GCE

Computer Science

H446/02: Algorithms and programming

A Level

Mark Scheme for June 2023

OCR (Oxford Cambridge and RSA) is a leading UK awarding body, providing a wide range of qualifications to meet the needs of candidates of all ages and abilities. OCR qualifications include AS/A Levels, Diplomas, GCSEs, Cambridge Nationals, Cambridge Technicals, Functional Skills, Key Skills, Entry Level qualifications, NVQs and vocational qualifications in areas such as IT, business, languages, teaching/training, administration and secretarial skills.

It is also responsible for developing new specifications to meet national requirements and the needs of students and teachers. OCR is a not-for-profit organisation; any surplus made is invested back into the establishment to help towards the development of qualifications and support, which keep pace with the changing needs of today's society.

This mark scheme is published as an aid to teachers and students, to indicate the requirements of the examination. It shows the basis on which marks were awarded by examiners. It does not indicate the details of the discussions which took place at an examiners' meeting before marking commenced.

All examiners are instructed that alternative correct answers and unexpected approaches in candidates' scripts must be given marks that fairly reflect the relevant knowledge and skills demonstrated.

Mark schemes should be read in conjunction with the published question papers and the report on the examination.

MARKING INSTRUCTIONS

PREPARATION FOR MARKING

1. Make sure that you have accessed and completed the relevant training packages for on-screen marking: *RM assessor Online Training*; *OCR Essential Guide to Marking*.

2. Make sure that you have read and understood the mark scheme and the question paper for this unit. These are posted on the RM Cambridge Assessment Support Portal http://www.rm.com/support/ca

3. Log-in to RM assessor and mark the **required number** of practice responses ("scripts") and the **number of required** standardisation responses.

   YOU MUST MARK 5 PRACTICE AND 10 STANDARDISATION RESPONSES BEFORE YOU CAN BE APPROVED TO MARK LIVE SCRIPTS.

MARKING

1. Mark strictly to the mark scheme.

2. Marks awarded must relate directly to the marking criteria.

3. The schedule of dates is very important. It is essential that you meet the 50% and 100% deadlines. If you experience problems, you must contact your Team Leader (Supervisor) without delay.

4. If you are in any doubt about applying the mark scheme, consult your Team Leader by telephone or the RM assessor messaging system, or by email.

5. **Crossed Out Responses**
   Where a candidate has crossed out a response and provided a clear alternative then the crossed-out response is not marked. Where no alternative response has been provided, examiners may give candidates the benefit of the doubt and mark the crossed-out response where legible.

   Contradictory Responses
   When a candidate provides contradictory responses, then no mark should be awarded, even if one of the answers is correct.

   Short Answer Questions (requiring only a list by way of a response, usually worth only **one mark per response**)
   Where candidates are required to provide a set number of short answer responses then only the set number of responses should be marked. The response space should be marked from left to right on each line and then line by line until the required number of responses have been considered. The remaining responses should not then be marked. Examiners will have to apply judgement as to whether a 'second response' on a line is a development

of the 'first response', rather than a separate, discrete response. *(The underlying assumption is that the candidate is attempting to hedge their bets and therefore getting undue benefit rather than engaging with the question and giving the most relevant/correct responses.)*

Short Answer Questions (requiring a more developed response, worth **two or more marks**)
If the candidates are required to provide a description of, say, three items or factors and four items or factors are provided, then mark on a similar basis – that is downwards (as it is unlikely in this situation that a candidate will provide more than one response in each section of the response space.)

Longer Answer Questions (requiring a developed response)
Where candidates have provided two (or more) responses to a medium or high tariff question which only required a single (developed) response and not crossed out the first response, then only the first response should be marked. Examiners will need to apply professional judgement as to whether the second (or a subsequent) response is a 'new start' or simply a poorly expressed continuation of the first response.

6. Always check the pages (and additional objects if present) at the end of the response in case any answers have been continued there. If the candidate has continued an answer there then add a tick to confirm that the work has been seen.

7. Award No Response (NR) if:
   - there is nothing written in the answer space

   Award Zero '0' if:

   - anything is written in the answer space and is not worthy of credit (this includes text and symbols).

