Qualification Accredited

OCR
Oxford Cambridge and RSA

A LEVEL

Examiners' report

COMPUTER SCIENCE

H446
For first teaching in 2015

H446/02 Summer 2019 series

Version 1

Contents

Introduction

Our examiners' reports are produced to offer constructive feedback on candidates' performance in the examinations. They provide useful guidance for future candidates. The reports will include a general commentary on candidates' performance, identify technical aspects examined in the questions and highlight good performance and where performance could be improved. The reports will also explain aspects which caused difficulty and why the difficulties arose, whether through a lack of knowledge, poor examination technique, or any other identifiable and explainable reason.

Where overall performance on a question/question part was considered good, with no particular areas to highlight, these questions have not been included in the report. A full copy of the question paper can be downloaded from OCR.

Paper 2 series overview

The paper differentiated the candidates effectively and the scripts included some very strong candidate responses.

Questions that targeted Knowledge and Understanding required candidates to have studied the whole specification and to have learnt relevant definitions. Some candidates had not been prepared by covering the whole specification and thus did not achieve marking points targeted at lower grades for basic recall.

Questions targeting Application required higher order skills to be able to use knowledge gained in context to solve problems. There was clear differentiation between candidates who understood the concepts and who could apply them, and those who displayed little ability to apply what they had learnt.

It was clear that a significant number of candidates had limited exposure to Object Oriented Programming (OOP) and had little experience writing programs that required classes to be defined. Section B of the paper will always have a scenario that uses OOP and candidates do need experience of writing code in this paradigm to do well. Candidates would benefit from being prepared for this by practically implementing solutions to past paper questions or similar scenarios.

Certain types of question benefit greatly from being answered by using clear diagrams. The questions on Dijkstra's algorithm, bubble sort, and merge sort, were all best answered through the use of clear diagrams and tables with appropriate annotation. Where possible, when a computational algorithm is being executed, the use of verbose text should be discouraged.

Question 1 (a)

1 The temperatures of an ocean are input into a computer system. They are recorded, and will be accessed, in the order in which they arrive. The data for one week is shown:

5, 5.5, 5, 6, 7, 6.5, 6

(a) The data is to be stored in a data structure. The programmer stores the data in a queue.

Explain why a queue is used instead of a stack.

.....

.....

.....

..... **[2]**

The majority of candidates were able to define a queue as a FIFO structure. Most of those who did so were able to successfully go on to relate this to the context of the question, which was to be able to retrieve the temperature values in the same order as they had originally been recorded. A number of candidates answered equally well by explaining why a stack would not have been appropriate.

Question 1 (b) (i)

(b) The data is processed. After processing, the value for the first day is stored as 0. The value for each following day is stored as an increase, or decrease, from the first day.

For example: if the first day was 7, the second was 6 and the third was 9, after processing it would be stored as 0, –1, 2.

(i) The queue uses `dequeue()` to return the first element of the queue.

`dequeue()` is a function.

Explain why `dequeue()` is a function, not a procedure.

.....

..... **[1]**

Most candidates had learnt that a function always returns a value and scored full credit for recall of knowledge. Some candidates gave an ambiguous response stating that functions produce an output (procedures can produce outputs too, e.g. printing to the screen), which lacked the precision of definition required to gain credit.

Question 1 (b) (ii)

(ii) Complete the algorithm to process the data in the queue and store the results in an array called `processedData`.

```
processedData[0] = 0

firstDay = .....

for count = 1 to 6

    processedData[.....] = dequeue() - .....

next count
```

[3]

Many candidates found it difficult to complete the code successfully – most often finding the application of the *dequeue()* function to return the first item in the queue the most difficult mark. Candidates also need to remember that variable identifiers need to be constructed correctly. Identifier names cannot have spaces in them, so those who answered *first Day* instead of *firstDay* did not provide a correct solution.

Question 1 (b) (iii)

(iii) The contents of `processedData` are shown.

0	0.5	0	1	2	1.5	1

The data needs to be sorted into ascending order.

Explain how a bubble sort algorithm sorts data. Use the current contents of `processedData` in your explanation.

.....

.....

.....

.....

