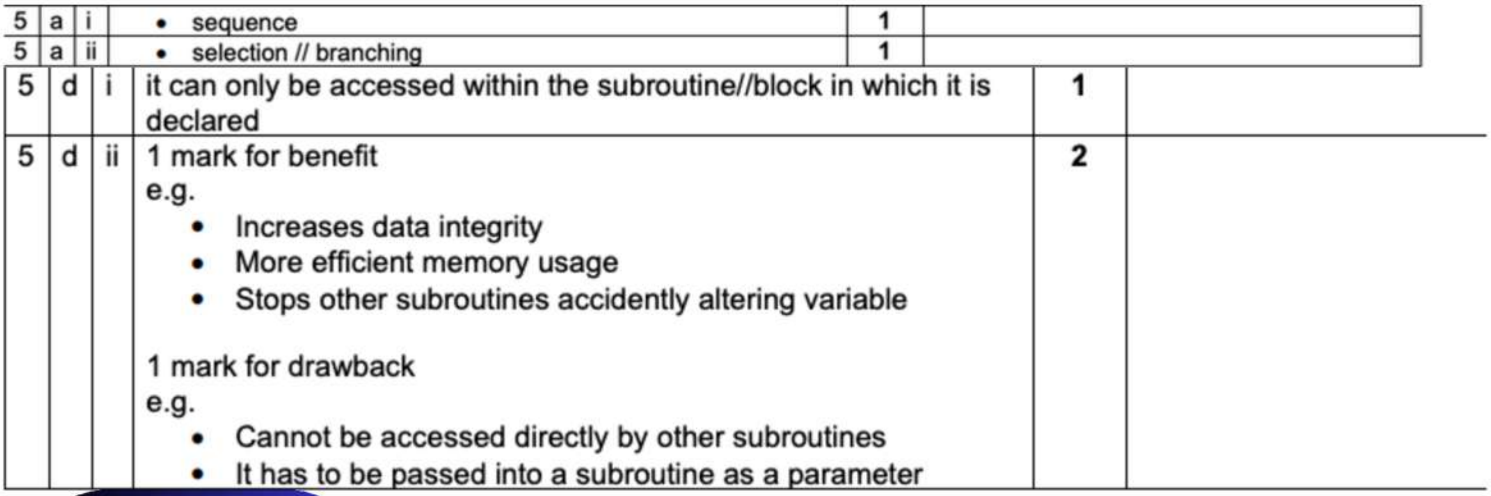
<b>AO2: Application</b>
• If board is local it can only be accessed in the main program
• This will need to be passed to any sub-programs that need to use it
• If the board needs to be changed it will need passing by reference, so that the board is updated
• If it only needs to be accessed and not changed it can be passed by value

<b>AO3: Evaluation</b>
• If global then this would be present in memory throughout hence using more memory
• …however the board will be required throughout the program so may be as efficient as passing it through parameters
• …if global then the programming may be more straight forward, and less likely to have errors with passing the board incorrectly to subprograms, i.e. it may not be updated when it needs to be
• Using local means that the board can be manipulated by subprograms without affecting the</td></tr>
</table>

3			**Mark Band 3–High Level** **(7-9 marks)** The candidate demonstrates 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 provides a thorough discussion which is well-balanced. Evaluative comments are consistently relevant and well-considered. There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated. **Mark Band 2-Mid Level** **(4-6 marks)** The candidate demonstrates reasonable knowledge and understanding of 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 provides a reasonable discussion, the majority of which is focused. Evaluative comments are for the most part appropriate, although one or two opportunities for development are missed. There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.	9	**AO1: Knowledge and Understanding** e.g. IDE: • pretty print / syntax highlighting • auto-complete • auto-correction • breakpoints • stepping Editor: • no helpful writing/debugging features • no excess features/interface **AO2.1: Application** e.g. IDE • identify syntax errors as writing • …saves trying to find them • easier debugging as can step through a program • auto-indenting avoids errors from incorrect indentation • May have built in unit testing to automate testing and avoid new errors being introduced. Editor • does not offer suggestions on code corrections • Has a lower footprint on memory/CPU which may be suited to quick alterations or working on lowed spec'd systems. • 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. **AO3.3: Evaluation** e.g. • IDE is helpful in reducing original errors • IDE is helpful in finding and correcting errors

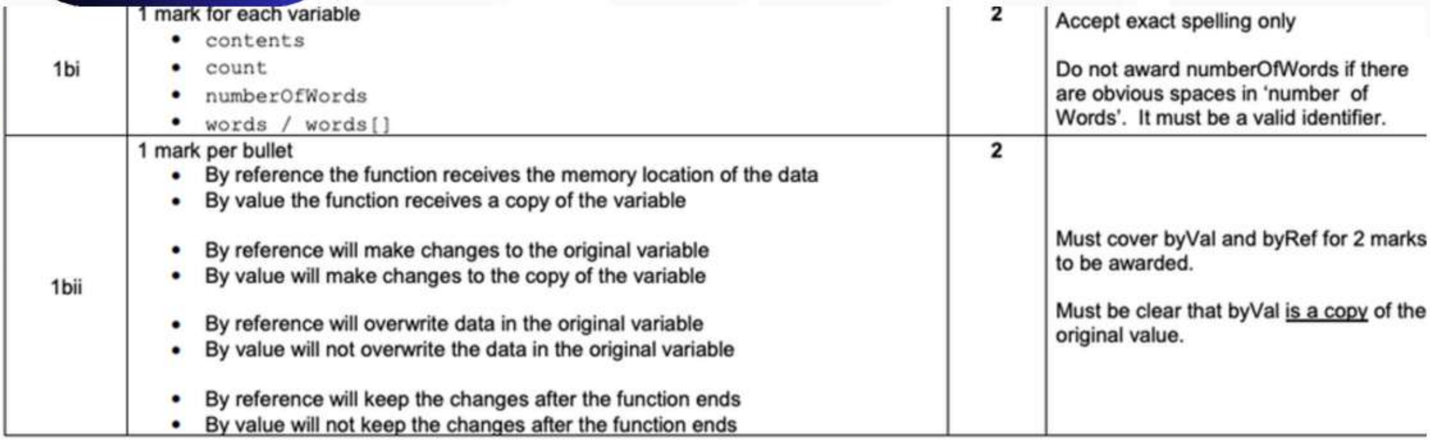
4	a		1 mark per bullet to max 6
			• function header taking parameter
			• looping appropriately e.g. until value is 0
			• dividing by 2 and finding remainder e.g. MOD
			• adding 1 or 0 correctly
			• …appending to a value to be returned // final string reversed
			• reducing value to use within loop
			• returning calculated value
			
			e.g.
			`function toBinary(denary)`
			`  binaryValue=""`
			`  while denary > 0`
			`    temp = denary MOD 2`
			`    if temp == 1 then`
			`      binaryValue = "1" + binaryValue`
			`    else`
			`      binaryValue = "0" + binaryValue`
			`    endif`
			`  denary = denary DIV 2`
			`  endwhile`
			`  return binaryValue`
			`endfunction`
				6	Award a recursive algorithm as equivalent

4	b		1 mark per bullet to max 4
			• taking value as input
			• looping until valid between 1 and 255
			• calling function with correct parameter
			• outputting return value
			
			`denary = -1`
			`while denary < 1 or denary > 255`
			`  denary = input("Enter denary value between 1 and 255")`
			`endwhile`
			`print(toBinary(denary))`
				4	Allow other checks for a valid number. For example
					
					`denary.isInteger == False`

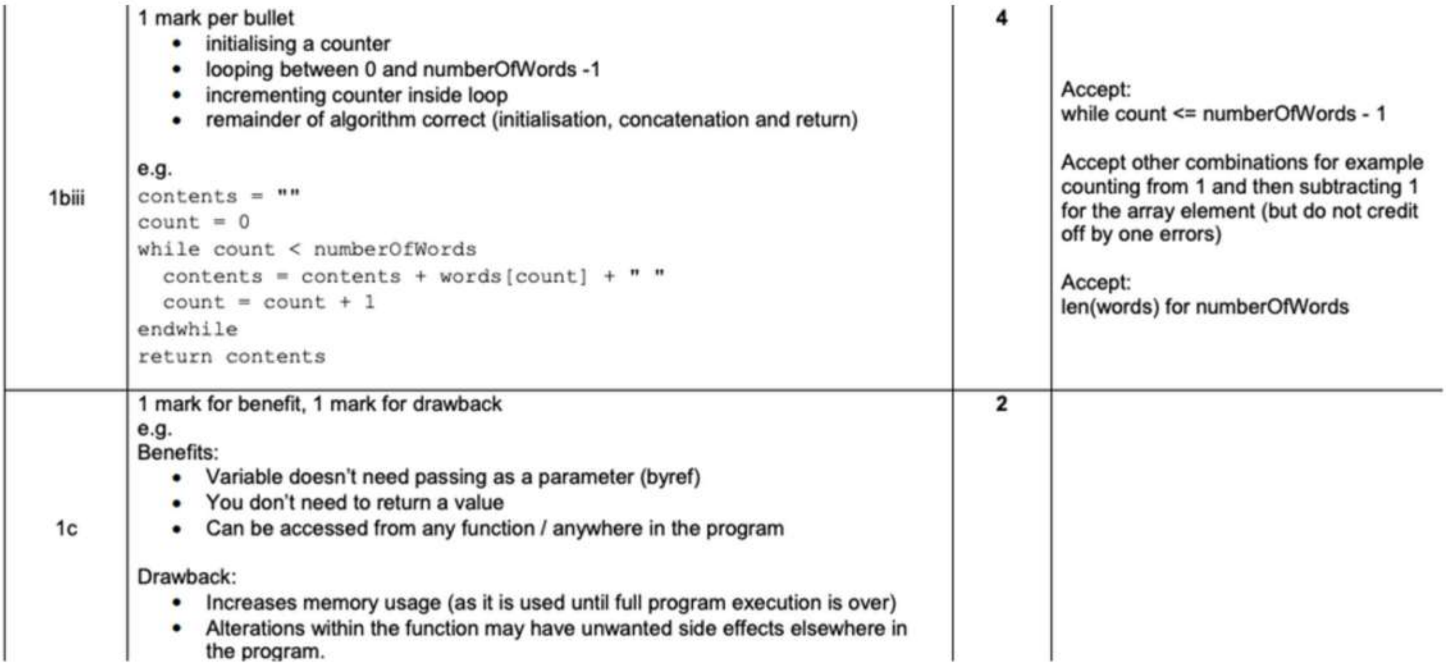


5	a	i	• sequence	1	
5	a	ii	• selection // branching	1	
5	d	i	it can only be accessed within the subroutine//block in which it is declared	1	
5	d	ii	1 mark for benefit e.g. • Increases data integrity • More efficient memory usage • Stops other subroutines accidently altering variable 1 mark for drawback e.g. • Cannot be accessed directly by other subroutines • It has to be passed into a subroutine as a parameter	2	

2022



1bi	1 mark for each variable • `contents` • `count` • `numberOfWords` • `words / words[]`	2	Accept exact spelling only Do not award numberOfWords if there are obvious spaces in 'number of Words'. It must be a valid identifier.
1bii	1 mark per bullet • By reference the function receives the memory location of the data • By value the function receives a copy of the variable • By reference will make changes to the original variable • By value will make changes to the copy of the variable • By reference will overwrite data in the original variable • By value will not overwrite the data in the original variable • By reference will keep the changes after the function ends • By value will not keep the changes after the function ends	2	Must cover byVal and byRef for 2 marks to be awarded. Must be clear that byVal is a copy of the original value.