   Team Leaders must confirm the correct use of the NR button with their markers before live marking commences and should check this when reviewing scripts.

8. The RM assessor **comments box** is used by your team leader to explain the marking of the practice responses. Please refer to these comments when checking your practice responses. **Do not use the comments box for any other reason.**
   If you have any questions or comments for your team leader, use the phone, the RM assessor messaging system, or e-mail.

9. Assistant Examiners will send a brief report on the performance of candidates to their Team Leader (Supervisor) via email by the end of the marking period. The report should contain notes on particular strengths displayed as well as common errors or weaknesses. Constructive criticism of the question paper/mark scheme is also appreciated.

10. For answers marked by levels of response:
    a. **To determine the level** – start at the highest level and work down until you reach the level that matches the answer
    b. **To determine the mark within the level**, consider the following:

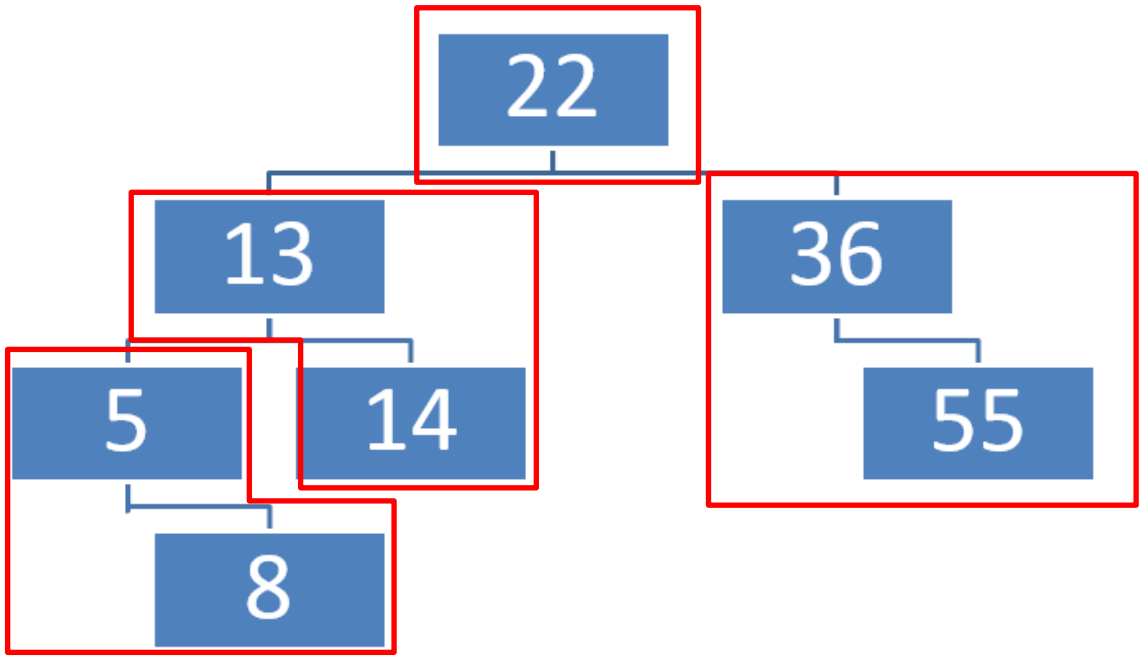
Descriptor	Award mark
On the borderline of this level and the one below	At bottom of level
Just enough achievement on balance for this level	Above bottom and either below middle or at middle of level (depending on number of marks available)
Meets the criteria but with some slight inconsistency	Above middle and either below top of level or at middle of level (depending on number of marks available)
Consistently meets the criteria for this level	At top of level

11. Annotations

Annotation	Meaning
	Omission mark
	Benefit of the doubt
	Incorrect point
	Follow through
	Not answered question
	No benefit of doubt given
	Repeat
	Correct point
	Too vague
	Zero (big)
	Blank Page – this annotation must be used on all blank pages within an answer booklet (structured or unstructured) and on each page of an additional object where there is no candidate response.
	Level 1
	Level 2
	Level 3

12. Subject Specific Marking Instructions

Question			Answer	Mark	Guidance
1	(a)	(i)	1 mark each to max 2 • It is a hierarchical structure / not directed • Data is stored in nodes • Nodes are linked by branches/edges • It has a root node • Each node has zero or more nodes 'beneath' it // nodes can link to child nodes • It has leaf nodes / nodes without any lower nodes are leaf nodes • It has no cycles/loops (distinguishing it from a graph)	2	
1	(a)	(ii)	1 mark each • Root node 22 at the start • 13 and 14 in correct order • 5 and 8 in correct order • 36 and 55 in correct order	4	Do not allow nodes to be drawn downwards.

