Mark scheme

Question			Answer/Indicative content	Marks	Guidance
1			1 mark for condition and 1 mark for outcome to max 4 e.g. • **Condition:** Check if the square has an instruction ... • **Outcome:** ... move the player the number of places specified • **Condition:** Check if they have landed on square 100 ... • **Outcome:** ... Announce the player as the winner • **Condition:** Check if the timer is 0 ... • **Outcome:** ... Announce the game as a draw	4	Allow other suitable examples that are relevant to the scenario. The outcome must be related to the condition for a mark to be awarded. **Examiner's Comments** Some candidates did not present clear conditions but gave descriptions of actions to perform within the game such as 'roll dice'. The condition required an identification within the context of something that would lead to a Boolean result with consequent outcome. For example, *if* the timer runs out *then* the game will end in a draw.
			Total	**4**	
2	a		1 mark each to max 3. Max 2 for generic answers with no relation to scenario. e.g. • Has a set/fixed number of values • …and the number of spaces in the road will not change • Stores data of one type • … as the array is only made up of prize objects • Stores data linearly • … match the linear nature of the road • Array contents are mutable • … so prizes can be added/removed from the road • A single indentifier is used to directly index • … any position in the road • Can be iterated by index • … to perform an operation on all road positions	3	**Examiner's Comments** Many responses were too vague, showing little knowledge of the properties of arrays. Relatively few candidates appeared to be able to make explicit links to the scenario to achieve full marks.

	b	i	1 mark each
			• Function/subroutine with identifier `getName` taking **no** parameters
			• Returning `name`
			
			e.g.
			`public function getName()`
			`return name`
			`endfunction`
			
			`public getName(){`
			`return name`
			`}`
			
			`def getName(self):`
			`return self.__name`
			
			`function getName() {`
			`return this.name`
			`}`
				2	BP1 Do not award procedure or method
					
					BP1 Allow self as an additional parameter if Python is used.
					
					BP1 If an access modifier is given for the method, it must be public and not private.
					
					BP2 Do not allow any modified name attribute to be returned.
					
					Examiner's Comments
					
					While many candidates had little difficulty giving code for a `getter()` there were a number of common errors. Some candidates used a private access modifier when a `getter()` needs to be public. There was often erroneous use of 'procedure' whereas a `getter()` is a function that must return a value.
					Some candidates tried to set values within the `getter()` function when it should only have returned the class attribute value.

		ii	1 mark each
			• New instance of `prize` …
			• … with "Box", "money" and 25 as parameters
			• Assigned to `allPrizes` index 3
			
			e.g.
			`allPrizes[3] = new prize("Box", "money", 25)`
			`allPrizes[3] = prize.new("Box", "money", 25)`
			`allPrizes[3] = prize("Box", "money", 25)`
				3	MP2 allow any order of parameters
					
					"`Box`" and "`Money`" must be strings and `25` must be an integer
					
					Allow `prize.new()` as new is given as the constructor method in the class diagram
					
					Examiner's Comments
					
					Many candidates struggled with the instantiation of an object. Where candidates made an attempt to instantiate some did not use a string for "box" and "money" or did not give 25 as an integer but instead gave the string "25".

		iii	1 mark for each bullet to maximum 3 e.g. • **Decision** - check whether the space already has a prize allocated …	3	Give: • 1 mark for stating a decision • 1 mark for the action required if true

			• **Action if true** - another space/number will need to be generated • **Action if false** - the prize will be stored here • **Decision** - check if all 10 prizes have been allocated … • **Action if true** - the algorithm needs to stop generating numbers • **Action if false** - a new number/space needs to be generated and checked		• 1 mark for the action required if false **Examiner's Comments** There were only two reasonable decisions that could be given from the scenario details. Candidates needed to make it clear that a decision with a Boolean output was present that would dictate two potential outcomes. Some candidates quoted actions such as 'randomly assign space for prize' which did not represent a decision. Many responses described the mechanics of setting up the game and the random spaces but did not highlight the program conditions/decisions as required.

	c	i	1 mark each
			
			• Constructor header (any suitable name e.g. new, constructor, create, init)
			• …taking **one** parameter only
			• Initialising name to the parameter
			• Initialising money to 5
			• Initialising experience to 0 and roadPosition to 0
			
			e.g.
			
