Qualification Accredited

OCR
Oxford Cambridge and RSA

A LEVEL

Examiners' report

COMPUTER SCIENCE

H446

For first teaching in 2015

H446/02 Summer 2022 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. A selection of candidate answers are also provided. 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 and the mark scheme can be downloaded from OCR.

Advance Information for Summer 2022 assessments

To support student revision, advance information was published about the focus of exams for Summer 2022 assessments. Advance information was available for most GCSE, AS and A Level subjects, Core Maths, FSMQ, and Cambridge Nationals Information Technologies. You can find more information on our website.

Paper 2 series overview

Paper 2 focuses on algorithms and problem solving. It tests candidates' computational thinking ability to analyse and solve problems. Candidates are expected to be able to write algorithms fluently in either pseudocode or program code and to be able to trace algorithms.

Candidates who did well on this paper generally did the following:	Candidates who did less well on this paper generally did the following:
• Demonstrated good factual knowledge of definitions of terms on the specification. • Demonstrated an ability to trace algorithms carefully to determine their output and purpose. • Demonstrated an ability to analyse questions and respond with appropriately detailed answers for the number of marks available. • Demonstrated an ability to trace and discuss standard algorithms such as the linear search, stack and queue operations, merge sort, breadth-first traversal, depth-first traversal and Dijkstra's algorithm. • Demonstrated an ability to write pseudocode to solve a problem.	• Showed some factual knowledge but were less able to show application. • Produced responses that were often too vague that identified a point but lacked an expansion in terms of relevant qualification. • Struggled to trace the logic of algorithms presented to them. • Showed little detailed knowledge of, or ability to describe, standard algorithms. • Showed an inability to write pseudocode or program code.

Section A overview

Section A includes questions where the context for each separate question is set in the stem of the question. Candidates need to be mindful of the fact that context is always important when considering their responses.

Question 1 (a) (i)

1 A computer program stores data in an array named `words`.

(a) The data in the array needs to be searched for a value that the user inputs.

(i) One example of a searching algorithm is a binary search.

Identify the precondition for a binary search.

.....

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

Most candidates successfully identified that data must be ordered as a precondition for a binary search. A few candidates were too vague giving unqualified answers such as 'must be sorted', without specifying what had to be sorted.

Question 1 (a) (ii)

(ii) A second example of a searching algorithm is a linear search.

Describe how a linear search works.

.....

.....

.....

.....

.....

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

Many candidates scored some marks for describing the steps involved in a linear search, but relatively few presented a comprehensive and detailed description for full marks. Many responses used vague language such 'checking if the value is found' without explaining that a comparison between the current term and the target search value must be performed. Other examples of vague language use included responses such as 'keep going until you find what you're looking for', which begged the question, how do you know when you've found what you're looking for?

Common errors included candidates who mistakenly described a binary search, and those who did not answer the question. Examples of not answering the question included giving properties of a linear search such as its linear run time or the fact that it can be run on an unordered data set.

Question 1 (b) (i)

(b) The array `words` is defined as a global variable and contains these values:

"house"	"boat"	"car"	"telephone"	"garden"	"spice"	"elephant"

The pseudocode function `useWords()` here uses the global array `words`.

The number of words in the array `words` is passed as a parameter.

```
function useWords(numberOfWords : byVal)
  contents = ""
  for count = 0 to numberOfWords - 1
    contents = contents + words[count] + " "
  next count
  return contents
endfunction
```

(i) Identify **two** variables in the function `useWords()`.

1 .....

2 .....

[2]

The majority of candidates answered correctly, with the most popular answers being `count` and `contents`. A few candidates incorrectly gave data values from the array rather than identifying variables in the function.

Question 1 (b) (ii)

(ii) `numberOfWords` is a parameter passed by value.

Describe the difference between passing a parameter by value and by reference.

.....

.....

.....

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

It was pleasing to see an improvement in responses to this topic this session. There were still some answers that were too vague that did not specify that *by value* uses a copy of the parameter and that *by reference* passes the memory address. Other examples of vagueness that were not given marks included answers such as 'by value can't change the value while by reference can' that did not qualify the scope within which changes can be made. A variable passed by value can clearly be changed in a function, but it is the local copy that is changed and then disregarded when the function finishes.

Question 1 (b) (iii)

(iii) Rewrite the function `useWords()` to use a while loop instead of a for loop.

The function header and close have been written for you.

Write your answer using pseudocode or program code.

```
function useWords(numberOfWords : byVal)
```

.....

.....

.....

.....

.....

.....

```
endfunction
```

[4]

Many candidates struggled with this question. Some candidates rewrote the `for` loop putting the word *while* in place of *for* showing little understanding of the difference between a counter-controlled and a conditional loop.