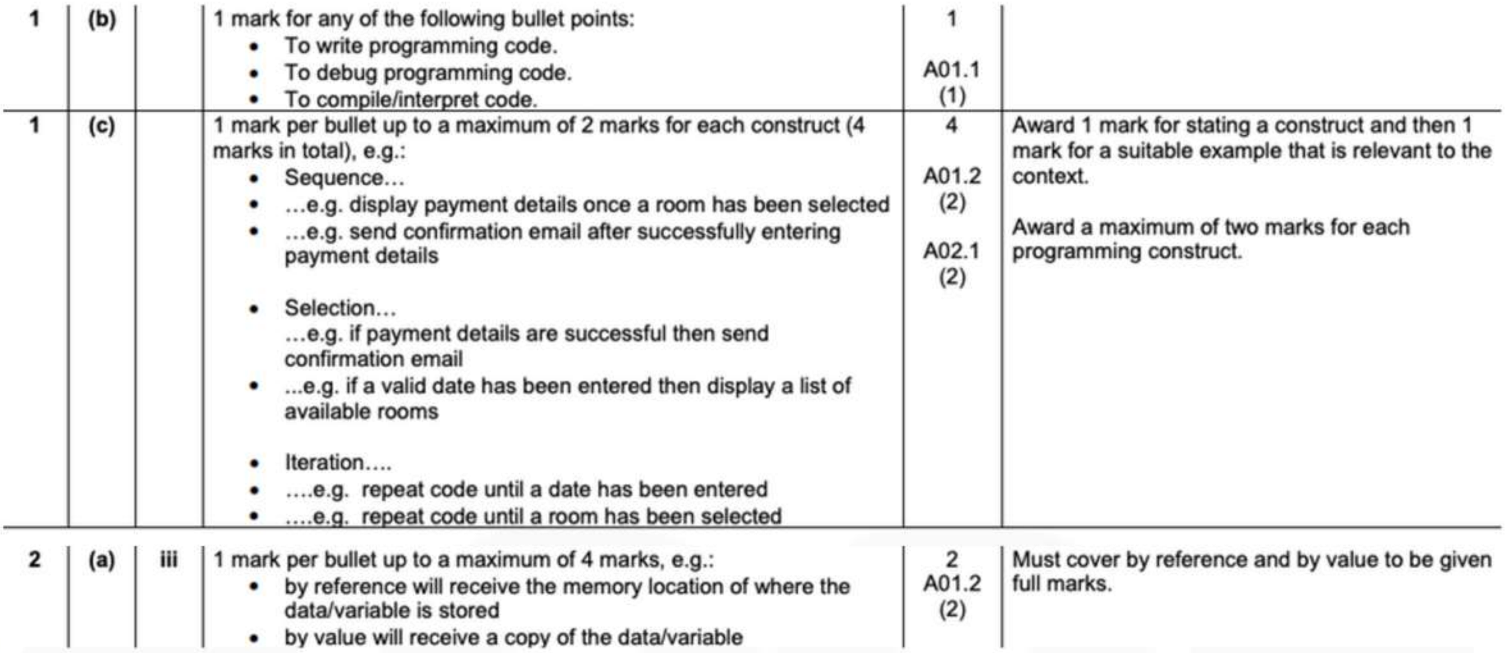


1biii	1 mark per bullet
	• initialising a counter
	• looping between 0 and numberOfWords -1
	• incrementing counter inside loop
	• remainder of algorithm correct (initialisation, concatenation and return)
	e.g.
	`contents = ""`
	`count = 0`
	`while count < numberOfWords`
	`  contents = contents + words[count] + " "`
	`  count = count + 1`
	`endwhile`
	`return contents`
		4	Accept:
			while count <= numberOfWords - 1
			Accept other combinations for example counting from 1 and then subtracting 1 for the array element (but do not credit off by one errors)
			Accept:
			len(words) for numberOfWords

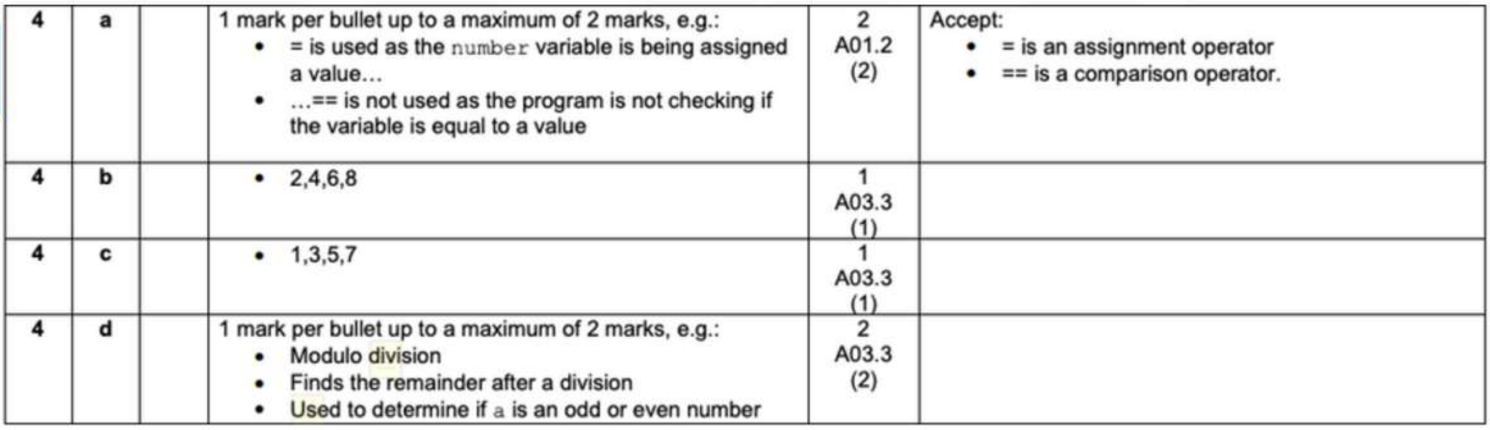
1c	1 mark for benefit, 1 mark for drawback e.g. Benefits: • Variable doesn't need passing as a parameter (byref) • You don't need to return a value • Can be accessed from any function / anywhere in the program Drawback: • Increases memory usage (as it is used until full program execution is over) • Alterations within the function may have unwanted side effects elsewhere in the program.	2	

1d	1 mark per identification 1 mark for expansion, max 2 each. Write: e.g. • Auto-complete • Start typing an identifier/command and it fills in the rest • Auto-indent • Indents code automatically within structures to avoid errors • Coloured command words // pretty printing // syntax highlighting • Shows which commands are correct // help identify key elements Test e.g. • Breakpoints • Stop the program running at a set point to check variables • Variable watch window • Display the values of the variables while the program is run • Stepping • Run one line at a time and check variables Unit Testing • Automated tests to be run to check changes ensure changes haven't introduced errors.	4	
1e	1 mark per bullet to max 2 • Saves time from having to write the same algorithm repeatedly • Reduced testing requirements • Can be taken and used in different programs as well as the program they are written in // can be used in a program library	2	Allow other suitable answers

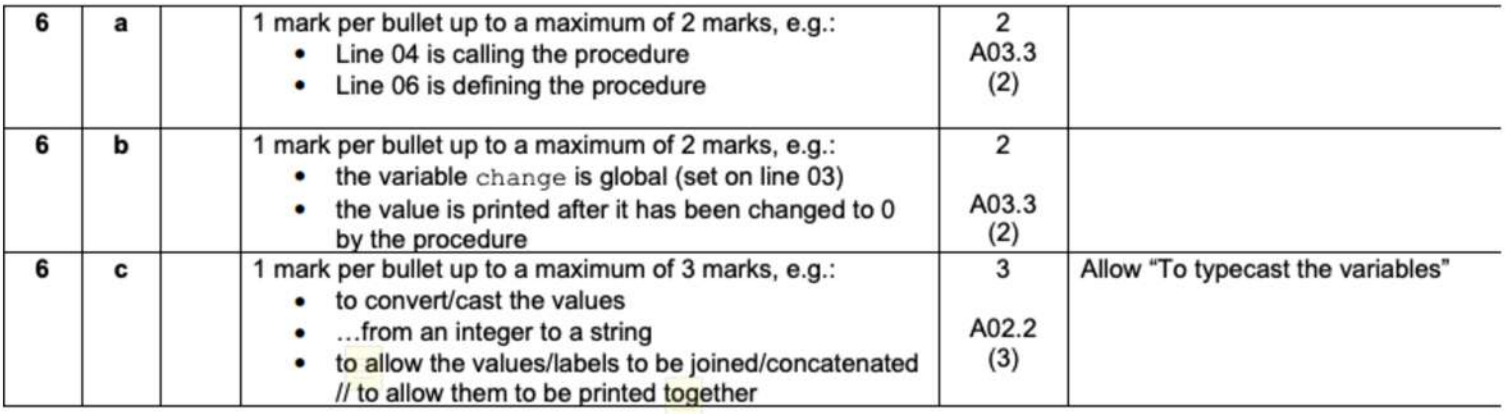
<table>
<tr><td>7ai</td><td>Line number 5</td><td>1</td></tr>
<tr><td>7aii</td><td>1 mark per feature
• A function that calls itself // a function that is defined in terms of itself
• …has a base case (that terminates the recursion)</td><td>2</td></tr>
<tr><td>7b</td><td>
<table>
<tr><th>Function call</th><th>number</th><th>return</th><th>Marking Guidance</th></tr>
<tr><td>calculate(5)</td><td>5</td><td>15</td><td>1 Mark</td></tr>
<tr><td>calculate(4)</td><td>4</td><td>10</td><td>1 Mark</td></tr>
<tr><td>calculate(3)</td><td>3</td><td>6</td><td>1 Mark</td></tr>
<tr><td>calculate(2)</td><td>2</td><td>3</td><td>1 Mark</td></tr>
<tr><td>calculate(1)</td><td>1</td><td>1</td><td>1 Mark</td></tr>
</table>
</td><td>5</td></tr>
<tr><td>7c</td><td>calculate(10)</td><td>1</td></tr>
</table>



1	(b)		1 mark for any of the following bullet points: • To write programming code. • To debug programming code. • To compile/interpret code.	1 A01.1 (1)	
1	(c)		1 mark per bullet up to a maximum of 2 marks for each construct (4 marks in total), e.g.: • Sequence… • …e.g. display payment details once a room has been selected • …e.g. send confirmation email after successfully entering payment details • Selection… …e.g. if payment details are successful then send confirmation email • ...e.g. if a valid date has been entered then display a list of available rooms • Iteration…. • ….e.g. repeat code until a date has been entered • ….e.g. repeat code until a room has been selected	4 A01.2 (2) A02.1 (2)	Award 1 mark for stating a construct and then 1 mark for a suitable example that is relevant to the context. Award a maximum of two marks for each programming construct.
2	(a)	iii	1 mark per bullet up to a maximum of 4 marks, e.g.: • by reference will receive the memory location of where the data/variable is stored • by value will receive a copy of the data/variable	2 A01.2 (2)	Must cover by reference and by value to be given full marks.