			Pseudocode example:
			`public procedure new(pName)`
			`name = pName`
			`experience = 0`
			`roadPosition = 0`
			`money = 5`
			`endprocedure`
			
			Python Example:
			`def __init__(self, pName):`
			`self.__name = pName`
			`self.__experience = 0`
			`self.__roadPosition = 0`
			`self.__money = 5`
			
			C# Example:
			`public Character(string pName) {`
			`string name = pName;`
			`int experience = 0;`
				5	Allow minor changes to identifiers as long as purpose is clear.
					
					Allow
					`procedure new(pName)`
					`this.name = pName`
					…
					(or similar e.g. self.name)
					
					Allow two parameters if one is *self* and the response is clearly in Python.
					
					The parameter name should be different to the attribute name.
					
					Examiner's Comments
					
					It was clear that those candidates with limited OOP programming knowledge found the writing of a relatively simple constructor method difficult. Those with relevant programming experience often found this to be a very straightforward question. Common errors included passing additional values to set the `experience`, `roadPosition` and `money` attributes rather than setting them to

			`int roadPosition = 0;`
			`int money = 5;`
			`}`
					the constant values indicated in the question.

		ii	1 mark each
			
			• Procedure/method header …
			• … taking **two** parameters, type (or similar) followed by value (or similar) …
			• … compare type parameter with "`money`"
			• … compare type parameter with "`experience`"
			• … both attributes updated correctly and nothing else modified
			
			e.g.
			`public procedure updateValues(pType, pValue)`
			`if pType == "money" then`
			`money = money + pValue`
			`elseif pType == "experience"`
			`experience = experience + pValue`
			`endif`
			`endprocedure`
			
			`def updateValues(self, pType, pValue):`
			`if pType == "money":`
			`money += pValue`
			`elif pType == "experience":`
			`experience += pValue`
				5	Do not allow Function for BP1
					
					BP2 parameters must be given in the correct order to match the calls to `updateValues()` in the question.
					
					"`money`" and "`experience`" must be string values
					
					Examiner's Comments
					
					The `updateValues` procedure again proved problematic for candidates with limited OOP experience. No marks were given for the first mark point if a function was declared as there was no return value. Parameter names needed to be fit for purpose, understandable, and had to match the order given in the question scenario to work for the given example calls.

	d		1 mark for each completed space
			
			`character1 = new Character("Jamal")`
			`newPosition = 0`
			`while newPosition < 50`
			`move = random(1, 4)`
			`character1.changePosition(move)`
			`newPosition = character1.getRoadPosition()`
			`if newPosition < 50 and road[newPosition] != null then`
			`prizeType = road[newPosition].getType()`
			`valueAmount = road[newPosition].getValue()`
			`character1.updateValues(prizeType, valueAmount)`
				6	Allow `road.length / len(road)` instead of `50`
					
					Allow `<=49` instead of `< 50`
					
					Examiner's Comments
					
					Nearly all candidates achieved some marks, and a majority scored five or six marks.

			`print("Congratulations you are in position", newPosition, "and found", road[newPosition].getName())`
			`print("Money", character1.getMoney(), "and experience", character1.getExperience())`
			`endif`
			`endwhile`
			`print("You reached the end of the road")`		

	e		1 mark each • (Line 02) `for x = 0 to 49` • (Line 03) `print("Space", x)` • (Line 06) `else / elseif road[x] <> null` • (Line 07) `print(road[x].getName())`	4	Line `07` allow `print(road[x].name)` **Examiner's Comments** Many candidates scored three or four marks but in general candidates found it harder to identify errors in the code than to complete code in the previous question. Some candidates didn't give the line number but rewrote the incorrect line before giving the corrected line, which was acceptable, although not ideal given the scaffolding.
	f		**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of global variables and the alternatives; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. *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 global variables and the alternatives; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the	9	**AO1: Knowledge and Understanding** **Indicative content** • Global variables are created when the program starts, all subroutines can access/update the contents • Local variables are created in the subroutine they are created in, they are not accessible directly from any other subroutine • Local variables are removed from memory when the subroutine ends. • Local variables can be passed as parameters to a function to be updated, and then returned to override the original local variable • Local variables can be passed by reference to a subroutine to allow 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 global variables and the alternatives with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgements if made are weak and unsubstantiated. *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 mark** No attempt to answer the question or response is not worthy of credit.		content of the variable to be updated **AO2: Application** • The variables will be stored in memory throughout the whole code execution. However, the amount of data they are storing is relatively low so would not use a lot of memory. • When the game is expanded, the amount of data may increase so it could be memory intensive, especially if graphics are used in the game. • Both arrays are needed throughout the whole game so keeping them as global will make writing the code easier as the programmer will not need to keep passing them as parameters and setting return values. • Only one part of the game is being created initially and therefore the use of global variables would not affect the efficiency greatly. However, when the program expands, it could cause accuracy / testing / debugging and maintenance problems. **AO3: Evaluation** • As this is only a prototype, the use of global variables would be beneficial. • However, when the game expands, the use of global variables could create issues such as running out of memory, coupling, testing & debugging problems and maintenance problems. • The programmer may be best to keep the variables as local and then pass them