.....

..... **[5]**

Exemplar 1

A bubble sort compares each pair of adjacent values and swaps them if they are incorrectly sorted. Once each pair has been compared 1 pass is complete, the process then repeats, stopping once no swaps are completed in a pass.

Data will be sorted like this:

0, 0.5, 0, 1, 2, 1.5, 1

0, 0, 0.5, 1, 1.5, 1, 2 (0.5 > 0, 2 > 1.5, 2 > 1)

0, 0, 0.5, 1, 1, 1.5, 2 (1.5 > 1)

0, 0, 0.5, 1, 1, 1.5, 2 (No swaps, so list is sorted)

[5]

Some candidates gave a purely theoretical description of bubble sort and were limited to a maximum of three marks for doing so. Where questions provide a specific data set to be used, candidates must make reference to it in their answers.

Candidates who gave diagrams in their answers (such as the exemplar) often demonstrated all of the steps of the algorithm in the clearest fashion. Candidates should be encouraged to use annotated diagrams rather than verbose prose where possible.

Question 1 (b) (iv)

(iv) A bubble sort has the following complexities:

Best time	O(n)
Average and worst time	$O(n^2)$
Worst space	O(1)

Describe what each of these complexities mean.

Best time O(n)

.....

.....

.....

.....

Average and worst time $O(n^2)$

.....

.....

.....

.....

Worst Space O(1)

.....

.....

.....

.....

[6]

Many candidates had not learnt the appropriate terminology for linear, polynomial and constant complexity. Stronger candidates recognised that bubble sort had a best case when the items were already sorted, and a worse case when the items were in reverse order.

Candidates struggled most with the concept of space complexity. Many scored marks for recognising that O(1) was constant, but then confused space complexity with time complexity. Few appreciated that O(1) meant that no additional storage was required *in addition to* the initial data set itself.

?	**Misconception**	A number of candidates equated O(n) linear complexity to mean the number of steps was equal to n. The number of steps is proportional to n since there are many other factors at play. It should be noted that ½ n and 2n would still give time complexity of O(n).

Question 2 (a) (i)

2 A program needs to store the names of plants that are in a garden, so they can be easily found and accessed in alphabetical order.

The data is stored in a tree structure. Part of the tree is shown.

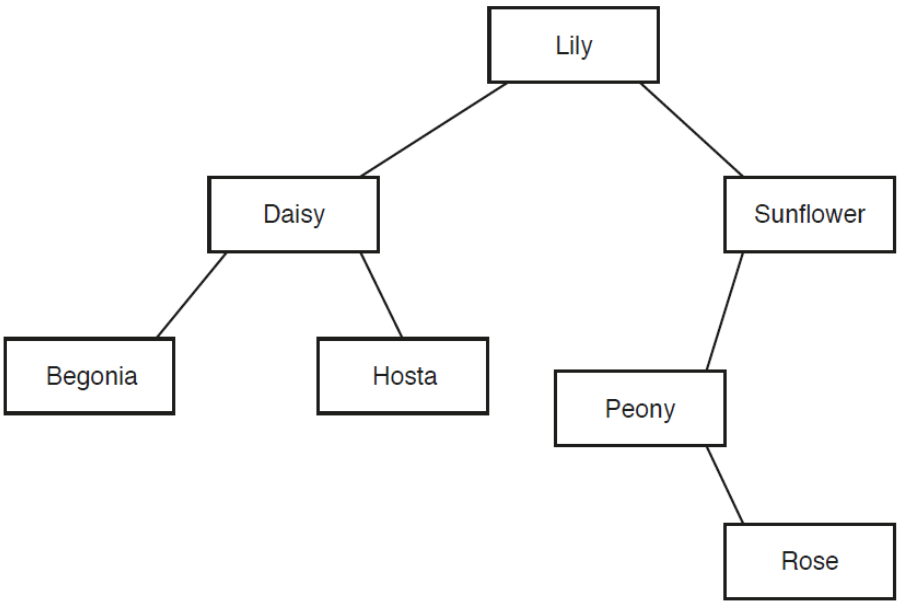


Fig. 2.1

(a) (i) State the type of tree shown in Fig. 2.1.