Common errors included not initialising the `count` variable before using it within the body of the while loop, off-by-one errors, and forgetting to increment the `count` variable within the loop. Poor indentation was often a problem, and a number of candidates erroneously placed the return statement inside the body of the loop.

Off-by-One errors

There were many off-by-one errors observed, e.g. `while count <= numberOfWords` rather than `while count < numberOfWords`.

Candidates need to give code that is logically accurate, and, in this instance, it required the loop to run the correct number of times so that all the words in the array were processed.

Question 1 (c)

(c) Give **one** benefit and **one** drawback of declaring the array as a global variable instead of a local variable.

Benefit .....

.....

Drawback .....

.....

[2]

Many candidates found it easier to describe a benefit than to give a drawback. The most commonly identified benefit was that the array would have global scope (and would therefore not need to be passed as a parameter), but often the descriptions given were too vague, e.g. 'can be accessed anywhere'. The correct technical vocabulary is required.

Drawbacks were poorly described. Potential side effects and resultant complexity debugging were frequently alluded to as 'accidental change' but not fully developed into complete qualified points.

There was also a frequent misconception that you cannot have multiple variables with the same name, which is not true. When a local variable is declared with the same name as a global variable that already exists, it takes precedence within the local scope.

Question 1 (d)

(d) Describe **one** feature of an Integrated Development Environment (IDE) that can be used to help write the program **and one** feature that can be used to help test the program.

Write .....

.....

.....

Test .....

.....

.....

[4]

Most candidates achieved at least 2 marks by identifying two suitable features, but often the expansions to describe each feature were less clear. Developments in modern IDEs meant that there were many valid features, but auto-completion and breakpoints were particularly common answers.

Question 1 (e)

(e) Functions and procedures are reusable components.

Give **two** benefits of writing a program with reusable components.

1 .....

.....

2 .....

.....

[2]

Many candidates scored 1 mark, but fewer gave two clear benefits. Unqualified statements that were too vague were often given. A frequent response was 'saves time', but as an unqualified point it did not state how time was saved, e.g. 'saves time as you don't have to retype the same code/routine again'.

Candidates need to be reminded that at this level responses of a 'quicker' or 'easier' nature will not gain marks unless qualified.

Exemplar 1

1 Saves time, as code has already been tested and written so does not need to be tested and written again (when it's reused).

2 Easier to debug and test, as each component can be tested individually (against its own well-defined functionality).

Candidates are expected to fully qualify the points that they make. This response shows clear identification of points followed by qualifying statements.

Question 2 (a)

2 A computer program is being written to store data about students.

Fig. 2 shows a binary search tree that stores data about students.

Each student is represented by their ID number. The current contents of the binary search tree are:

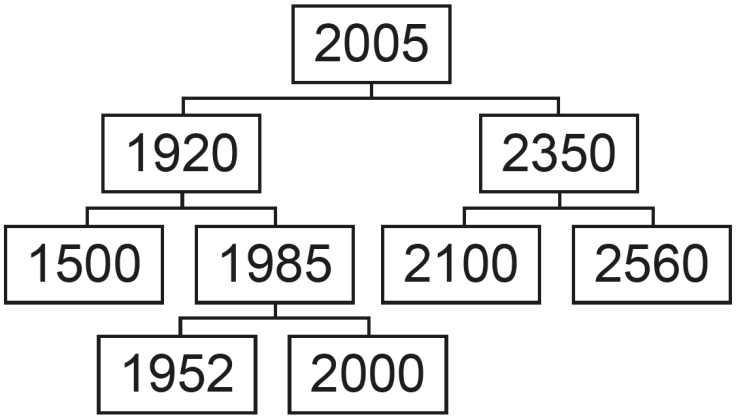


Fig. 2

(a) Identify the root node in the binary tree shown in **Fig. 2**.

.....

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

Most candidates correctly identified the root node as 2005.

Question 2 (b)

(b) Identify **two** leaf nodes in the binary tree shown in **Fig. 2**.

1 .....

2 .....