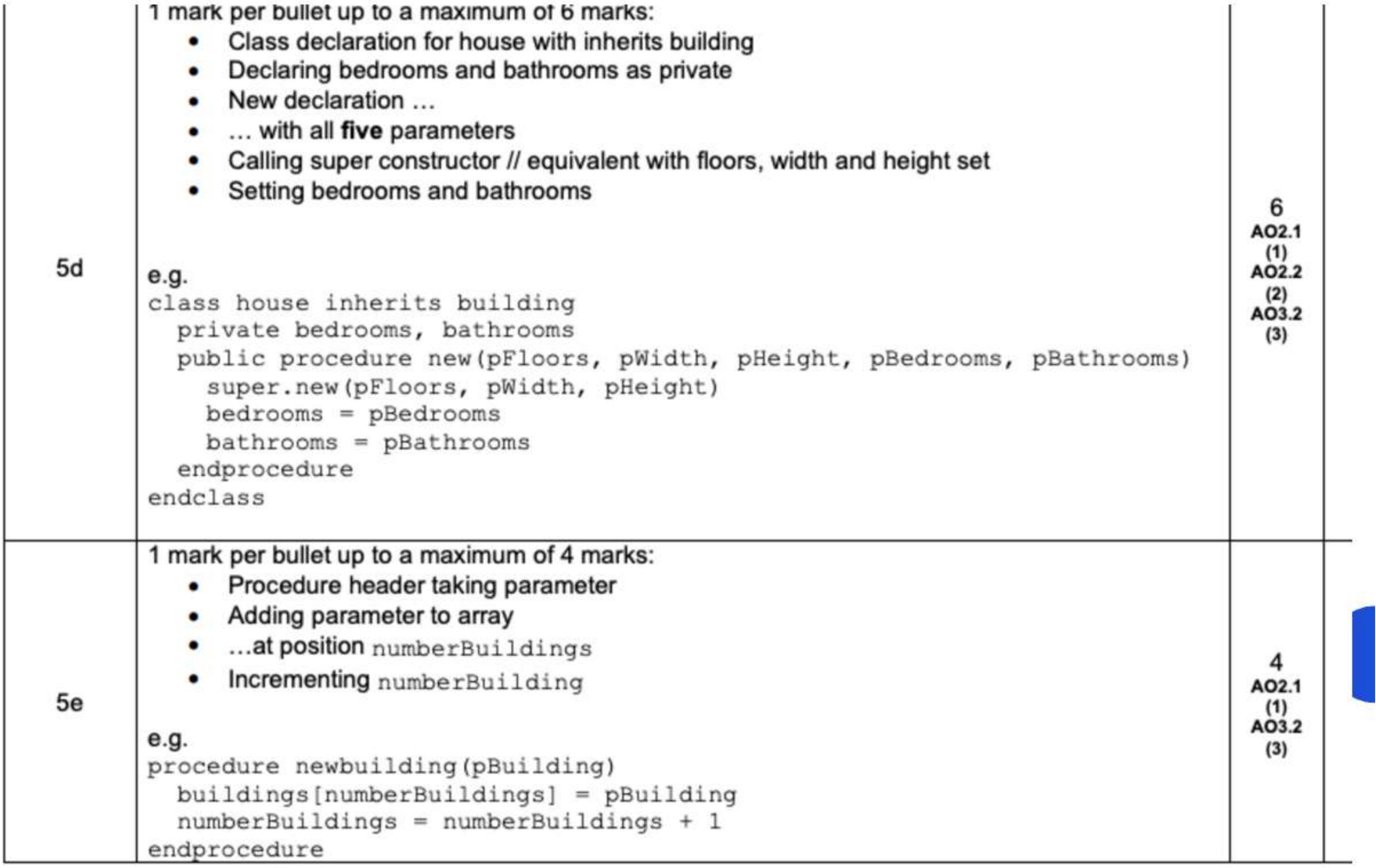
4	a		1 mark per bullet up to a maximum of 2 marks, e.g.: • = is used as the `number` variable is being assigned a value… • …== is not used as the program is not checking if the variable is equal to a value	2 A01.2 (2)	Accept: • = is an assignment operator • == is a comparison operator.
4	b		• 2,4,6,8	1 A03.3 (1)	
4	c		• 1,3,5,7	1 A03.3 (1)	
4	d		1 mark per bullet up to a maximum of 2 marks, e.g.: • Modulo division • Finds the remainder after a division • Used to determine if `a` is an odd or even number	2 A03.3 (2)	



6	a		1 mark per bullet up to a maximum of 2 marks, e.g.: • Line 04 is calling the procedure • Line 06 is defining the procedure	2 A03.3 (2)	
6	b		1 mark per bullet up to a maximum of 2 marks, e.g.: • the variable `change` is global (set on line 03) • the value is printed after it has been changed to 0 by the procedure	2 A03.3 (2)	
6	c		1 mark per bullet up to a maximum of 3 marks, e.g.: • to convert/cast the values • …from an integer to a string • to allow the values/labels to be joined/concatenated // to allow them to be printed together	3 A02.2 (3)	Allow "To typecast the variables"

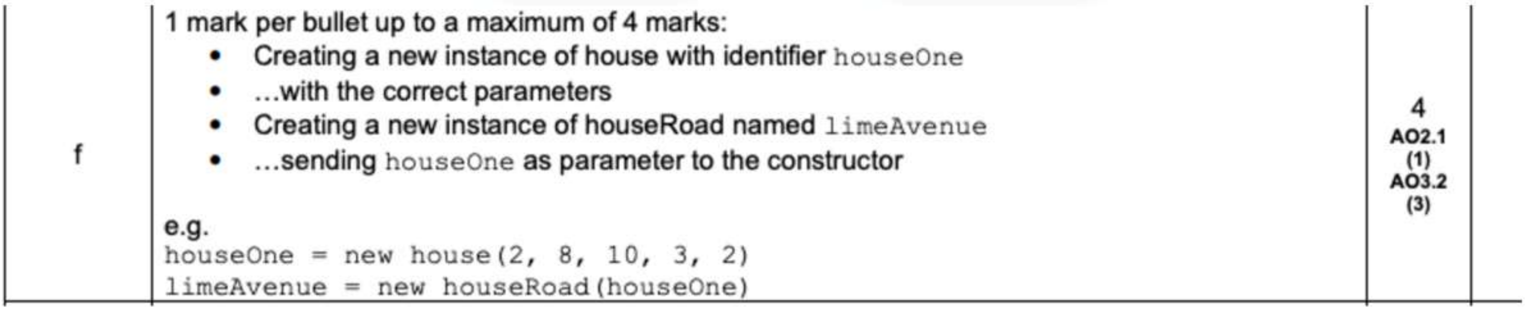
4	**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of IDEs; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples 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. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level (4-6 marks)** The candidate demonstrates reasonable knowledge and understanding of 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. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence* **Mark Band 1 – Low Level (1-3 marks)** The candidate demonstrates a basic knowledge of IDEs with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides nothing more than an unsupported assertion. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* **0 marks** No attempt to answer the question or response is not worthy of credit.	9 AO1.1 (2) AO1.2 (2) AO2.1 (2) AO3.3 (3)	**AO1: Knowledge and Understanding** **Indicative content** e.g. • IDE is software that includes an editor, compiler, run-time environment Creating • Autocorrect • Autocomplete • Pretty printing Testing • Breakpoints • Stepping • Variable watch window **AO2: Application** e.g. • Tell you when you make a syntax error • Allows you to write and run the code in one piece of software • Suggests code so you don't have to remember code, or autocorrect spelling mistakes • Helps you trace the program so you can see what happens when values change without having to manually insert print statements etc. • Autogenerate boilerplate code. **AO3: Evaluation** Candidates will need to evaluate the benefits and drawbacks of using IDEs. e.g. • Reduce errors through autocorrect and suggestions • 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 • Write and test in one environment so you don't have to close and run elsewhere, then re-open etc.
5a	To create an instance of an object from a class	1 AO2.1 (1)	
5b	1 mark per bullet up to a maximum of 2 marks, e.g: • When the child/derived/subclass class office/house takes on attributes/methods… • … from building / parent/base/superclass/ class	2 AO1.2 (1) AO2.2 (1)	
5c	1 mark for each completed statement up to a maximum of 5 marks:	5 AO2.2 (3) AO3.2 (2)	Accept other specific language conventions that would correctly achieve the same outcomes.

```
class building
  private numberFloors
  private width
  private height
  public procedure new(pFloors, pWidth, pHeight)
    numberFloors = pFloors
    width = pWidth
    height = pHeight
  endprocedure
  public function getNumberFloors()
    return numberFloors
  endfunction
  public function setNumberFloors(pFloors)
    if pFloors >= 1 then
      numberFloors = pFloors
      return true
    else
      return false
    endif
  endfunction
endclass
```



5d	1 mark per bullet up to a maximum of 6 marks:
	• Class declaration for house with inherits building
	• Declaring bedrooms and bathrooms as private
	• New declaration …
	• … with all **five** parameters
	• Calling super constructor // equivalent with floors, width and height set
	• Setting bedrooms and bathrooms
	e.g.
	`class house inherits building`
	`  private bedrooms, bathrooms`
	`  public procedure new(pFloors, pWidth, pHeight, pBedrooms, pBathrooms)`
	`    super.new(pFloors, pWidth, pHeight)`
	`    bedrooms = pBedrooms`
	`    bathrooms = pBathrooms`
	`  endprocedure`
	`endclass`
		6
		AO2.1
		(1)
		AO2.2
		(2)
		AO3.2
		(3)

5e	1 mark per bullet up to a maximum of 4 marks:
	• Procedure header taking parameter
	• Adding parameter to array
	• …at position `numberBuildings`
	• Incrementing `numberBuilding`
	e.g.
	`procedure newbuilding(pBuilding)`
	`  buildings[numberBuildings] = pBuilding`
	`  numberBuildings = numberBuildings + 1`
	`endprocedure`
		4
		AO2.1
		(1)
		AO3.2
		(3)



f	1 mark per bullet up to a maximum of 4 marks:
	• Creating a new instance of house with identifier `houseOne`
	• …with the correct parameters
	• Creating a new instance of houseRoad named `limeAvenue`
	• …sending `houseOne` as parameter to the constructor
	e.g.
	`houseOne = new house(2, 8, 10, 3, 2)`
	`limeAvenue = new houseRoad(houseOne)`
		4
		AO2.1
		(1)
		AO3.2
		(3)

8bi	1 mark for each correctly completed statement up to a maximum of 5 marks:
	`function countRow(puzzle:byref, rowNum:byval)`
	   `count = 0`
	   `output = " "`
	   `for i = 0 To` **4**
	      `if puzzle[rowNum, i] ==` **1** `then`
	         `count = count + 1`
	      `elseif count >= 1 then`
	         `output = output + str(`**count**`) + " "`
	         `count = 0`
	      `endif`
	   `next i`
	   `if count>= 1 then`
	         `output=ouput+str(count)`
	   `elseif output == "" then`
	         `output =` **"0"**
	   `endif`
	   `return` **output**
	`endfunction`
		5
		AO2.2
		(2)
		AO3.2
		(3)
			Accept
			for i = 0 to **row.length-1**
			for i = 0 to **row.length**
			for i=0 to **5**

8bii	1 mark per bullet up to a maximum of 2 marks, e.g: • Initialise the variable output… • …with a space • …for use later on in the code… • …So it can be used for concatenation later in the code … • …to avoid an error being generated	2 AO1.2 (1) AO2.2 (1)	
8biii	1 mark per bullet up to a maximum of 3 marks, e.g: • check the value stored in each index • check whether it is at the end of a row • check whether each row has been given an output or not	3 AO2.2 (3)	

8biv	1 mark per bullet up to a maximum of 6 marks:
	• Procedure heading for `displayRowAnswer`
	• …taking puzzle as parameters
	• Nested loops through all array elements
	• …outputting all rows
	• … at the end of each row calling `countRow` ….
	• …..with parameters `puzzle` and the current loop counter
	
	e.g.
	`procedure displayRowAnswer(puzzle)`
	   `for i = 0 To 4`
	      `for j = 0 To 4`
	         `print(puzzle[i, j] + " ")`
	      `next j`
	      `print ("    " + countRow(puzzle, i))`
	   `next i`
	`endprocedure`
		6
		AO2.2
		(3)
		AO3.2
		(3)
			Accept
			for i = 0 to **row.length-1**
			for i = 0 to **row.length**
			for i=0 to **5**