..... **[1]**

Most candidates had the knowledge to recall the correct term.

Question 2 (a) (ii)

(ii) Show the output of a breadth-first traversal of the tree shown in Fig. 2.1.

.....

.....

.....

.....

.....

..... **[3]**

Most candidates could successfully show the result of executing a breadth-first search on the tree. Where candidates did not achieve the marks, they often confused depth-first with breadth-first.

Question 2 (a) (iii)

(iii) Explain how backtracking is used in a depth-first (post-order) traversal. Use the tree in Fig. 2.1 in your explanation.

.....

.....

.....

.....

.....

..... **[4]**

Candidates who gave generic answers limited themselves to a maximum of two marks. Where questions ask for a specific dataset to be used, candidates will need to do so if they are to access all of the marks available. While the general principles of a depth-first traversal were often understood in terms of traversing down the leftmost branches from Lily to Daisy to Begonia, the order in which the nodes would be output was less often appreciated.

Question 2 (b) (i)

(b) The elements in the tree in Fig. 2.1 are read into a linked list producing an alphabetised list.

(i) Complete the following table to show the linked list for the data.

Data item	Data	NextPointer
0	Begonia	
1	Daisy	
2	Hosta	
3	Lily	
4	Peony	
5	Rose	
6	Sunflower	

[2]

Most candidates displayed some understanding of the use of pointers in a linked list and successfully gave the correct values. Some candidates erroneously gave 0 as a null pointer value or left the NextPointer value for Sunflower empty. Null, -1 and Ø were all accepted null pointer values.

Question 2 (b) (ii)

(ii) A new plant, Lavender, needs adding to the linked list. The linked list needs to retain its alphabetical order.

Complete the table to show the linked list after Lavender is added.

Data item	Data	NextPointer
0	Begonia	
1	Daisy	
2	Hosta	
3	Lily	
4	Peony	
5	Rose	
6	Sunflower	

[3]

The majority of candidates appreciated that a new entry could be made to the list by using the last available space, and then updating the relevant pointers. It was pleasing to see few candidates tried to shift the data items down, rather than updating the pointers.

Question 2 (b) (iii)

(iii) Hosta needs removing from the linked list.

Explain how a data item is removed from a linked list. Use the removal of Hosta in your answer.

.....

.....

.....

.....

.....

..... **[4]**

Candidates sometimes struggled to break their explanations into a clear sequence of steps, using the specific data items given. Those candidates who scored well made clear reference to the relevant data items Daisy, Hosta and Lavender. A few candidates gave annotated diagrams that explained the process particularly clearly.

Question 2 (b) (iv)

(iv) The linked list is stored as a 2D array with the identifier `plantList`. The index of the first element of the linked list is stored in the identifier `firstElement`.

All contents of the linked list need to be output in alphabetical order.

Write an algorithm to follow the pointers to output the contents of the linked list in alphabetical order.

Add comments to explain your code.

.....

.....

.....

.....

.....

..... **[5]**

Candidates need to be familiar with a range of data structures and the associated algorithms for performing basic functions on the data structures. Many candidates erroneously thought that the individual elements of a linked list could be indexed and accessed with a *for* loop. Relatively few candidates appreciated that the items in the linked list could only be traversed by following each item's next pointer until the end of the list was located.

Many candidates continue to show limited ability to write pseudocode.

Question 3 (a) (i)

3 A recursive function, `GCD`, is given in pseudocode.

```
function GCD(num1, num2)
        if num2 == 0 then
                return num1
        else
                return GCD(num2, num1 MOD num2)
        endif
endfunction
```

(a) The function uses branching.

(i) Identify the type of branching statement used in the function.

..... **[1]**

A number of candidates overthought the answer and did not identify the *if* branching statement used in the code.

Question 3 (a) (ii)

(ii) Explain the difference between branching and iteration.

.....

.....

.....

..... **[2]**

Many candidates found difficulty in defining either branching or iteration clearly. Often the two terms were confused, with the clear differences between branching and iteration not expanded on.

Question 3 (a) (iii)

(iii) Identify the **two** parameters in the function.