					between the different subroutines as parameters byVal and byRef. **Examiner's Comments** Most responses were Level 2 for definitions and some expansion to passing parameters. Very few candidates were able to go into depth about alternatives to global variables such as passing by value and passing by reference in detail or extending to issues such as scalability within a larger more extended game. Few candidates picked up on the fact that this was a more limited prototype that was likely to be expanded on which would require more consideration to be given to variable scope.
			Total	**40**	
3	a	i	1 mark for: • `isInteger` • `number` • `result` • `count` • `asciiValue`	1	Penalise excessive spaces in identifiers such as *ascii Value* instead of *asciiValue* **Examiner's Comments** This question was generally well done (although slightly less well done than parts (a) (ii) and (a) (iii)), but many candidates did very well. The most common erroneous responses were giving the names of predefined functions/properties or giving relational operators.
		ii	(0)5	1	**Examiner's Comments** This question required an exact answer only and was answered correctly by the majority of candidates.
		iii	(0)3	1	**Examiner's Comments** This question also required an exact answer only and was answered correctly by the majority of candidates.

	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	**Examiner's Comments** This question required the *purpose* of the lines of code to be described, but many candidates just described the functionally of the lines of code such as 'line 03 is a counter controlled loop from `0 to number.length -1`'. The expected purpose of this line of code was to set up a loop to iterate through each character in the input string parameter. While many candidates could describe the purpose of at least one of the lines of code given, few could clearly describe the purpose of all three lines.
			Total	**6**	
4			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. **Function call** | **value** | **return** `recursiveAlgorithm(10)` | 10 | 29 `recursiveAlgorithm(7)` | 7 | 19 `recursiveAlgorithm(6)` | 6 | 12 `recursiveAlgorithm(3)` | 3 | 6 `recursiveAlgorithm(2)` | 2 | 3 `recursiveAlgorithm(-1)` | -1 | 1	5	The table is given as guidance, but actual process may be presented in different ways. **Examiner's Comments** While many candidates continue to find recursion a challenging topic there were many who encouragingly achieved full marks. Weaker candidates traced the initial sequence of calls but found it harder to identify the last call to `recursiveAlgorithm(-1)` that triggered the base case and then found it harder again to calculate the unwind sequence.
			Total	**5**	
5		i	355	1	**Examiner's Comments** Many candidates did not appreciate that a bubble sort will require a

					maximum of n -1 passes in the worst case since the first item in the list will be in position after that number of passes and so would not require an additional pass. The most common incorrect responses were 356, and 3562 which confused the worst case time complexity $O(n^2)$ with the number of passes.
		ii	Insertion sort	1	Accept any valid sorting algorithm e.g. Merge sort, Quick sort **Examiner's Comments** Nearly all candidates could identify an additional sorting algorithm with the most common response being 'Insertion sort'.
			Total	**2**	
6		i	1 mark per bullet e.g. • stage (e.g. stage 1, stage 2, stage 3) • city name (e.g. London) • speed (e.g. slow, normal, fast)	2	**Examiner's Comments** Most candidates successfully identified the relevant potential inputs of city/stage/speed from the given scenario. Some candidates suggested alternative inputs such as 'magnitude of earthquake' which were also given marks as they were valid inputs for a simulation relating to earthquakes.
		ii	1 mark per bullet to max 2, e.g. • Does the build-up stage need to be shown? • Does the earthquake taking place needs to be shown? • Does the aftershock stage needs to be shown?	2	Allow other suitable examples **Examiner's Comments** Few candidates were given full marks for this question. There was some repetition of checking user input values, which was given in the stem of the question. There were also many statements of possible calculations rather than clearly expressed conditions or questions. More successful responses included decisions such as 'have buildings of a certain typ e survived the earthquake?'.
			Total	**4**	