[2]

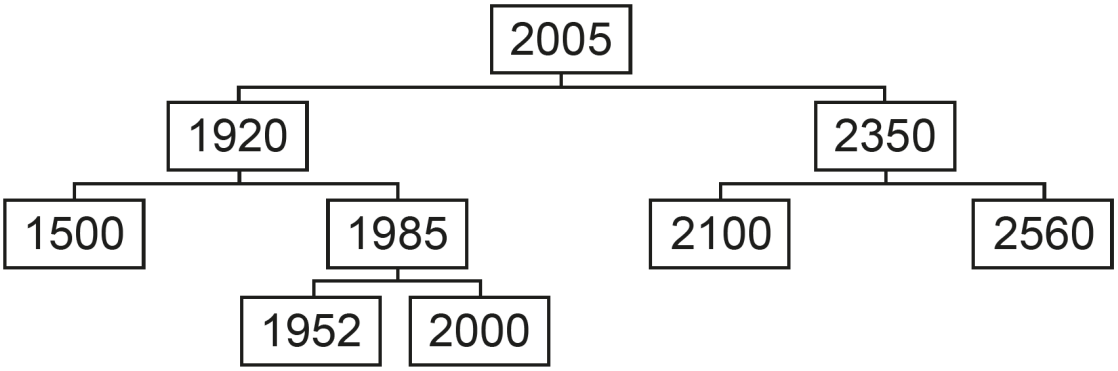
Most candidates correctly identified valid leaf nodes, but a significant number erroneously gave 1920 and 2350 as the child nodes of the root instead of identifying leaf nodes.

Question 2 (c)

(c) Four more students are added to the binary search tree shown in **Fig. 2** in this order:

1420 2050 2780 2600

Complete the binary search tree here by adding the new students to it.



[4]

Many candidates scored at least 2 marks, but many erroneously inserted 2600 before 2780, or just put 2600 as the left child node of 2560. A common mistake was to use straight lines for new child nodes rather than clearly indicating whether the new child node was a left/right child of the parent node.

Question 2 (d)*

(d)* A programmer would like to traverse the binary search tree shown in **Fig. 2**.

Compare the use of a breadth-first traversal and a depth-first (post-order) traversal on the binary search tree.

You should include the following in your answer:

- how each traversal works
- the order of the return values for each traversal.

.....

.....

.....

.....

.

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

Many candidates achieved some marks, but a few did confuse Breadth-first search (BFS) and Depth-first search (DFS), getting them the wrong way round.

Most had a much clearer understanding of BFS than DFS and were able to score marks in Level 1 for showing how a BFS would be carried out. Fewer managed to accurately show how a DFS would work. While there was an understanding that DFS would traverse leftward to the lowest leftmost node, descriptions of back tracking were not given as many marks. A common mistake made with DFS was to output the nodes in the order they were first visited rather than the order in which they are output.

Few could describe the mechanics of the algorithms with BFS using a queue and DFS using a stack. Those who did achieved the top of Level 2.

Very few candidates could give exemplar uses of BFS or DFS (e.g. deleting nodes in a tree) or compare cases where they might be used (e.g. distance of target node(s) from root), with very little evaluation. This meant there were very few Level 3 responses seen.

Question 3 (a)

3 A bubble sort will sort an array of 50 integer values called `numberArray`.

(a) State why the integer values are stored in an array instead of separate variables.

.....

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

The concept of having a single indexable identifier seemed to be poorly understood with few candidates gaining the mark. While many candidates understood that an array can be iterated through, they did not link this to the fact that elements in an array are indexable. Similarly, unqualified points such as 'an array is easier to sort' were not given marks for the same reason.

There was a common misconception that an array would use far less memory space than 50 separate variables. There would be minimal difference since each separate value would take the same amount of storage space as the corresponding value in an array. The only overhead for separate variables would be the extra pointers to the memory locations for each, compared to one pointer for an array.

Question 3 (b)

(b) This bubble sort algorithm is written to sort `numberArray` into ascending numerical order.

Complete this bubble sort algorithm.

```
arrayLength = .....
tempValue = 0
do
  flag = false
  for y = 0 to arrayLength - .....
    if numberArray[y] > numberArray[y + 1] then
      ..... = numberArray[y]
      numberArray[.....] = numberArray[y + 1]
      numberArray[y + 1] = .....
      flag = true
    endif
  next y
until flag == false
```

[5]

Many candidates scored at least 1 mark for correctly initialising the `arrayLength`, although some erroneous `length.arrayLength` rather than `arrayLength.length` answers were seen. If candidates are assuming the existence of inbuilt methods they should reference them in the correct OOP way.

Very few candidates achieved the second marking point for the loop. A very common off-by-one error was seen with the value `1` given. If `arrayLength` was set to 50 this would cause a run time error for an out-of-bounds reference when the loop ran.

Some candidates tried to swap the array values directly in a Pythonic style that was not suitable for a pseudocode solution in context, and some used incorrectly formatted variable identifiers.

Variable identifiers

Valid identifiers must be single words. In a number of instances, it was clear that `tempValue` was given as `temp Value`. Where candidates give an answer that clearly has spaces within an intended identifier name no marks will be given.

Question 3 (c)

(c) One section of `numberArray` is shown here.

2	12	1	9	3	5	15	7

A second sorting algorithm that could be used to sort this data is a merge sort.