8bv	1 mark for clearly identifying each error and giving the correction. • Line 01 needs `answerGrid` as parameter • Line 04 `==` should be `!=` • Line 08 should be `next row`	3 AO2.1 (3)	Do not award marks for line numbers alone without stating the error. Consider 1 mark for not changing line 04 but changing 05 to true and 09 to False

8c	**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of local and global variables; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. The candidate is able to weigh up the use of both local and global variables which results in a supported and realistic judgment as to whether it is possible to use them in this context. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level (4-6 marks)** The candidate demonstrates reasonable knoledge and understanding of local and global variables; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate makes a reasonable attempt to come to a conclusion showing some recognition of influencing factors that would determine whether it is possible to use local and global variables in this context. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence* **Mark Band 1 – Low Level (1-3 marks)** The candidate demonstrates a basic knowledge of local and global variables with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides nothing more than an unsupported assertion. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* **0 marks** No attempt to answer the question or response is not worthy of credit.	**9** **AO1.1** **(2)** **AO1.2** **(2)** **AO2.1** **(2)** **AO3.3** **(3)**	**AO1: Knowledge and Understanding** **Indicative content** Local variables: • Scope within the module defined within • Cannot access externally unless passed as parameter, or returned from function • When module is exited, memory of variable is freed Global variables: • Scope within the entire program • Can access from anywhere • Retained in memory permanently ByRef Points to location of variable ByVal Sends the value **AO2: Application** • If global the arrays can be accessed from all modules by direct reference • If local to the main, the arrays will need to be passed as parameters byreference • Can send ByVal – but not always possible with arrays in some languages • Modules are self contained and then can be reused in other programs he wants to create without needing to take the global variables with them **AO3: Evaluation** e.g. • +ve Local = memory efficient • +ve Global = easier programming, simpler to follow, easier to debug • -ve Global = memory inefficient, not good programming technique • -ve Local = more difficult to trace/debug/follow where the values are passed • Relatively small program – don't know about overall plan for it, it might not be memory intensive, unlikely anyone else is going to access/amend e.g. use as a library – therefore global would not waste significant resources

3a

1 mark per bullet

- Calculation of result to 3
- Call with `thisFunction(theArray, num1=4, num2=7, num3=35)`
- Result = 5
- call with `thisFunction(theArray,num1=6,num2=7,num3=35)`
- (Result = 6) return of value 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

5
AO2.1
(3)
AO2.2
(2)

3b

Binary search

1
AO2.1
(1)

3c

1 mark per bullet to max 4, e.g.

- Recursion uses more memory…
- …iteration uses less memory
- Recursion declares new variables //variables are put onto the stack each time…
- ...iteration reuses the same variables
- Recursive can run out of memory/stack space…
- …while iteration cannot run out of memory
- Recursion can express a problem more elegantly // in fewer lines of code…
- …while iteration can take more lines of code // be harder to understand
- Recursion will be self-referential // will call itself…
- … whereas iteration does not

4
AO1.1
(2)
AO1.2
(2)

3d

1 mark per bullet to max 6

- Retains function call
- Uses a loop
- …that will loop until all elements inspected or value found
- Updates num1 appropriately
- Updates num2 appropriately
- Returns -1 in the correct place if the value has not been found
- Returns the result in the correct place if the value has been found

e.g.

```
function thisFunction(theArray, num1, num2, num3)

  while (true)
    result = num1 + ((num2 - num1) DIV 2)
    if num2 < num1 then
      return -1
    else
      if theArray[result] < num3 then
       num1 = result + 1
      elseif theArray[result] > num3 then
       num2 = result - 1
      else
       return result
      endif
    endif
  endwhile
endfunction
```

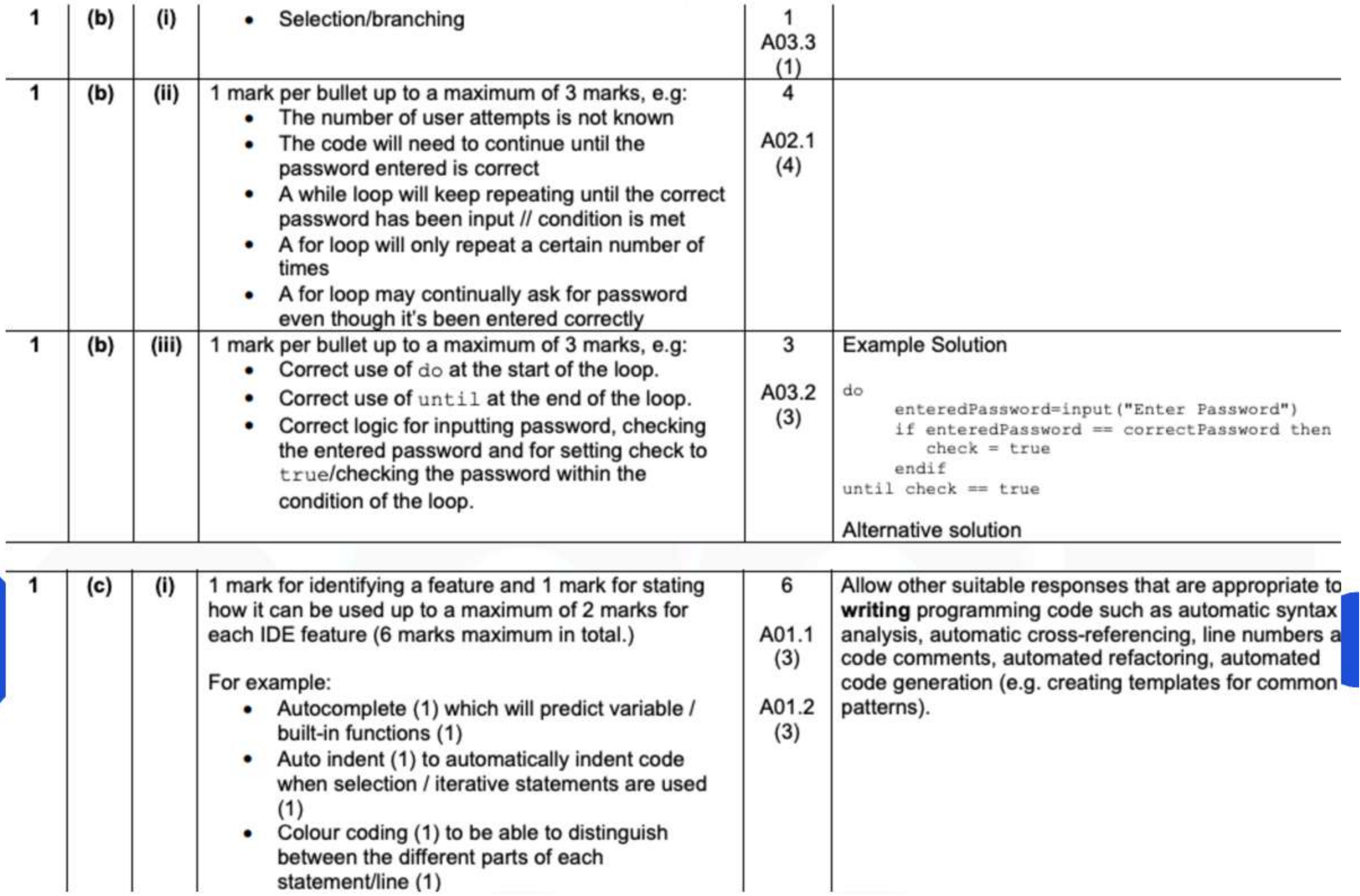
6
AO2.2
(3)
AO3.1
(3)

4a	1 mark per bullet • By reference will change the actual contents of the array in the main program// when control returns to the main program the array will be sorted • By value would create a copy and not change the original // when control returns to the main program the array will **not** be sorted • By value the array is local to the function. • By reference will use less memory	**2** AO1.2 (1) AO2.2 (1)
4b	1 mark pet bullet to max 3 • Descending order • Line 07 (`dataArray[tempos]<temp`) has the comparison… • …that checks if current position is less than item to insert and… • …breaks out of loop when current position is less than or equal to item to insert	**3** AO1.2 (1) AO2.2 (2)

6bi	1 mark per bullet
	• Class declaration and all code is nested within the class
	• Two private identifiers data and pointer (with suitable data types if given)
	• Public constructor heading as a procedure (public may be implied but cannot be private) taking both parameters as given in table
	• Assigns parameters to the attributes
	
	e.g.
	`public class node`
	`  private data as real`
	`  private pointer as integer`
	`  public procedure new(newData, newPointer)`
	`    data = newData`
	`    pointer = newPointer`
	`  endprocedure`
	`endclass`
		4
		AO2.2
		(1)
		AO3.3
		(3)

6bii	1 mark per bullet to max 2 • A get method allows the attribute to be accessed / returned • A set method allows the attribute to be changed (with parameters)	**2** AO2.2 (2)

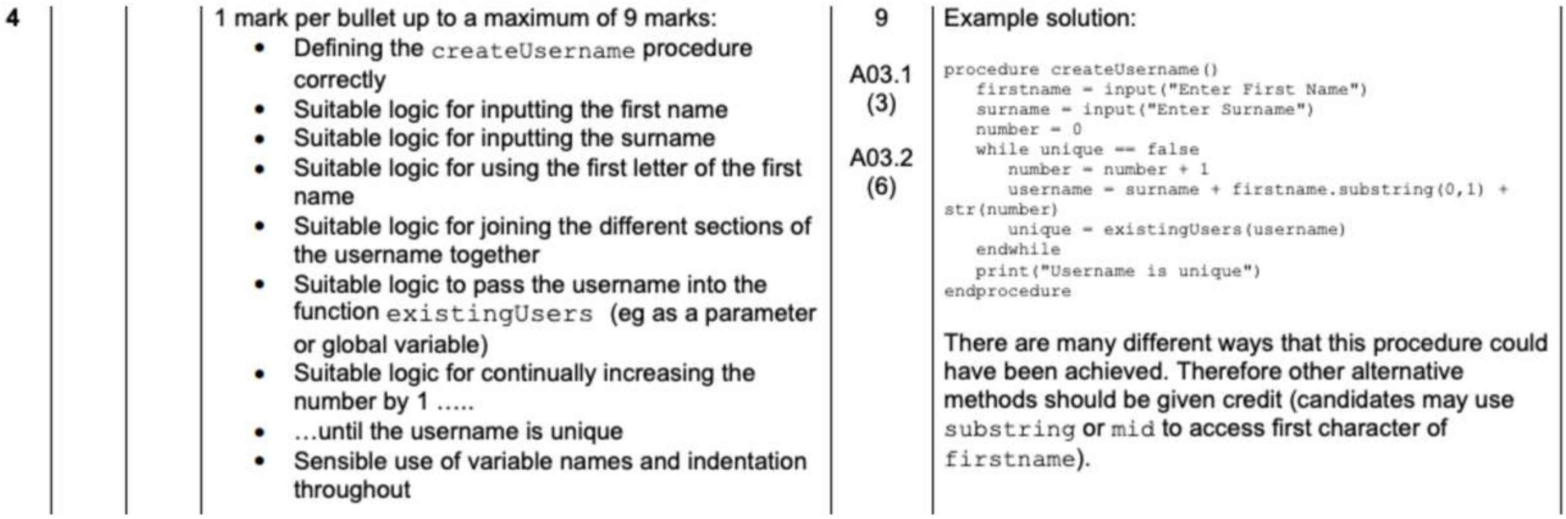
6e	**Mark Band 3 – High level (9-12 marks)** The candidate demonstrates a thorough knowledge and understanding of the object orientied techniques; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation.The candidate is able to weigh up the use of all of the object orientied techniques which results in a supported and realistic judgment as to whether it is possible to use them in this context. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level (5-8 marks)** The candidate demonstrates reasonable knowledge and understanding of the object orientied techniques; 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 each object orientied technique in this context. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence* **Mark Band 1 – Low Level (1-4 marks)** The candidate demonstrates a basic knowledge of the object orientied 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. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.*	12 AO1.1 (3) AO1.2 (3) AO2.1 (3) AO3.3 (3)	**AO1: Knowledge and Understanding** **Indicative content** • Classes, this a template. It will define what attributes and methods an object should have. • Objects, when you create an instance of a class. Each object that is instantiated from the same class will share the same attributes and methods. • Inheritance, when a sub class takes on the attributes and methods from a superclass/parent class. It can also have its own extra attributes/methods. • Overriding, when a method name is the same in a parent and sub class, then the method in the parent/super class will be overridden • Encapsulation, this protects attributes of an object by making them private so that they can't be accessed or altered accidentally by other objects. **AO2: Application** • A class can be used to declare the attributes and methods for the linked list. These will initialise the nodes and join them. • Objects can then be used be used to instantiate the class each time a new linked list is needed. Each can be given a different identifier by the other programs. • Further subclasses may be used by other programs. These can therefore take on the attributes and methods from the base class. These can also be changed or overridden depending on the purpose of the other programs. • Encapsulation can be used by using set and get methods to ensure that the nodes in the linked list are changed in a way that is intended. **AO3: Evaluation**
	0 marks No attempt to answer the question or response is not worthy of credit.		• Use of OPP techniques will allow for code reusability. His linked list can be saved as library and then reused many times leading to less code. • OOP also allows programs to be easier to modify and maintain.