					
1	**(a)**	**(iii)**	1 mark each • **Search/traverse tree** until the required node is found • Set the parent node pointer to the leaf node to null • Add the deleted node to the free storage list // leave for garbage clear up	2	
1	**(a)**	**(iv)**	1 mark each to max 4 • Check if the root node **is equal** to search value and if so…. • …return/output/report found • If value **is less** than root node take left subtree • If value **is greater** than root node take right subtree • Repeat process with the subtree… • …until search value is found • …until no more branches can be travelled.	4	

<table>
<tr><td>1</td><td>(a)</td><td>(v)</td><td>1 mark each
• Visiting A first…
• …Then visiting F, C…
• …Then visiting L, T, P…
• …Visiting H last

Solution: A, F,C, L,T,P, H</td><td>4</td><td></td></tr>
<tr><td>1</td><td>(a)</td><td>(vi)</td><td>1 mark each to max 2
• When a leaf node is reached…
• …the traversal backtracks to the leaf’s parent node
• …backtracks to last node with unvisited children</td><td>2</td><td>Candidates may use an example from the tree in 1a(v) to illustrate their response.

If an answer gives implementational detail of how a stack is used, map to the bullet points given.</td></tr>
<tr><td>1</td><td>(b)</td><td></td><td>1 mark for final path A, D, G
1 mark for final distance 14
1 mark for each SECTION or equivalent working shown.
<table>
<tr><th>Node</th><th>Distance travelled</th><th>Previous node</th><th>Marking Guidance</th></tr>
<tr><td>A</td><td>0</td><td>- / N/A / blank / None</td><td rowspan="2">1 Mark</td></tr>
<tr><td>B</td><td>5</td><td>A</td></tr>
<tr><td>C</td><td>2</td><td>A</td><td rowspan="2">1Mark</td></tr>
<tr><td>D</td><td>10</td><td>A</td></tr>
<tr><td>E</td><td>7</td><td>B</td><td rowspan="2">1 Mark</td></tr>
<tr><td>F</td><td>15</td><td>E</td></tr>
<tr><td>G</td><td><s>19</s> 14</td><td><s>E</s> D</td><td>1 Mark</td></tr>
</table></td><td>6</td><td>Nodes should appear in the alphabetical order given if candidates add them as the algorithm progresses but allow other orderings of the nodes.

For the last mark in the table there must be a clear indication that G 19 from E is overwritten by G 14 from D.

Allow equivalent discrete maths approach or textual description.

Check diagram for annotations / solution.</td></tr>
</table>

Question	Answer	Mark	Guidance
2	**Mark Band 3 – High level** **(7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of both computational thinking methods; 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 both computational thinking methods; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.* **Mark Band 1 – Low Level** **(1-3 marks)** The candidate demonstrates a basic knowledge of both computational thinking methods 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.	9	**AO1: Knowledge and Understanding** **Indicative content** • Problem recognition is identifying that there is a problem to be solved, to determine exactly what the problem is from a description/scenario and to determine if the problem can be solved with computational methods • Decomposition is splitting the problem down into subproblems that can be solved independently **AO2: Application** • Problem recognition: identifying the need for the scheduling system, what it will take as its inputs, what will need to be output etc. • Decomposition: subproblems could include: o inputting the requirements o generating possible routes o evaluating the routes o outputting the schedule **AO3: Evaluation** Computational methods allow the e.g.: • programmer to determine what the problem, what the challenges may be and what additional information is required before starting to code the solution • identification of the key features for programmers to focus on • splitting of the task into smaller, more manageable/solvable problems which allows for a solution to be developed quicker • design of an effective/efficient solution that makes best use of a processor • splitting of a task to allow programmers to focus on areas they specialise in.