1 .....

2 .....

[1]

Most candidates had little difficulty correctly identifying the parameters num1 and num2.

Question 3 (a) (iv)

(iv) State whether the parameters should be passed by value, or by reference. Justify your answer.

.....

.....

.....

..... **[2]**

Most candidates identified that the parameters should be passed by value, but fewer could successfully justify their choice, and hence show a deeper understanding of the reason for the choice they made.

Question 3 (a) (v)

(v) Describe the arithmetic operation of `MOD`. Use an example in your answer.

.....

.....

.....

.....

.....

..... **[2]**

Most candidates could successfully define the MOD function. A few confused it with the absolute() function or with integer division, and some did not illustrate their answer with the required example.

Question 3 (b)

(b) Trace the recursive function when it is called by the statement `GCD(250, 20)`. Give the final value returned.

.....

.....

.....

.....

.....

Final return value: .....

[3]

Many candidates successfully traced the algorithm given.

Question 3 (c) (i)

(c) The function has been rewritten using iteration instead of recursion.

(i) State **one** benefit and **one** drawback of using iteration instead of recursion.

Benefit.....

.....

Drawback.....

.....

[2]

Most candidates achieved some credit, but many answers were often too vague.

Question 3 (c) (ii)

(ii) Complete the missing statements in this iterative version of the function.

```
function newGCD(num1, num2)
  temp = 0
  while (num2 != ……)
    …… = num2
    num2 = num1 MOD ……
    num1 = temp
  endwhile
  return ……
endfunction
```

[4]

Most candidates attempted the question and achieved some credit. Relatively few achieved full marks for this question.

Question 4 (a)

4 Mabel is a software engineer. She is writing a computer game for a client. In the game the main character has to avoid their enemies. This becomes more difficult as the levels of the game increase.

(a) Mabel uses decomposition to design the program.

Explain how decomposition can aid the design of this program.

……

……

……

…… **[2]**

Decomposition was clearly understood and most candidates achieved at least one of the two marks. Those that did not gain full marks did not go beyond decomposition breaking the task into sub-tasks.

Question 4 (b) (i)

(b) The computer game allows a user to select a character (e.g. name, gender). They can then choose a level for the game (easy, normal, challenging). The user controls their character by moving it left or right. The character can jump using space bar as an input. If the character touches one of the enemies then it loses a life. The character has to make it to the end of the level without losing all their lives.

The game is designed in a modular way.

(i) One sub-procedure will handle the user input.

Describe **three** other sub-procedures Mabel could create for the given game description.

1 .....

.....

.....

.....

2 .....

.....

.....

.....

3 .....

.....

.....

.....

[6]

Most candidates presented reasonable procedures within the context of the question. Relatively few made the mistake of reiterating procedures to handle character movement that was specified as an example in the question.

Question 4 (b) (ii)

(ii) Describe the decision that the program will need to make within the user input sub-procedure and the result of this decision.

.....

.....

.....

..... **[2]**

A number of answers lacked the clarity of a clear decision and subsequent action that would be performed. A number of answers given did not relate to the user input sub-procedure as specified.

Question 4 (b) (iii)

(iii) Define pipelining and give an example of how it could be applied in the program.

.....

.....

.....

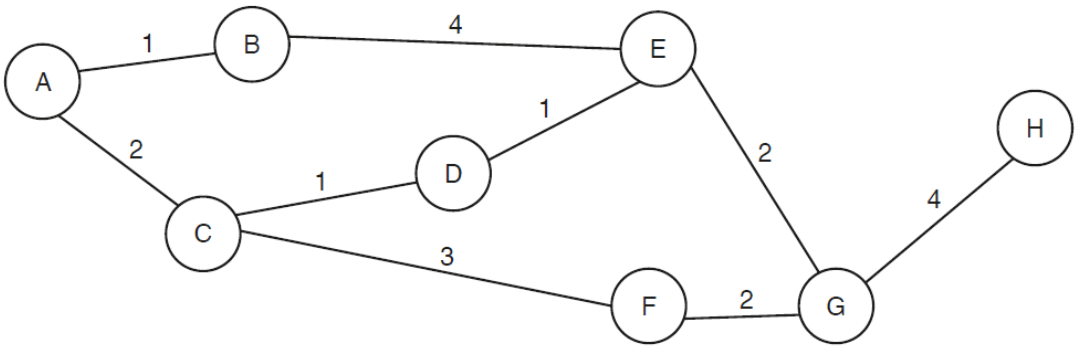
..... **[2]**

Many candidates could define the concept of pipelining. Fewer were able to apply it within the context of the given scenario.