1	(b)	(i)	- Selection/branching	1 A03.3 (1)	
1	(b)	(ii)	1 mark per bullet up to a maximum of 3 marks, e.g: - The number of user attempts is not known - The code will need to continue until the password entered is correct - A while loop will keep repeating until the correct password has been input // condition is met - A for loop will only repeat a certain number of times - A for loop may continually ask for password even though it's been entered correctly	4 A02.1 (4)	

1	(b)	(iii)	1 mark per bullet up to a maximum of 3 marks, e.g:
			- Correct use of `do` at the start of the loop.
			- Correct use of `until` at the end of the loop.
			- Correct logic for inputting password, checking the entered password and for setting check to `true`/checking the password within the condition of the loop.
				3
				A03.2
				(3)
					Example Solution
					`do`
					`    enteredPassword=input("Enter Password")`
					`    if enteredPassword == correctPassword then`
					`        check = true`
					`    endif`
					`until check == true`
					Alternative solution

1	(c)	(i)	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.) For example: - Autocomplete (1) which will predict variable / built-in functions (1) - Auto indent (1) to automatically indent code when selection / iterative statements are used (1) - Colour coding (1) to be able to distinguish between the different parts of each statement/line (1)	6 A01.1 (3) A01.2 (3)	Allow other suitable responses that are appropriate to **writing** programming code such as automatic syntax analysis, automatic cross-referencing, line numbers a code comments, automated refactoring, automated code generation (e.g. creating templates for common patterns).



4			1 mark per bullet up to a maximum of 9 marks:
			- Defining the `createUsername` procedure correctly
			- Suitable logic for inputting the first name
			- Suitable logic for inputting the surname
			- Suitable logic for using the first letter of the first name
			- Suitable logic for joining the different sections of the username together
			- Suitable logic to pass the username into the function `existingUsers` (eg as a parameter or global variable)
			- Suitable logic for continually increasing the number by 1 .....
			- ...until the username is unique
			- Sensible use of variable names and indentation throughout
				9
				A03.1
				(3)
				A03.2
				(6)
					Example solution:
					`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`
					There are many different ways that this procedure could have been achieved. Therefore other alternative methods should be given credit (candidates may use `substring` or `mid` to access first character of `firstname`).

3ai	if	1 AO1.1 (1)	
3aii	1 mark per bullet • Branching decides which code is run / only runs code once • Iteration repeatedly runs the same code in the same sequence	2 AO1.2 (2)	
3aiii	num1, num2	1 AO2.1 (1)	Exact identifier names require
3aiv	1 mark per bullet • By Value • … the original values do not need to be modified • … byRef would not work / would cause the routine to crash	2 AO2.2 (2)	
3av	1 mark per bullet • Gives the remainder after division • E.g. 10 MOD 3 = 1	2 AO1.1 (1) AO1.2 (1)	
3b	1 mark per bullet to max 3 • Num2 != 0 therefore return GCD(20,10) • Num2 != 0 therefore return GCD(10,0) • Final return value = 10	3 AO2.1 (1) AO2.2 (2)	Allow FT for numerical errors
3ci	1 mark for benefit, 1 mark for drawback Benefit: • The program can/might run faster • Cannot run out of stack space/memory • Easier to trace/follow Drawback: • Iteration can lead to lengthier code • Iteration can lead to code that looks more complex / is harder to understand • some problems are more elegantly coded with a recursive solution	2 AO1.1 (1)	

3cii	1 mark for each correct statement
	function newGCD(num1, num2)
	  temp = 0
	  while (num2 != **0**)
	    temp = num2
	    num2 = num1 MOD **num2**
	    num1 = temp
	  endwhile
	  return **num1**
	endfunction
		4
		AO2.2
		(2)
		AO3.2
		(2)	

7bi	1 mark per bullet to max 4
	• Class declaration
	• 3 attributes declared
	• Constructor
	• …taking parameters
	• …setting the attributes to the parameters
	
	e.g.
	`class GardenItem`
	`  private itemName`
	`  private length`
	`  private width`
	`  public procedure new(pItemName, pLength, pWidth)`
	`    itemName = pItemName`
	`    length = pLength`
	`    width = pWidth`
	`  endprocedure`
	`endclass`
		4
		
		AO1.1
		(1)
		AO2.1
		(1)
		AO2.2
		(1)
		AO3.2
		(1)
			Note that example answers are given in the specification pseudocode. Any pseudocode answer that can be understood by a 'competent programmer' should be accepted

7bii	1 mark per bullet
	• Class declaration inheriting from `gardenItem`
	• Additional 3 properties declared as private
	• Constructor takes all 5 parameters
	• Use of super (or equivalent) to set super class parameters
	• Remainder of properties set to parameters
	
	e.g.
	`class Tree inherits GardenItem`
	`  private height`
	`  private sun`
	`  private shade`
	`  public procedure new(pName, pHeight, pLenWidth,`
	`pSun, pShade)`
	`    super.itemName = pName`
	`    super.length = pLenWidth`
	`    super.width = pLenWidth`
	`    height = pHeight`
	`    sun = pSun`
	`    shade = pShade`
	`  endprocedure`
	`endclass`
		5
		
		AO1.1
		(1)
		AO2.2
		(1)
		AO3.2
		(3)
			Accept solutions that call the parent's constructor.
			
			`class Tree inherits GardenItem`
			`  private height`
			`  private sun`
			`  private shade`
			`  public procedure new(pName,`
			`pHeight, pLenWidth, pSun, pShade)`
			`    height = pHeight`
			`    sun = pSun`
			`    shade = pShade`
			
			`super.new(pName,pLenWidth,pLenWidth)`
			`  endprocedure`
			`endclass`

7biii	1 mark per bullet • Declaration of instance of tree (i.e. `new Tree`) • Storing result in `firstTree` • All parameters included and in the same order as 7bii • …with appropriate data types e.g. `firstTree = new Tree("Common Oak",40,40,true, true)`	4 AO1.1 (1) AO2.2 (1) AO3.2 (2)	

7biv	1 mark per bullet
	• Get method declaration
	• Returns `itemName`
	• Set method declaration
	• …takes value as a parameter
	• …assigns parameter to `itemName`
	e.g.
	`function getItemName()`
	`  return itemName`
	`endfunction`
	
	`procedure setItemName(newname)`
	`  itemName = newname`
	`endprocedure`
		4
		
		AO1.2
		(2)
		AO2.2
		(2)	

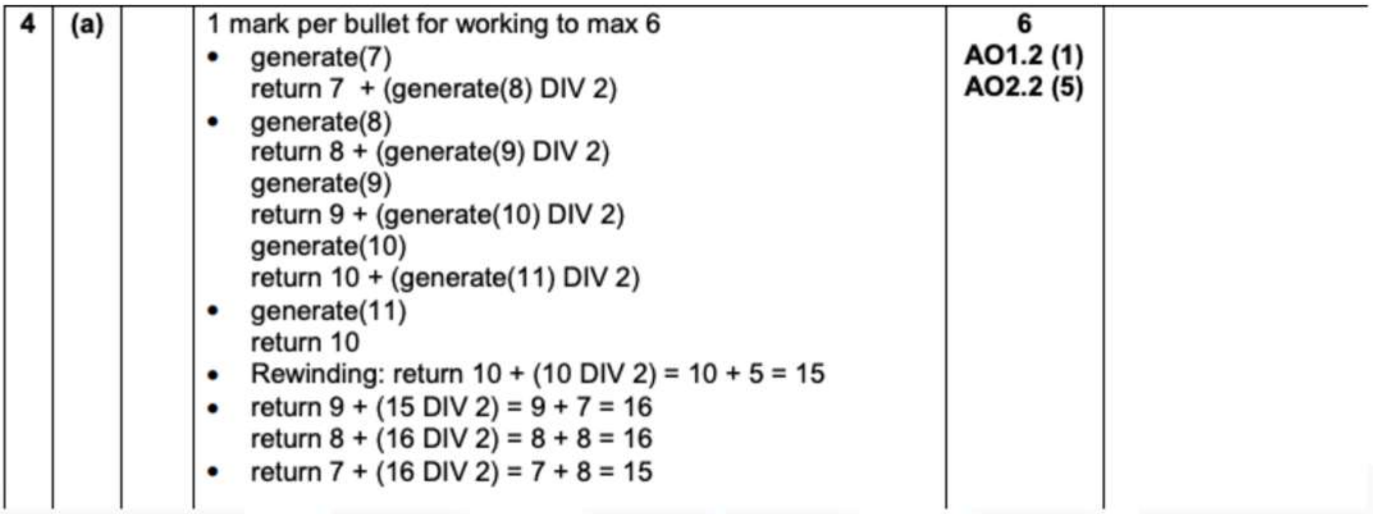
2	a	i	08	1 AO2.1 (1)	
2	a	ii	1 mark per bullet to max 3 • Compare the size/number of elements in the two arrays `data` and `nextData` • To set the number of times the loop will run // set the value of `loopCount` • … as many times as the array with the fewest items • Otherwise it will attempt to add an empty value • … which could cause a logic error	3 AO2.1 (1) AO2.2 (2)	
2	b	i	`data, nextData`	2 AO2.1 (2)	
2	b	ii	The actual data stored in the array will be changed	1 AO1.2 (1)	

4	a		1 mark per bullet to max 4 • Function declaration with parameter passed • Opening a file to read • Reading the data from a file • Closing the file and then returning the string	4 AO3.2 (4)