	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.		

3	**(a)**		1 mark each • `headPointer`: To indicate the first element in the list • `freeListPointer`: To indicate the next index to store data in (the freeList)	2	
3	**(b)**		It doesn't point to another **node** Indicates the end of the linked list	1	
3	**(c)**		• first output red… • …remainder of list correct e.g. red blue grey green purple orange	2	

3	(d)		1 mark each to max 4 Check space available in the free list • Check to make sure `freeListPointer` is not Null Add new data item to first free space in free list • Insert new data item at index `freeListPointer` (index 4) Append e.g. • Traverse to / locate the end of the list (index 3 'orange') • Set the pointer of the last item in the linked list to `freeListPointer` (pointer at index 3 'orange' changes from Null to 4)… • … update `freeListPointer` to the location that new data item pointer is pointing to at present. (`freeListPointer` changes from 4 to 5) • … update pointer from new data item to Null (index 4 pointer changes from 5 to Null) Prepend e.g. • Update `freeListPointer` to point to the location that the pointer from the first item in the free list is pointing to (`freeListPointer` changes from 4 to 5) • … Update pointer from new data item to `headPointer` (index 4 pointer changes from 5 to 1) • … Update `headPointer` to the index of new data item (`headPointer` changes from 1 to 4)	4	Note descriptions could be for either appending an item to the end of the current list or prepending it to the start. There are different ways to achieve this. Allow answers that illustrate solutions by example from the table in Fig 3 at the start of the question. Reponses must refer to the relevant pointers or give clear exemplifications.

Question			Answer	Mark	Guidance

3	(e)		1 mark for each statement
			
			`function findNode(toFind, headPointer, linkedList)`
			`  currentNode = headPointer`
			`  while(currentNode != NULL)`
			`  if linkedList[currentNode].data == toFind then`
			`  return currentNode`
			`  else`
			`  currentNode = linkedList[currentNode].pointer`
			`  endif`
			`  endwhile`
			`  return -1`
			`endfunction`
				5	Ignore case of identifiers in pseudocode
					
					Only penalise excessive spaces within identifier names if obvious.

Question			Answer	Mark	Guidance
4	(a)	(i)	1 mark for: • `isInteger` • `number` • `result` • `count` • `asciiValue`	1	Penalise excessive spaces in identifiers such as *ascii Value* instead of *asciiValue*
4	(a)	(ii)	(0)5	1	
4	(a)	(iii)	(0)3	1	
4	(b)		1 mark each 03 • Loop through each of the **characters**/**digits** in the `number` string (parameter) 04 • Find the ASCII value of the current **character/digit** 09 • Return true if the value is an integer and false otherwise	3	
4	(c)		1 mark each to max 2: • One piece of code can be used many times / in multiple places / makes code more efficient • No need to write the same code multiple times • Takes less time to plan/design/code the program • Easier error detection as fix once and it corrects in each place // less likely to have errors as code is not written multiple times • Makes it easier to maintain the program	2	

<table>
<tr>
<td>5</td>
<td>(a)</td>
<td></td>
<td>1 mark each to max 3
• The function calls itself….
• …..such as line 05 / 07
• Each recursive call will create a new copy of the values in the function….
• ….and add all of the values of the copy the call is being made from to a stack
• There is a base case // condition that stops the recursive calls…
• …condition in line 02
• There may be more than one base case</td>
<td>3</td>
<td>Allow answers in context as long as they are clear what the features are.</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
• First call with 10 and second call with 7
• Remainder of calls 6, 3, 2
• Final call value -1
• Adding/showing return values (1 + 2 + 3 + 6 + 7 + 10)

e.g.