<table>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>1 mark per bullet

• 1st swap of 5 and 3
• Remainder of first pass
• Pass 2
• Pass 3

<table>
<tr><td>1</td><td>5</td><td>3</td><td>9</td><td>2</td><td>7</td><td></td></tr>
<tr><td>1</td><td>3</td><td>5</td><td>9</td><td>2</td><td>7</td><td></td></tr>
<tr><td>1</td><td>3</td><td>5</td><td>2</td><td>9</td><td>7</td><td></td></tr>
<tr><td>1</td><td>3</td><td>5</td><td>2</td><td>7</td><td>9</td><td>End of pass 1</td></tr>
<tr><td>1</td><td>3</td><td>2</td><td>5</td><td>7</td><td>9</td><td>End of pass 2</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>5</td><td>7</td><td>9</td><td>End of pass 3</td></tr>
</table>
</td>
<td>4</td>
<td>Candidates do not need to show each swap, so if the candidate has clearly shown the end of pass 1, they have met the first two marking points.

Marks can be awarded for correctly showing the results of each pass.

<u>Examiner’s Comments</u>

Most candidates clearly showed the steps that would take place in a bubble sort for the date given, with most achieving full marks. Some candidates did not explicitly label each pass as required in the question, but marks were given where the passes could be implied.

Few candidates described the principles of a bubble sort instead of applying it to the data given.</td>
</tr>
<tr>
<td></td><td></td><td></td><td>Total</td><td>4</td><td></td>
</tr>
<tr>
<td>8</td><td>a</td><td></td><td>1 mark per bullet up to a maximum of 2 marks, e.g.:

• = is used as the <code>number</code> variable is being assigned a value…
• …== is not used as the program is not checking if the variable is equal to a value</td><td>2
(AO1.2)
(2)</td><td>Accept:

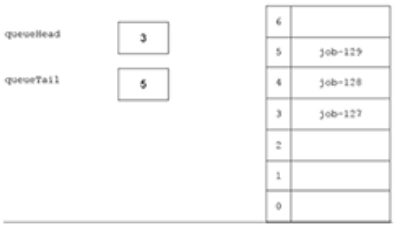
• = is an assignment operator
• == is a comparison operator.

<u>Examiner’s Comments</u>
Most candidates were correctly able to distinguish between assignment and equality.</td>
</tr>
<tr>
<td></td><td>b</td><td></td><td>• 1,3,5,7</td><td>1
(AO3.3)
(1)</td><td><u>Examiner’s Comments</u>
Candidates who answered the previous part of the question generally went on to score full credit for a second trace of the algorithm with a different calling argument value.</td>
</tr>
<tr>
<td></td><td>c</td><td></td><td>1 mark per bullet up to a maximum of 2 marks, e.g.:

• Modulo division
• Finds the remainder after a division</td><td>2
(AO3.3)
(2)</td><td><u>Examiner’s Comments</u>
Candidates who successfully identified that a remainder was calculated found it harder to</td>
</tr>
</table>

			• Used to determine if `a` is an odd or even number		determine that its purpose was to see if the number was even or odd. **Misconception** Some candidates think that a modulo division calculation resulting in 0 means that there is no remainder. Where the result of modulo division is 0 it means that the remainder is 0.
			Total	**5**	
9			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)	
			Total	**3**	
10	a		**Mark Band 3 – High level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of bubble sorts; 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 bubble sorts within the context which results in a supported and realistic judgment as to whether it is suitable to use within the 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 bubble sorts; 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	9 AO1.1 (2) AO1.2 (2) A02.1 (2) A03.3 (3)	**Knowledge and Understanding** • All adjacent items are compared against each other. • The biggest number in the adjacent pair is swapped over with the smallest number. A temp variable is used to hold the data while it's being moved. • When a swap is made a flag is set. • This is repeated for all adjacent values, known as one pass. • At the end of one pass, the largest item should appear at the end of the list. • If at the end of the list the flag has been set the flag is unset and the algorithm starts from the beginning of the list again.