Question 4 (c)

(c) The game's 'challenging' level has intelligent enemies that hunt down the character in an attempt to stop the user from winning. The program plans the enemies' moves in advance to identify the most efficient way to stop the user from winning the game.

The possible moves are shown in a graph. Each node represents a different state in the game. The lines represent the number of moves it will take to get to that state.



Show how Dijkstra's algorithm would find the shortest path from A to H.

.....

.....

.....

.....

.....

..... **[6]**

Exemplar 2

Show how Dijkstra's algorithm would find the shortest path from A to H.

Node	Visited	Shortest distance from start	Previous node
A	✓	0	–
B	✓	1	A
C	✓	2	A
D	✓	3	C
E	✓	~~5~~ 4	~~B~~ D
F	✓	5	C
G	✓	6	E
H	✓	10	G

Shortest path is A C D E G H
and takes a total of 10 moves.

[6]

Dijkstra's shortest path algorithm has appeared in previous sessions and candidates generally had a good grasp of the principles of the algorithm. Verbose text often made some responses difficult to follow. The clearest responses (as exemplified) tabulated the steps in the algorithm.

Question 4 (d)

(d)* Mabel has been told that true programmers write programs in a text editor, and do not use IDEs. Mabel does not agree with this statement.

Discuss the use of an IDE in the development of this program.

.....

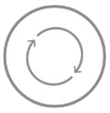
.....

.....

.....

.....

..... **[9]**

	**AfL**	Many candidates would benefit from being able to present a clear and logical structure to their answers. Candidates must show knowledge, application and evaluation. First, relevant knowledge (definitions) should be introduced. Secondly, examples that are relevant to the given scenario should be covered. Finally, evaluative comments should be made to compare the different methods in terms of relative advantages / disadvantages or impacts / consequences.

Candidates had first-hand experience of using IDEs and features found within IDEs. Most candidates were able to give a Level 2 response giving a range of features found within an IDE along with descriptions of how they could be used. The evaluation commentary was often limited to generic comments such as increased productivity.

Few candidates could offer more insightful depth to their evaluatory comments linking features to categories of users such as those features particularly beneficial to inexperienced programmers (such as predictive text, error correction suggestions) or experienced teams of programmers (collaborative online features, version control).

Question 5 (a)

5 A 1-dimensional array stores the following data:

Index	0	1	2	3	4	5
Data	2	18	6	4	12	3

(a) The array needs sorting into descending order.

Describe how a merge sort would sort the given array into descending order.

.....

.....

.....

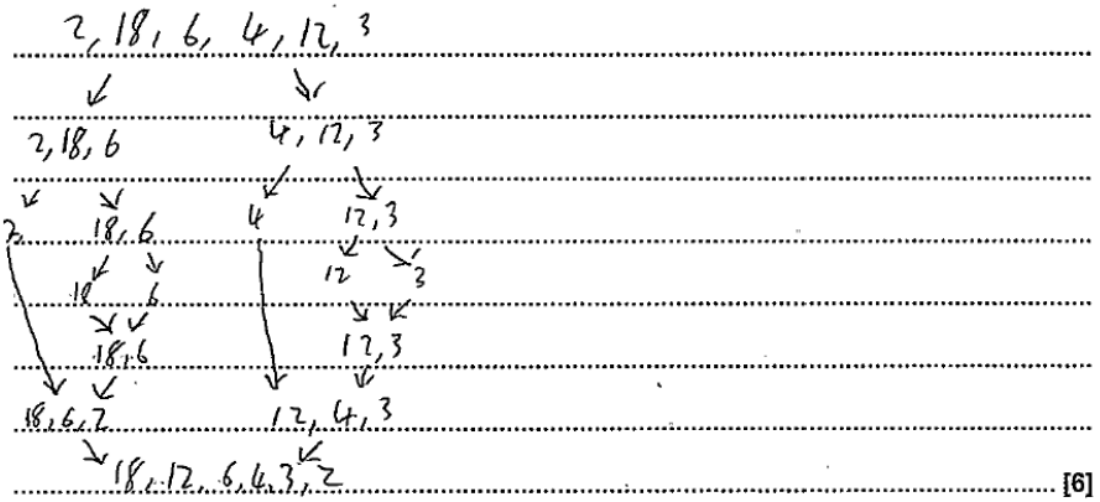
.....