<table>
<tr><th>Function call</th><th>value</th><th>return</th></tr>
<tr><td>recursiveAlgorithm(10)</td><td>10</td><td>29</td></tr>
<tr><td>recursiveAlgorithm(7)</td><td>7</td><td>19</td></tr>
<tr><td>recursiveAlgorithm(6)</td><td>6</td><td>12</td></tr>
<tr><td>recursiveAlgorithm(3)</td><td>3</td><td>6</td></tr>
<tr><td>recursiveAlgorithm(2)</td><td>2</td><td>3</td></tr>
<tr><td>recursiveAlgorithm(-1)</td><td>-1</td><td>1</td></tr>
<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>
</table>

6			1 mark each to max 6
			• Taking number as input
			• Calculating remainder after division by 8
			• Calculating integer after division by 8
			• Correct loop until 0 is reached (or equivalent method)
			• Concatenating each remainder // storing each remainder in an array/list
			• Outputting the correct result
			
			e.g. pseudocode
			`number = input("Enter a number")`
			
			`endResult = ""`
			
			`while number != 0`
			`  remainder = number MOD 8`
			`  number = number DIV 8`
			`  endResult = str(remainder) + str(endResult)`
			`endwhile`
			`print endResult`
				6	Note candidates can reverse the string before output if they don't concatenate in the order given in the example.
					E.g.
					`endResult = str(endResult) + str(remainder)`
					
					The final markpoint can only be awarded where the correct output will be produced by the algorithm.

7	(a)		**Mark Band 3 – High level** **(7-8 marks)** The candidate demonstrates a thorough knowledge and understanding of Big O; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level** **(4-6 marks)** The candidate demonstrates reasonable knowledge and understanding of Big O; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.* **Mark Band 1 – Low Level** **(1-3 marks)** The candidate demonstrates a basic knowledge of Big O with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgements if made are weak and unsubstantiated. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* **0 marks** No attempt to answer the question or response is not worthy of credit.	9	**AO1: Knowledge and Understanding** **Indicative content** • Big O measures the number of steps and memory usage change according to the data as the amount of data being processed increases • Linear - grows in proportion to amount of data • Exponential – the rate of increase is at the rate k^n as n increases • Constant - it does not change • Logarithmic – means the rate of increase gets smaller as the amount of data increases time / time increases at a rate of $\log_k n$ as n increases. **AO2: Application** • Algorithm 1 – The time taken increases as the data set grows. The space taken also significantly increases. This algorithm is not memory efficient. • Algorithm 2 – The time increases significantly and therefore this algorithm is not time efficient. The space will never change which means the amount of memory will not change as the data set grows. • Algorithm 3 – The time will grow less fast as the data set grows relative to the other algorithms. The space required will also increase, but not insurmountably. This is therefore an efficient

					algorithm with large data sets compared to algorithm 1 and 2 overall. **AO3: Evaluation** • Number of elements is unknown. Exponential is least appropriate because this could increase significantly and be unmanageable. • Constant is the most ideal as the time will not increase. • Algorithm 3 is more suitable because it has a logarithmic time complexity, so it increases less quickly than the other algorithms. It will be reasonable with a small amount (2 items) of data, but then when very large amounts (2 billion items) are needed it will not be significantly more.
7	**(b)**	**(i)**	1 mark for each to max 2 • Processes happen at the same time // processes overlap • One process can start before another one finishes • Each process is given a slice of processor time • Different processes can be executed (in parallel) by different processors/cores	2	
7	**(b)**	**(ii)**	1 mark each to max 2 e.g. • More efficient processor use // Less idle time for processor // Greater throughput • Long running tasks do not delay short running tasks • Tasks requiring preconditions can wait and then resume execution • User is able to interact with the computer while other tasks are running	2	

7	**(c)**	**(i)**	1 mark each to max 5 • The data list is split into two lists • These sublists continue to be (recursively) split… • …until each sublist is one item • The first element in two different sublists is compared… • …the smaller item is then selected… • …and written to a new list • …until both sublists fully merged • Repeated until all sorted sublists are recombined	5	Allow array/list as equivalent
7	**(c)**	**(ii)**	1 mark for benefit e.g. • More efficient time complexity (for large data sets) // takes fewer steps to sort the data • Time complexity O(n log n), rather than $O(n^2)$ • Uses divide and conquer • Can apply concurrent processing to reduce sorting time 1 mark for drawback e.g. • More difficult to implement // needs more complex code • Less efficient space complexity // uses more memory with more data items • Space complexity of O(n)/linear, rather than O(1) / constant • Merge sort is always $O(n\log_2 n)$ whereas the best case for bubble sort is O(n)	2	

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.