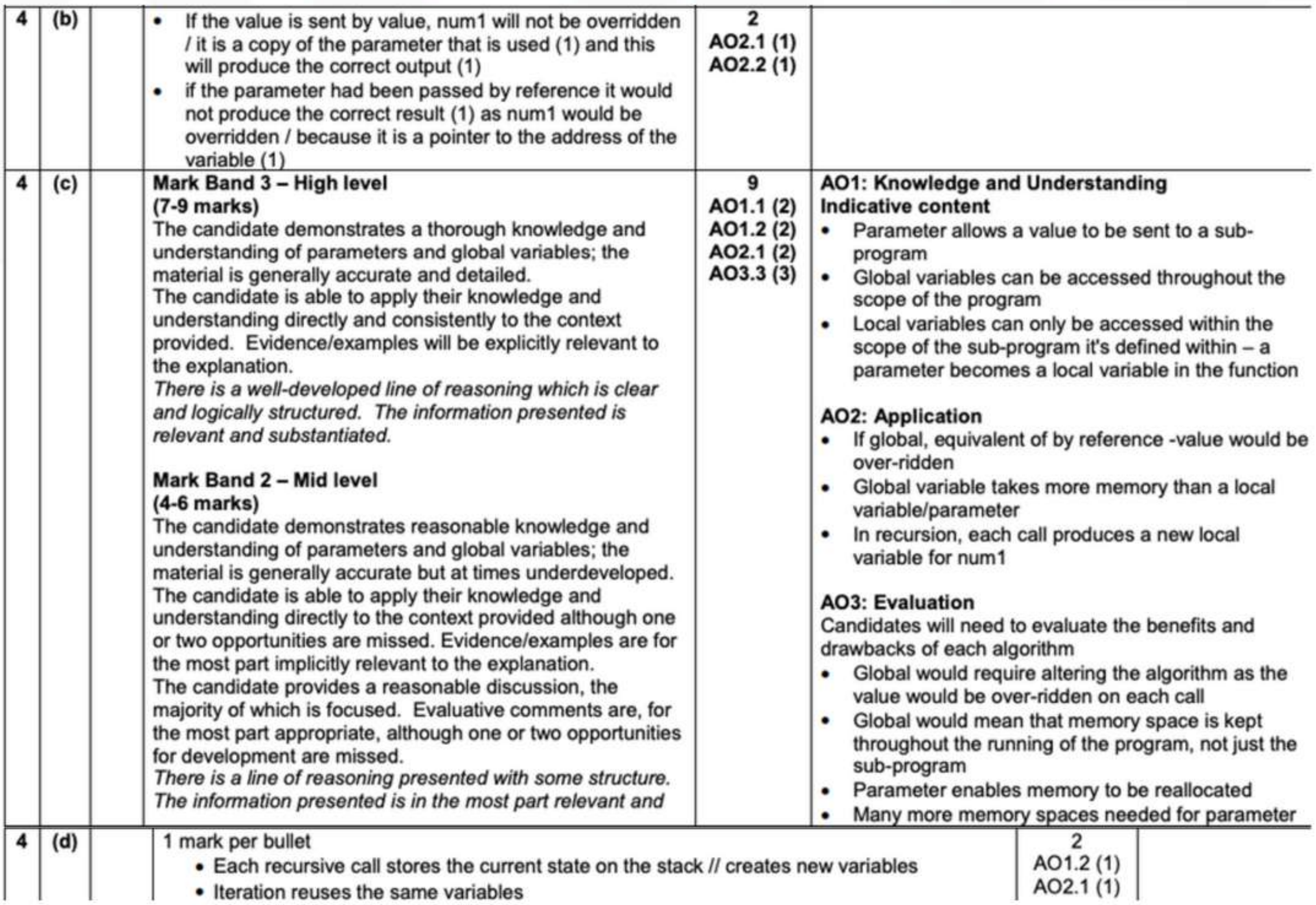
```
function getText(filename)
  file = openRead(filename)
  dataString = file.readLine()
  file.close()
  return dataString
endfunction
```

4	B		1 mark per bullet to max 7 • Procedure declaration • Reading file name as input • Calling getText() function with filename • Looping through all characters in the data • Checking if there is a "." (ASCII code 46) • … checking if the next character is a space " " (ASCII code 32) • … looping until there is no " " (ASCII code 32) • … checking if the character is lowercase • … changing the character to uppercase • Writing output to text file	7 AO2.2 (1) AO3.2 (6)

```
procedure fullStop()
  filename = input("Enter filename")
  textString = getText(filename)
  output = ""
  i = 0
  newSentence = True

  while i < textString.length - 1:

     nextChar = textString.substring(i, 1)

     if newSentence == True:
        if nextChar == " ":
           output = output + " "
        else:
            output = output + upper(nextChar)
            newSentence = False
     else:
        if nextChar == ".":
            newSentence = True
        output = output + char
     endif
       i = i + 1

  endwhile

  file = openWrite(filename)
  file.write(output)
  file.close()
endprocedure
```



4	(a)		1 mark per bullet for working to max 6
			• generate(7)
			return 7 + (generate(8) DIV 2)
			• generate(8)
			return 8 + (generate(9) DIV 2)
			generate(9)
			return 9 + (generate(10) DIV 2)
			generate(10)
			return 10 + (generate(11) DIV 2)
			• generate(11)
			return 10
			• Rewinding: return 10 + (10 DIV 2) = 10 + 5 = 15
			• return 9 + (15 DIV 2) = 9 + 7 = 16
			return 8 + (16 DIV 2) = 8 + 8 = 16
			• return 7 + (16 DIV 2) = 7 + 8 = 15
				6
				AO1.2 (1)
				AO2.2 (5)	



4	(b)		• If the value is sent by value, num1 will not be overridden / it is a copy of the parameter that is used (1) and this will produce the correct output (1) • if the parameter had been passed by reference it would not produce the correct result (1) as num1 would be overridden / because it is a pointer to the address of the variable (1)	**2** **AO2.1 (1)** **AO2.2 (1)**	
4	(c)		**Mark Band 3 – High level** **(7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of parameters 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. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level** **(4-6 marks)** The candidate demonstrates reasonable knowledge and understanding of parameters and global variables; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and*	**9** **AO1.1 (2)** **AO1.2 (2)** **AO2.1 (2)** **AO3.3 (3)**	**AO1: Knowledge and Understanding** **Indicative content** • Parameter allows a value to be sent to a sub-program • Global variables can be accessed throughout the scope of the program • Local variables can only be accessed within the scope of the sub-program it's defined within – a parameter becomes a local variable in the function **AO2: Application** • If global, equivalent of by reference -value would be over-ridden • Global variable takes more memory than a local variable/parameter • In recursion, each call produces a new local variable for num1 **AO3: Evaluation** Candidates will need to evaluate the benefits and drawbacks of each algorithm • Global would require altering the algorithm as the value would be over-ridden on each call • Global would mean that memory space is kept throughout the running of the program, not just the sub-program • Parameter enables memory to be reallocated • Many more memory spaces needed for parameter
4	(d)		1 mark per bullet • Each recursive call stores the current state on the stack // creates new variables • Iteration reuses the same variables	2 AO1.2 (1) AO2.1 (1)	

GCST

5 (d) (iii)

1 mark per bullet
- Setting variable to start at 0
- Suitable while structure (endwhile or clear indentation)
- looping 50 times
- Incrementing the variable within the loop

e.g. 1
```
function searchItem(dataItem)
    count = 0
    while count < 50
        if dataArray(count) == dataItem then
            return(count)
        endif
        count = count + 1
    endwhile
    return(-1)
endfunction
```

e.g. 2
```
function searchItem(dataItem)
    count = 0
    while count < 50 and dataArray[count]!=dataItem
        count = count + 1
    endwhile
    if  count==50
        count=-1
    endif
    return(count)
endfunction
```

4
AO1.2 (1)
AO3.1 (1)
AO3.2 (2)

AS - Level

1 b ii

1 mark per bullet, max 2 for by value, max 2 for by reference

by value:
- A local copy of the data is used
- Data is discarded when the subprogram exits
- Does not override/change the original data

by reference:
- Memory location of data is sent
- Changes are made to the original data
- Changes remain after the subprogram exits

4
AO1.2
(4)

1 b iv

1 mark for each bullet to max 6
- Initialising `total` to 0 and `add` to true
- `top` = 4 and `total` = 8
- `total` = 20 and `add` = false
- `top` = 3, `total` = 14, `add` = true
- `top` 2, 1. `Total` 16, -4, `add` false, true
- Output = -4

	numStack								
top	**0**	**1**	**2**	**3**	**4**	**5**	**total**	**add**	**Output**
5	20	2	6	12	8				
							0	true	
4							8		
							20	false	
3							14	true	
2							16	false	
1							-4	true	
0									-4

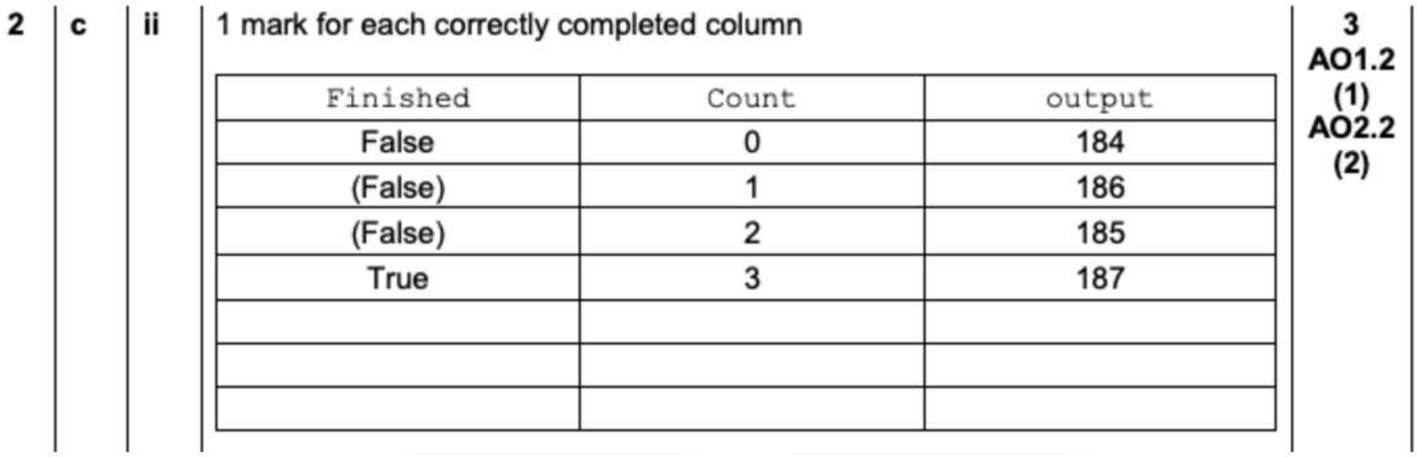
6
AO1.2
(3)
AO2.2
(3)

2 e

1 mark per bullet to max 3

e.g.
- Provides a text editor / allows the code to be written
- Provides debugging tools / allows the code to be tested
- Provides a translator/compiler/interpreter / provides a run-time environment / allows the code to be run
- Description of key feature e.g. colour coding keywords, autocomplete, breakpoints etc.