.....

..... **[6]**

Exemplar 3

A merge sort is completed by splitting a list in half, then merge sorting each half, then combining each sorted half into 1 sorted list by looking at the first item of each list and putting the ~~smaller~~ larger one into the new list until all are placed. The base case is 2 lists of length 1 which are just placed into descending order.



[6]

A number of candidates simply presented learnt definitions without application to the given dataset, while other candidates presented diagrammatic solutions to the workings of the merge sort for the given data, but did not explain what was happening.

Good responses such as the exemplar had explanatory text alongside a clear diagram, showing the steps that took place when the algorithm was run.

Question 5 (b)

(b) An insertion sort can be used to sort the array instead of a merge sort.

Explain why an insertion sort might use less memory than a merge sort.

.....

.....

.....

..... **[2]**

Many candidates recognised that merge sort could generate an additional array each time a list was split. Fewer could explain that insertion sort worked in-situ and has a space complexity O(1).

Question 6

6* Benedict runs a social networking website. He has been told he should use data mining to help him enhance and improve his website.

Evaluate the use of data mining to help Benedict enhance and improve his social networking website.

.....

.....

.....

.....

.....

..... **[9]**

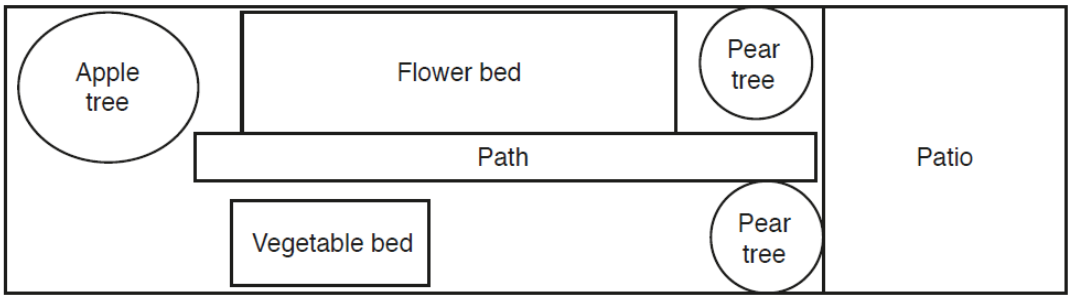
Many candidates were familiar with the context of social networking and could provide examples of how data mining could present useful data to assist in facilitating customer connections or targeted advertising. However, on occasion, a number of candidates did forget to actually define what they actually meant by data mining or illustrate the type of data that could be collected for processing.

A pleasing number of candidates did go on to discuss and evaluate the computational resources that would be required and/or the impact on users' personal liberties and freedoms that needed to be weighed against the potential benefits to the company.

Question 7 (a) (i)

7 A program is needed to plan the layout of a garden.

The program will allow the user to create an image of the garden, for example:



(a) The programmer will use abstraction to produce the program interface to represent the garden.

(i) Give **two** different examples of how abstraction has been used to produce the layout of the garden.

1 .....

.....

2 .....

..... **[2]**

Candidates understood the concept of abstraction and had little difficulty giving examples of how abstraction had been applied to the layout of the garden given.

Question 7 (a) (ii)

(ii) Explain the need for abstraction in the production of this program.

.....

.....

.....

.....

.....

..... **[3]**

Many candidates confused defining abstraction with the actual need to use abstraction. Candidates also often related their answers to the potential users of the system, rather than to the production of the program, as specified by the question.