8	**(a)**		1 mark for each input to max 2 • Username • Password 1 mark for output e.g. • Message to request input • Message to state login successful • Message to say login unsuccessful	3	
8	**(b)**		1 mark each to max 2 e.g. • Connect to database • Access usernames in file/database • Check username against file/database • Hash password • Access password/hash in file/database • Check password entered/hashed vs stored • Output result	2	Allow other suitable subprocedures that link to the scenario.

Question			Answer	Mark	Guidance
9	(a)	(i)	1 mark for each description to max 2 and 1 mark for example e.g. • Removal of unnecessary detail… • ….to allow programmers to focus on core aspects of the problem…. • ….simplifies a complex problem Examples, e.g: • Treasure objects are replaced with text labels // no images of treasure are used • Island is set of coordinates and no info as to environment/layout and other objects	3	Allow other suitable examples that are relevant to the treasure game.
9	(a)	(ii)	1 mark each to max 3 e.g. • Reduces programming time • Reduces complexity of code (through abstraction by generalisation) • Reduces amount of memory required / computational power • Simplifies the problem so it's easier to solve / understand (by recognising common patterns) • Allows programmers to focus on core aspects of the problem	3	

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	

9	**(d)**		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 `getGridItem`
			• Checking (treasure) object's level/value…
			• …using correct get method `getLevel` `getValue`
			• …outputting "No treasure" if empty
			• …otherwise outputting value and level
			
			e.g.
			`procedure guessGrid(gameboard)`
			`  rCoord = input("Enter R coordinate")`
			`  cCoord = input("Enter C coordinate")`
			`  treasureItem = gameBoard.getGridItem(rCoord, cCoord)`
			`  if treasureItem.getLevel() = "" then`
			`  print("No treasure")`
			`  else`
			`  print("This treasure is level ", treasureItem.getLevel(), " with value ", treasureItem.getValue())`
			`endprocedure`
				7	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.

9	**(e)**		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.	4	1 mark per benefit identified and 1 mark per expansion. Max 2 benefits and 1 expansion per benefit.

9	(f)		**Mark Band 3 – High level** **(7-9 marks)** 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. *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 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. *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 parameters and local/global variables with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a	9	**AO1: Knowledge and Understanding** **Indicative content** • 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 **AO2: Application** • 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 **AO3: Evaluation** • 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

			limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgements if made are weak and unsubstantiated. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* **0 marks** No attempt to answer the question or response is not worthy of credit.		actual board if needed, for example to simulate potential changes.

Need to get in touch?

If you ever have any questions about OCR qualifications or services (including administration, logistics and teaching) please feel free to get in touch with our customer support centre.

Call us on

01223 553998

Alternatively, you can email us on

support@ocr.org.uk

For more information visit

ocr.org.uk/qualifications/resource-finder

ocr.org.uk

Twitter/ocrexams

/ocrexams

/company/ocr

/ocrexams

CAMBRIDGE
UNIVERSITY PRESS & ASSESSMENT

OCR is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

OCR operates academic and vocational qualifications regulated by Ofqual, Qualifications Wales and CCEA as listed in their qualifications registers including A Levels, GCSEs, Cambridge Technicals and Cambridge Nationals.

OCR provides resources to help you deliver our qualifications. These resources do not represent any particular teaching method we expect you to use. We update our resources regularly and aim to make sure content is accurate but please check the OCR website so that you have the most up-to-date version. OCR cannot be held responsible for any errors or omissions in these resources.

Though we make every effort to check our resources, there may be contradictions between published support and the specification, so it is important that you always use information in the latest specification. We indicate any specification changes within the document itself, change the version number and provide a summary of the changes. If you do notice a discrepancy between the specification and a resource, please contact us.

Whether you already offer OCR qualifications, are new to OCR or are thinking about switching, you can request more information using our Expression of Interest form.

Please get in touch if you want to discuss the accessibility of resources we offer to support you in delivering our qualifications.