3
AO1.2
(3)

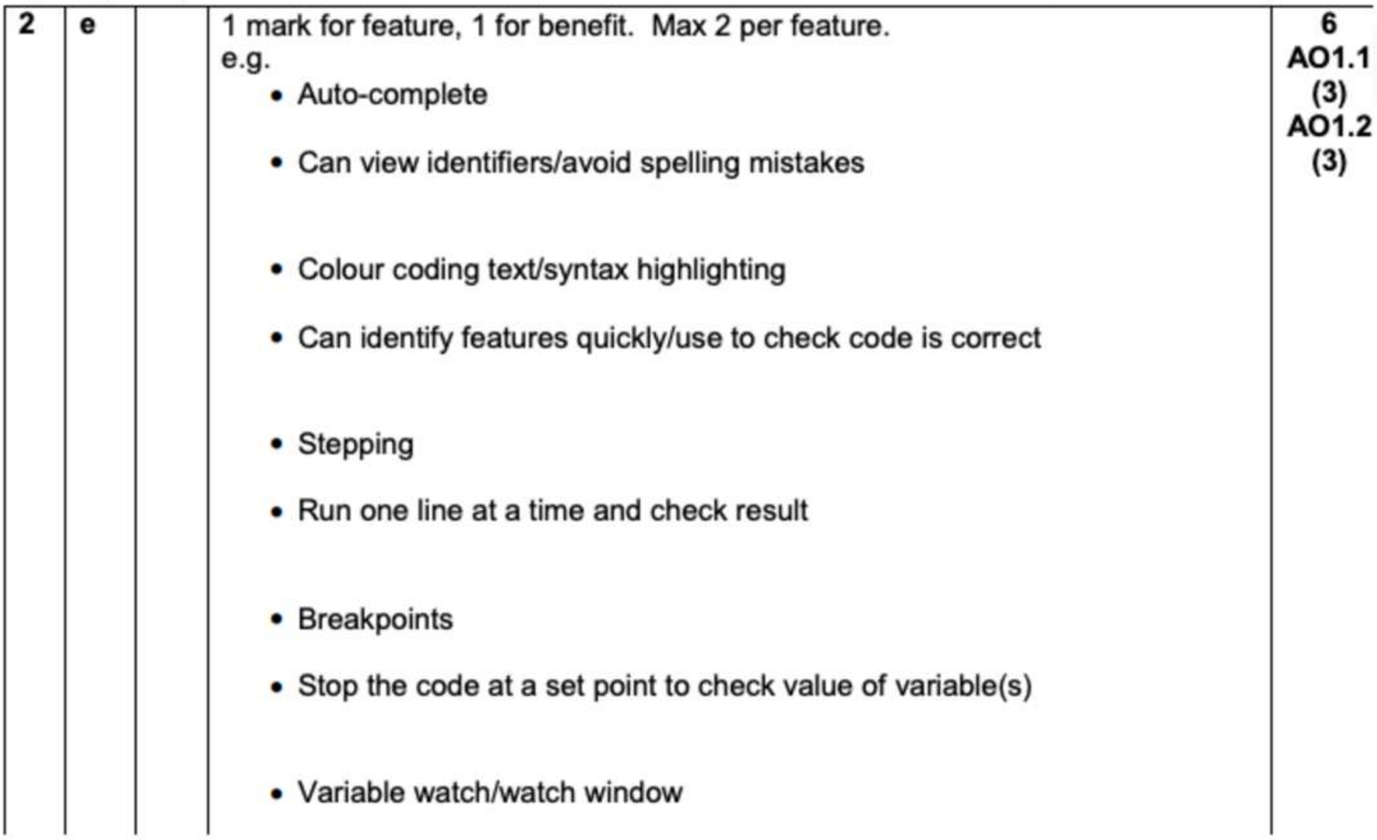


2 c ii

1 mark for each correctly completed column

Finished	Count	output
False	0	184
(False)	1	186
(False)	2	185
True	3	187

3
AO1.2
(1)
AO2.2
(2)



2 e

1 mark for feature, 1 for benefit. Max 2 per feature.
e.g.

- Auto-complete
- Can view identifiers/avoid spelling mistakes

- Colour coding text/syntax highlighting
- Can identify features quickly/use to check code is correct

- Stepping
- Run one line at a time and check result

- Breakpoints
- Stop the code at a set point to check value of variable(s)

- Variable watch/watch window

6
AO1.1
(3)
AO1.2
(3)

5 a

```
05 and 07
```

1
AO2.1
(1)

---

5 b

1 mark for each highlighted element

```
•  calculate(4,10)

    if 4 == 10 FALSE
    elseif 4 < 10  TRUE
       return calculate(4, (10-4))
       return calculate(4, 6)
            •  if 4 == 6 FALSE

               elseif 4 < 6 TRUE
                   return calculate(4, 6-4)
                   return calculate(4, 2)
                      •  if 4 == 2 FALSE

                         elseif 4 < 2 FALSE
                         else
                             return calculate(2, 4-2)
                             return calculate(2,2)
                                   •  if 2 == 2 TRUE

                                      return 2
                             return 2
                    return 2
     return 2
```

5
AO2.1
(5)

Allow trace table or any sensibl equivalent.

---

5 c

1 mark per bullet to max 4

- Suitable loop with correct condition
- In IF: Overwriting num2 with num2 – num1
- In ELSE: Overwriting num1 with num2...
- ... Overwriting num2 with num1-num2 correctly (using a temp variable)

e.g.

```
while num1 != num2
    if num1 < num2 then
        num2 = num2 - num1
    else
        temp = num1 - num2
        num1 = num2
        num2 = temp
    endif
endwhile
```

4
AO2.1
(1)
AO2.2
(1)
AO3.2
(2)

Alternatively swapping values by:

```
temp = num1
num1 = num2
num2 = temp - num2
```

---

6 c i

1 mark per bullet to max 3

- Defining procedure play
    - Resetting bored to 0
    - Outputting result

e.g.

```
procedure play()
    bored = 0
    print("bored: " + bored + "%")
endprocedure
```

3
AO3.2
(3)

---

6 c ii

1 mark per bullet to max 3

- Defining procedure read
    - Correct calculation
    - Outputting result

e.g.

```
procedure read()
    intelligence = intelligence * 1.006
    print("intelligence: " + intelligence)
endprocedure
```

3
AO2.2
(1)
AO3.2
(2)

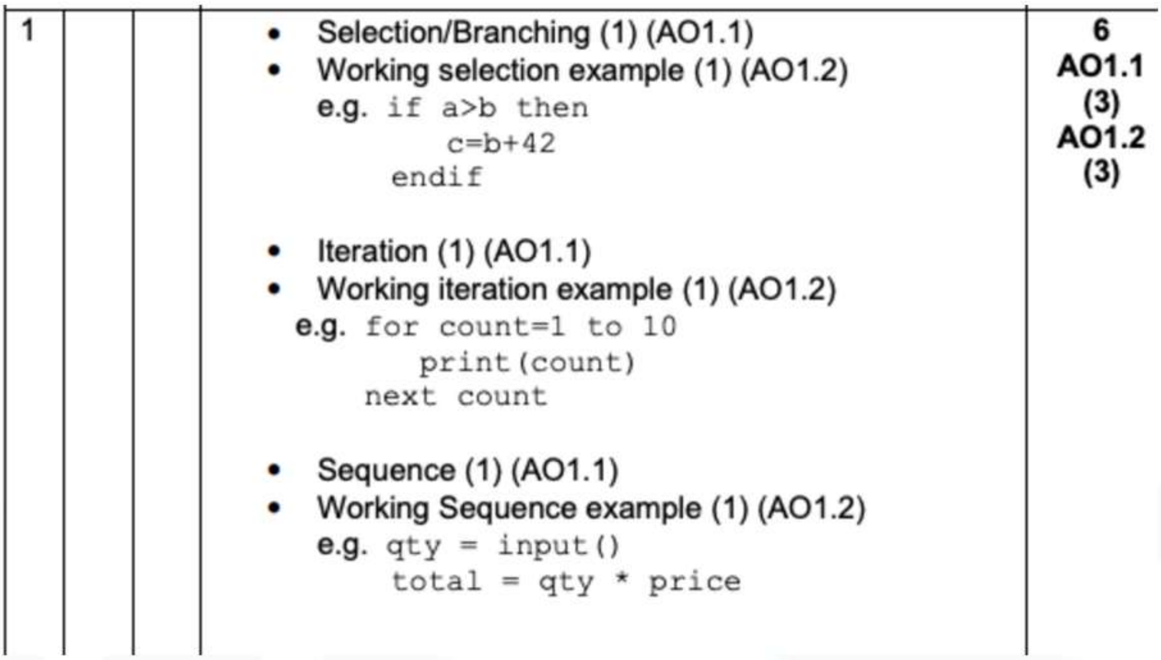
<table>
<tr><td>6</td><td>d</td><td>i</td><td>1 mark per bullet to max 4
• Correct declaration, appropriate name (e.g. <code>new</code>)
• Taking <code>name</code> and <code>theType</code> as a parameter
• Setting <code>petName</code> to parameter
• Setting <code>bored</code>, <code>hunger</code> and <code>intelligence</code> to 0

e.g.<pre>
public procedure new(name, theType)
     petName = name
     bored = 0
     hunger = 0
     intelligence = 0
     type = theType
endprocedure
</pre></td><td>4
AO2.2
(1)
AO3.2
(3)</td><td></td></tr>
<tr><td>6</td><td>d</td><td>ii</td><td>1 mark per bullet to max 2
• <code>myPet</code>/appropriate = <code>new pet</code>
• <code>Springy</code> and <code>Tiger</code>, in <code>""</code>, in same order as constructor declaration

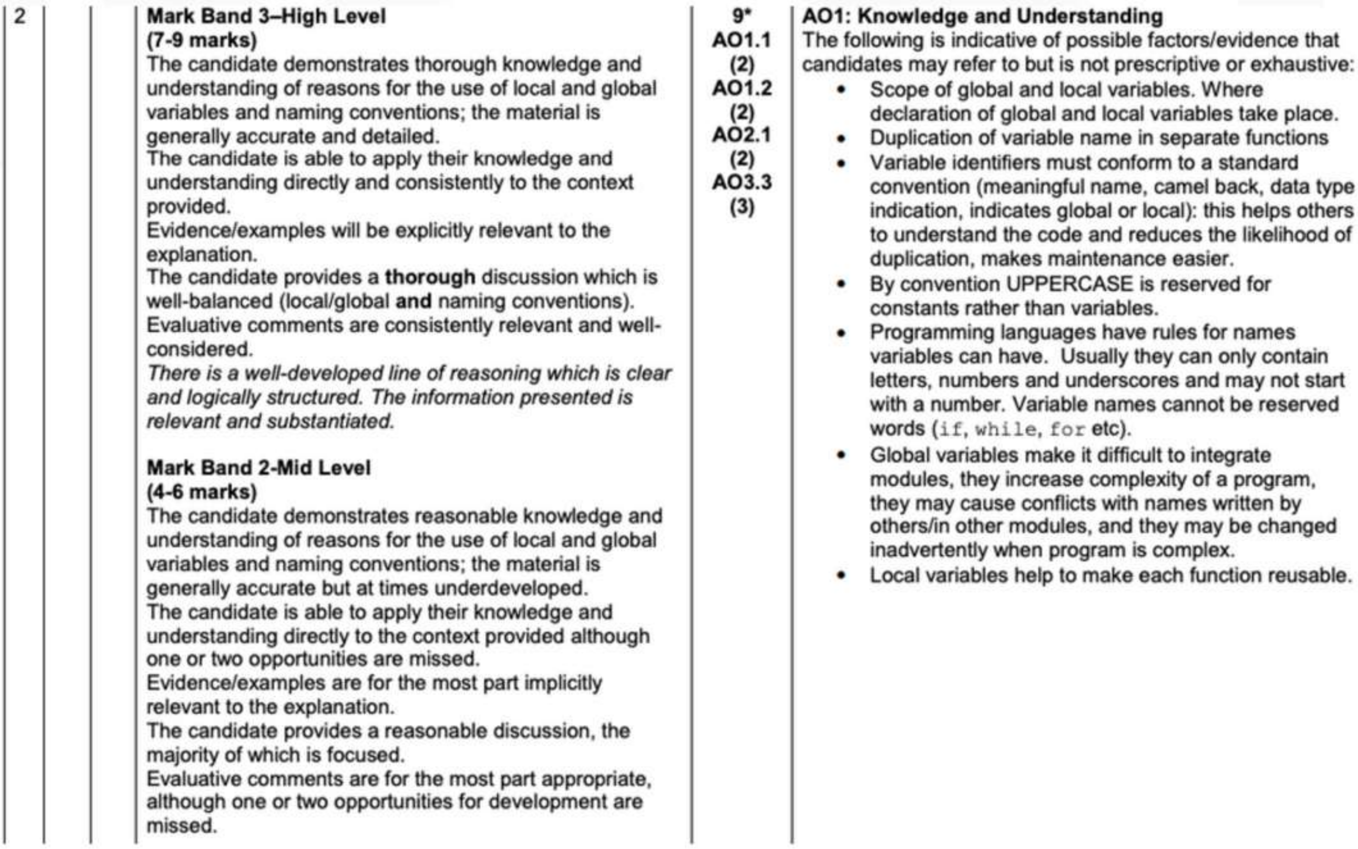
e.g.<pre>
myPet = new pet("Springy", "Tiger")
</pre></td><td>2
AO2.1
(2)</td><td></td></tr>
<tr><td>6</td><td>d</td><td>iii</td><td>1 mark per bullet to max
• Class declaration including inherit (or equivalent e.g. Tiger extends Pet, Tiger::Pet, Tiger(Pet))
• Constructor procedure (new) with all attributes present
    • bored = 10, hunger = 50, intelligence = 10, type = “Tiger”
• outputGreeting procedure
    • Outputting original and new messages correctly

e.g.<pre>
class Tiger inherits Pet
     public procedure new(name)
          petName = name
          bored = 10
          hunger = 50
          intelligence = 10
          type = "Tiger"
     endprocedure

     public procedure outputGreeting()
          print("Hello, I'm " + petName + ", I'm a " + type)
          print("I like to eat meat and roar")
     endprocedure
endclass
</pre></td><td>5
AO2.2
(2)
AO3.2
(3)</td><td>Accep
supe
In pla</td></tr>
</table>