Question 7 (a) (iii)

(iii) The user needs to input data into the program to set up their garden layout.

Identify **three** pieces of data that the user may input into this program.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

Most candidates had little difficulty identifying suitable inputs to the system. There were occasional instances of repetition, and candidates need to guard against this.

Question 7 (b) (i)

(b) The program is to be built using object oriented programming.

All items that can be added to the garden are declared as instances of the class `GardenItem`.

The class has the following attributes:

Attribute	Description	Example
`itemName`	The name of the item	Flowerbed
`length`	The length of the item in metres	2
`width`	The width of the item in metres	1

(i) The constructor method sets the attributes to values that are passed as parameters.

Write pseudocode or program code to declare the class `GardenItem` and its constructor. All attributes should be private and initialised through the constructor (e.g. `daisies = new GardenItem("Flowerbed",2,1)`). **[4]**

Many candidates struggled to differentiate between defining a class and defining a constructor method – with many passing parameters to the class definition rather than to the constructor method itself. While candidates are not required to use the pseudocode standard that appears in the specification, they are required to use pseudocode that is understandable. Many candidates need further practical experience of using OOP.

Question 7 (b) (ii)

(ii) The trees in the garden layouts are defined by the class `Tree`. This class inherits from `GardenItem`.

The class `Tree` has the additional attributes: `height`, `sun`, `shade`.

If `sun` is `true` then the tree can grow in full sun, if it is `false` then it cannot.

If `shade` is `true` then the tree can grow in full shade, if it is `false` then it cannot.

The length and width of a tree are the same. Only one value for these measurements is passed to the constructor.

Write an algorithm, using pseudocode or program code, to declare the class `Tree`. Declare all attributes as private.

.....

.....

.....

.....

.....

..... **[5]**

Few candidates had the knowledge and skills required to answer this question well. Common misconceptions included an assumption that only the additional properties of *height, sun* and *shade* needed to be passed to the *Tree* class. Very few candidates set the parameters *itemName, Length* and *Width* of the super class successfully.

Question 7 (b) (iii)

(iii) The Common Oak is a type of tree. It has a maximum height, length and width of 40 m. It can grow in sun and shade.

Write a statement, using pseudocode or program code, to declare an instance of tree for the Common Oak. Give the object the identifier `firstTree`.

.....

.....

..... **[4]**

Very few candidates achieved full credit, but many candidates did achieve some credit for assigning an instance of the *Tree* class to the *firstTree* identifier. Few went on to correctly pass the required parameters in the correct order.

Question 7 (b) (iv)

(iv) The classes `GardenItem` and `Tree` use get and set methods to access and alter their private attributes.

Write the get method `getItemName` and set method `setItemName` for class `GardenItem`. The set method takes the new value as a parameter.

Do not write any other methods, or re-declare the class.

.....

.....

.....

.....

.....

..... **[4]**

Those candidates who gained credit in the initial parts of the question generally went on to successfully define the *get* and *set* methods. A number of candidates erroneously passed parameters into the *get* function or did not return the *itemName* parameter. Another common misconception was to ask for a user input to set the value of *itemName* in the *set* procedure, rather than passing a parameter for the assignment.

Where candidates had experience of OOP is was apparent that they knew how to use getters and setters to access the private attributes of a class. Those who had no experience of this performed poorly.

Question 7 (b) (v)

(v) The trees in the garden layouts are stored in a 1-dimensional array, `treeArray`. The array can store a maximum of 1000 items. The array has global scope.

A procedure, `findTree`, takes as parameters:

- The maximum height of a tree
- The maximum width of a tree
- Whether the tree can live in full sun
- Whether the tree can live in full shade.

It searches the array, `treeArray`, for all trees that do not exceed the maximum height and width, and that can grow in the conditions available. If there are no suitable trees, a suitable message is output.

It outputs the name and details of the trees found in an appropriate message.

Call the get methods, `getItemName`, `getHeight`, `getWidth`, `getSun`, `getShade`, to access the attributes.

Write, using pseudocode or program code, the procedure `findTree`.

.....

.....

.....

.....

.....