			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 bubble sorts 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 bubble sorts 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.		• When the algorithm gets to the end of the list and the flag is unset the list is sorted. **Application** • As there are 250,000 items a bubble sort would perform very slowly as a lot of passes will need to be made in order to sort the items. • Bubble sorts are better suited to data sets where the items are almost/partly sorted. However the smaller numbers are currently towards the end and the larger numbers are towards the start. • This will therefore increase the amount of comparisons / passes/swaps required which will therefore slow the performance of the sort down. **Evaluation** • The algorithm is easy to implement as the number of lines of code is less than other standard sorting algorithms. • Although a bubble sort will be able to sort the items into order, it will take longer than other sorting algorithms due to the number of items and the current order or items in the unsorted list.
	b		• 249,999	1 A02.2 (1)	
			Total	**10**	
11	a		1 mark per pointer • queueHead: Point to the first element in the queue / next element to remove • queueTail: Point to the last element in the queue	2 AO1.2 (2)	

	b		1 mark per bullet up to max 5 • first 3 jobs removed • 128 and 129 added in positions 4 and 5 respectively • no additional jobs • queueHead being 3 (FT errors) • queueTail being 5 (FT errors)  	5 AO2.1 (2) AO2.2 (3)	The underlying implementation of the queue has not been specified, so allow alternative valid answers. e.g. queueHead = 0 queueTail = 2 Location 2: 129 Location 1: 128 Location 0: 127

	c	i	1 mark per bullet to max 5
			
			• Function declaration
			• Checking if queue is empty
			• …returning `null`
			• (Otherwise) incrementing `queueHead`
			• …returning `buffer[queueHead-1]`
			
			e.g.
			`function dequeue()`
			`  if queueHead > queueTail then`
			`  return null`
			`  else`
			`  queueHead = queueHead + 1`
			`  return buffer[queueHead-1]`
			`  endif`
			`endfunction`
				5
				AO2.2
				(2)
				AO3.3
				(3)
					Note: Accept alternative valid underlying implementation answers e.g. Shifting all elements in queue forward.

		ii	1 mark per bullet to max 6
			
			• Function declaration taking parameter
			• Checking if queue is full
			• …returning -1
			• (Otherwise) incrementing queueTail
			• Adding newJob to buffer(queueTail)
			• Returning 1
			
			e.g.
			`function enqueue(newJob)`
			`  if queueTail == 99 then`
			`  return -1`
			`  else`
			`  queueTail = queueTail + 1`
			`  buffer[queueTail] = newJob`
			`  return 1`
				6
				AO2.2
				(3)
				AO3.3
				(3)	

<table>
<tr><td></td><td></td><td></td><td>

```
  endif
endfunction
```

</td><td></td><td></td></tr>
<tr><td></td><td></td><td>iii</td><td>

1 mark per bullet to max 8

- Inputting user choice
- If enqueue chosen input job name
- …call enqueue with input value as parameter
- …check if return value is -1 and output full
- …otherwise output message that item is added
- If dequeue chosen
- …call dequeue **and** save returned value
- …output returned value (jobname) if not null
- …or output queue is empty

e.g.

```
main()
   choice = input("Add or remove?")
   if choice == "ADD" then
      jobname = input("Enter job name")
      returnValue = enqueue(jobname)
      if returnValue == -1 then
         print("Queue full")
      else
         print("Job added")
      endif
   else
      returnValue = dequeue()
      if returnValue == null then
         print("Queue empty")
      else
         output returnValue
      endif
   endif
endmain
```

</td><td>8
AO2.2
(2)
AO3.3
(6)</td><td>Allow equivalent checks / logic</td></tr>
<tr><td></td><td>d</td><td></td><td>

1 mark per bullet to 3

- Check if either head or tail are incremented to above 99
- … set to be 0 instead
- When checking if array is full check if (queueTail == queueHead – 1) OR (queueTail==99 AND queueHead==0)

</td><td>3
AO2.1
(1)
AO2.2
(2)</td><td>Credit equivalent modulo arithmetic solution</td></tr>
<tr><td></td><td>e</td><td></td><td>

1 mark per bullet to max 3, e.g.

- Use a different structure e.g. a linked list

</td><td>3
AO2.1
(2)</td><td></td></tr>
</table>

			• …items can be added at different points in the linked list depending on priority • …by changing the pointers to items needing priority • Have different queues for different priorities • …add the job to the queue relevant to its priority • …print all the jobs in the highest priority queue first	AO2.1 (1)	Allow other suitable descriptions that show how the program could be amended.
			Total	**32**	
12	a		• 10	1 A03.2 (1)	
	b		• 30	1 A03.2 (1)	
	c		• 10	1 A03.2 (1)	
			Total	**3**	