1			• Selection/Branching (1) (AO1.1)
			• Working selection example (1) (AO1.2)
			e.g. `if a>b then`
			`        c=b+42`
			`    endif`
			
			• Iteration (1) (AO1.1)
			• Working iteration example (1) (AO1.2)
			e.g. `for count=1 to 10`
			`        print(count)`
			`    next count`
			
			• Sequence (1) (AO1.1)
			• Working Sequence example (1) (AO1.2)
			e.g. `qty = input()`
			`    total = qty * price`
				6
				AO1.1
				(3)
				AO1.2
				(3)



2			**Mark Band 3–High Level** **(7-9 marks)** The candidate demonstrates thorough knowledge and understanding of reasons for the use of local and global variables and naming conventions; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. The candidate provides a **thorough** discussion which is well-balanced (local/global **and** naming conventions). Evaluative comments are consistently relevant and well-considered. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2-Mid Level** **(4-6 marks)** The candidate demonstrates reasonable knowledge and understanding of reasons for the use of local and global variables and naming conventions; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are for the most part appropriate, although one or two opportunities for development are missed.	**9*** **AO1.1** **(2)** **AO1.2** **(2)** **AO2.1** **(2)** **AO3.3** **(3)**	**AO1: Knowledge and Understanding** The following is indicative of possible factors/evidence that candidates may refer to but is not prescriptive or exhaustive: • Scope of global and local variables. Where declaration of global and local variables take place. • Duplication of variable name in separate functions • Variable identifiers must conform to a standard convention (meaningful name, camel back, data type indication, indicates global or local): this helps others to understand the code and reduces the likelihood of duplication, makes maintenance easier. • By convention UPPERCASE is reserved for constants rather than variables. • Programming languages have rules for names variables can have. Usually they can only contain letters, numbers and underscores and may not start with a number. Variable names cannot be reserved words (`if`, `while`, `for` etc). • Global variables make it difficult to integrate modules, they increase complexity of a program, they may cause conflicts with names written by others/in other modules, and they may be changed inadvertently when program is complex. • Local variables help to make each function reusable.

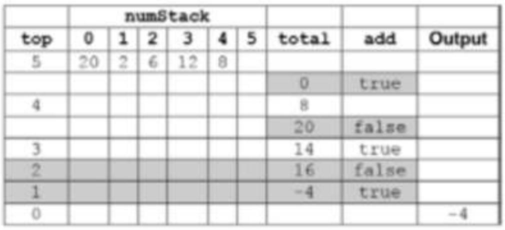
			There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence. **Mark Band 1-Low Level** **(1-3 marks)** The candidate demonstrates a **basic** knowledge of reasons for the use of local and global variables and naming conventions, with **limited** understanding shown; the material is basic and contains some inaccuracies. The candidate makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgments if made are weak and unsubstantiated. *The information is basic and communicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* **0 marks** No attempt to answer the question or response is not worthy of credit.		**AO2.1: Application** The selected knowledge/examples should be directly related to the specific question. The following is indicative of possible factors/evidence that candidates may refer to but is not prescriptive or exhaustive: • Explanation of how the standard rules for programming would impact upon the choices made for using variables and how they are addressed. • Discussion of how breaking the rules of variable naming results in syntax error, causing it not to compile. • Discussion around the use of different variables that are dependent, independent or interdependent. **AO3.3: Evaluation** Candidates will need to consider a variety of viewpoints in relation to following standard rules for functions and variables while developing management software and will make evaluative comments about the issues and solutions they are discussing e.g. • Why when using global variables complexity of program increase? • Why meaningful variable names and camel back are needed? • Why indication of data type and whether local or global • Why local variables allow functions to be reusable? • Why variable rules are important so the tokeniser can recognise variables in lexical analysis. • Why ignoring rules of naming variables can result in unexpected behaviour (e.g. if a language is case sensitive `netPrice` is different to `netprice`)
6	a)		• Read in `A` and `B`. • Correct comparisons • Correct output messages. • Open file • Write to and close file. E.g.	**5** **AO3.1** **(5)**	Max 5 marks Accept open file in append mode

```
A = input(“Enter value A”)
B = input(“Enter value B”)
myFile = openWrite(“output.txt”)
if A < B then
      myFile.writeLine(“A is less than B”)
elseif B < A then
      myFile.writeLine(“B is less than A”)
else
      myFile.writeLine(“A is equal to B”)
endif
myFile.close()
```

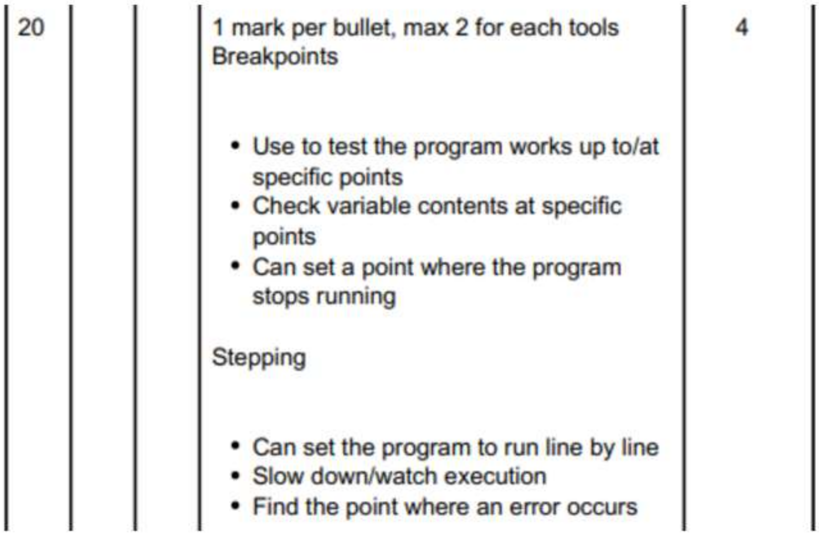
4		i	High-level language / 3GL / imperative language Gives a series of instructions in a (logical) order / line by line / what to do and how to do it	2	**Examiner's Comments** A mixed bag of answers for this question, a good example of candidates not reading the question. About half gave a perfect answer and the other half said something about using procedures and functions or that it used sequence, selection and iteration which was not what was required.
		ii	Declare (result) as a local variable in each procedure Accessible within one procedure (at a time) / the scope of the variable is for one procedure at a time / only exists as long as the procedure is running	2	**Examiner's Comments** Most candidates were able to give a complete answer to this question and it was good to see candidates talking about scope of the variables.
		iii	Parameters passed by value or by reference By value, local copy of data is used then discarded… …so value of (original) data is unchanged By reference, location of data is used… …so changes may be made to value of data	5	**Examiner's Comments** Some candidates knew this and were able to reel off the answers easily, some managed to get half way and gave vague answers on the detail and some missed the point entirely. This question was designed to cover a range of grades and this was demonstrated in the range of answers given.
5		i	• Defined within one module… • … accessible only in that module / Any mention of scope • Can be used as parameters • Data is lost at end of module • Same variable name can be used in other modules without overwriting values/causing errors • Can overwrite global variables (with the same name)	4	For module allow procedure / function / sub routine / block of code **Examiner's Comments** Well answered by most candidates.
		ii	• Defined at start of program • Exists throughout program / in all modules • Allows data to be shared by modules	2	**Examiner's Comments** Nearly all candidates were able to get at least one mark on this.
			Total	**6**	
6		i	• Identifier/name of a … • Memory location used to store data	2	**Examiner's Comments** Generally well answered.
		ii	• A range of statements/procedure/function/method that a variableis valid for • A local variable takes precedence over a global variable of thesame name/allow the same identifier to be used for different purposes without conflict	2	Accept block of code **Examiner's Comments** Most gained a mark, although many vague references to code were given which were rescued by definitions of global and local variables.

7			• Global variables are (usually) defined at the start of a program • Global variables can be seen / used everywhere in the program • Local variables can only be seen / used in a procedure / function / sub routine in which they are declared • Local variables cease to exist once the procedure / function / sub routine they are in is finished • Local variables with the same name as global variables… • …will overwrite / take precedence over the values in the global variable • Local variables within two different procedures will not interfere with one another	6	**For 4th bullet accept construct** **Examiner's Comments** A large majority of very good answers to this question with candidates writing confidently.

14		i	A procedure does not return a value / a function has to return a value	1 AO1.2 (1)	**Examiner's Comments** Many candidates answered well and understood the difference between functions and procedures, knowing that functions have to return a value.
		ii	1 mark per bullet, max 2 for by value, max 2 for by reference by value: • A local copy of the data is used • Data is discarded when the subprogram exits • Does not override/change the original data by reference: • Memory location of data is sent • Changes are made to the original data • Changes remain after the subprogram exits	4 AO1.2 (4)	**Examiner's Comments** Parameter passing by value and by reference continue to prove problematic to candidates, with many having a poor grasp of the concept. Those candidates who have used a variety of programming languages including those that allow for parameter passing by reference often had the practical experience to draw upon.

		iv	1 mark for each bullet to max 6 • Initialising `total` to 0 and `add` to true • `top` = 4 and `total` = 8 • `total` = 20 and `add` = false • `top` = 3, `total` = 14, `add` = true • `top` 2, 1. `Total` 16, -4, `add` false, true • Output = -4 	6 AO1.2 (3) AO2.2 (3)	**Examiner's Comments** Tracing code execution is an area that continues to prove challenging to candidates. Candidates need to have experience of completing dry-runs of code and setting out a trace table in a logical manner. Where candidates did perform well the initialisation of the variables *total* and *add* before the main body of the loop was entered was often omitted.

	numStack								
top	0	1	2	3	4	5	total	add	Output
5	20	2	6	12	8				
							0	true	
4							8		
							20	false	
3							14	true	
2							16	false	
1							-4	true	
0									-4



20			1 mark per bullet, max 2 for each tools Breakpoints - Use to test the program works up to/at specific points - Check variable contents at specific points - Can set a point where the program stops running Stepping - Can set the program to run line by line - Slow down/watch execution - Find the point where an error occurs	4

If you found this useful, drop a follow to help me out!

THANK YOU!

GCST

GCSECOMPUTERSCIENCETUTOR.COM