..... **[6]**

Many candidates successfully defined the *findTree* procedure declaration with the relevant parameters and then went on to correctly form a loop that would iterate the correct number of times. However, many candidates with poor knowledge of OOP then found it difficult to use the *Tree* attributes correctly through the use of the *get* methods. A sizeable number of candidates thought that the properties of the class could be addressed directly and thus showed a lack of appreciation of the encapsulation imposed by the OOP paradigm. Where candidates had a solid grounding in OOP they had little difficulty in obtaining full marks.

Question 7 (c)

(c)* The programmer is designing the program to make use of caching and re-useable components.

Explain and evaluate the use of caching and re-useable components in the design of the garden program.

.....

.....

.....

.....

.....

..... **[9]**

Most candidates were able to identify generic points about re-usable code / modules. Fewer could apply their knowledge of re-usable components in the context of the given scenario, and more context was required to achieve higher scores. It was pleasing to see a number of strong responses that referenced the use of classes and inheritance and how it could be applied within the scenario to achieve benefits from generalisation. Caching responses were limited to cache memory on the processor by less able candidates. Better responses explained how commonly accessed components like particular graphic items could be stored in RAM to make retrieval faster. The best responses evaluated at a level where the effectiveness of caching was measured in terms of hit ratios versus the extra development in terms of time and complexity were weighed. As with the other level of response questions candidates achieved higher scores for demonstrating critical thinking with relevant examples evaluated in depth.

Supporting you

For further details of this qualification please visit the subject webpage.

Review of results

If any of your students' results are not as expected, you may wish to consider one of our review of results services. For full information about the options available visit the OCR website. If university places are at stake you may wish to consider priority service 2 reviews of marking which have an earlier deadline to ensure your reviews are processed in time for university applications.

activeresults

Review students' exam performance with our free online results analysis tool. Available for GCSE, A Level and Cambridge Nationals.

It allows you to:

- review and run analysis reports on exam performance
- analyse results at question and/or topic level*
- compare your centre with OCR national averages
- identify trends across the centre
- facilitate effective planning and delivery of courses
- identify areas of the curriculum where students excel or struggle
- help pinpoint strengths and weaknesses of students and teaching departments.

*To find out which reports are available for a specific subject, please visit ocr.org.uk/administration/support-and-tools/active-results/

Find out more at ocr.org.uk/activeresults

CPD Training

Attend one of our popular CPD courses to hear exam feedback directly from a senior assessor or drop in to an online Q&A session.

Please find details for all our courses on the relevant subject page on our website.

www.ocr.org.uk

The small print

OCR Resources: *the small print*

OCR's resources are provided to support the delivery of OCR qualifications, but in no way constitute an endorsed teaching method that is required by OCR. Whilst every effort is made to ensure the accuracy of the content, OCR cannot be held responsible for any errors or omissions within these resources. We update our resources on a regular basis, so please check the OCR website to ensure you have the most up to date version.

This resource may be freely copied and distributed, as long as the OCR logo and this small print remain intact and OCR is acknowledged as the originator of this work.

Our documents are updated over time. Whilst every effort is made to check all documents, there may be contradictions between published support and the specification, therefore please use the information on the latest specification at all times. Where changes are made to specifications these will be indicated within the document, there will be a new version number indicated, and a summary of the changes. If you do notice a discrepancy between the specification and a resource please contact us at: resources.feedback@ocr.org.uk.

Whether you already offer OCR qualifications, are new to OCR, or are considering switching from your current provider/awarding organisation, you can request more information by completing the Expression of Interest form which can be found here: www.ocr.org.uk/expression-of-interest

Please get in touch if you want to discuss the accessibility of resources we offer to support delivery of our qualifications: resources.feedback@ocr.org.uk

Looking for a resource?

There is now a quick and easy search tool to help find **free** resources for your qualification:

www.ocr.org.uk/i-want-to/find-resources/

www.ocr.org.uk

OCR Customer Support Centre

General qualifications

Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk

OCR is part of Cambridge Assessment, a department of the University of Cambridge. *For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored.*

Cambridge Assessment

CERTIFIED ISO 9001 UKAS MANAGEMENT SYSTEMS 001