Show how a merge sort will sort this section of the array `numberArray` into ascending numerical order.

.....

.....

.....

.....

.....

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

Many candidates presented very clear solutions that used the values in `numberArray` to construct clearly annotated diagrams. Textual prose responses tended to either not refer at all to the given data set, or mentioned it only partially, so lost marks. There was occasional confusion with other sorting algorithms, but this was seen relatively infrequently,

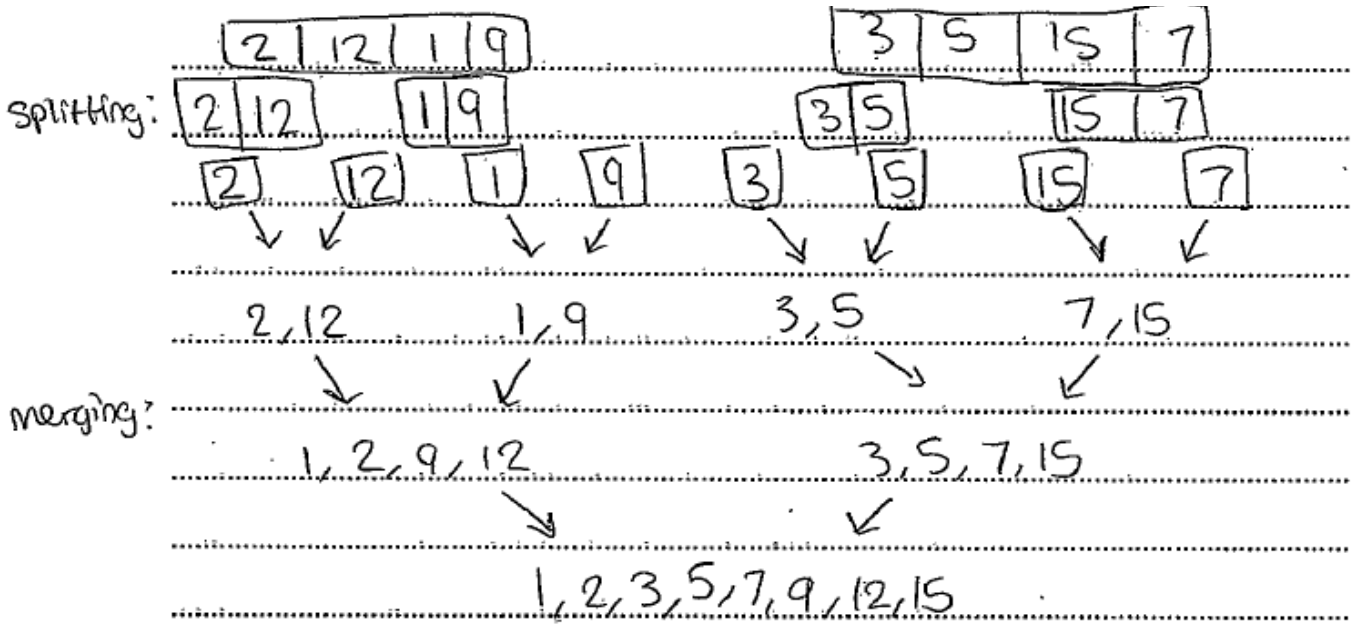
Where errors were made candidates did not split the data to the atomised level. Many did not fully understand how a merge sort works by merging two separate lists of ordered values together but performed in-place sorts in sub-lists.

Misconception

The merge phase in a merge sort takes two separate lists that are already ordered. Values in those two separate lists are taken and merged in sequence to form a new list. E.g. List 1 [1, 9] and List 2 [2, 10] are taken and merged 1 from List 1, 2 from List 2, 9 from List 1 and then 10 from List 2, into a new List.

Values in lists [1, 9] and [2, 10] are not placed in a list [1, 9, 2, 10] and then sorted in-situ. A significant number of candidates thought that a merge sort split lists up until values were in pairs and showed [15, 7] going directly to [7, 15] (i.e. an in-situ sort) without first being atomised into two separate lists of [15] and [7] and then being merged to give [7, 15].

Exemplar 2



Candidate responses to questions that require 'showing' how data sets are ordered by sorting algorithms are best tackled by using clearly annotated diagrams such as this exemplar response.

Question 3 (d)*

(d)* Another sorting algorithm is insertion sort.

The number of values stored in the array `numberArray` has been reduced to 10.

Compare the use of bubble, merge and insertion sorts on the array `numberArray`.

You should include the following in your answer:

- how each algorithm works
- the Big O complexities for each algorithm
- the suitability of each algorithm for sorting the 10 values.

.....

.....

.....

.....

.....

..... **[12]**

Most candidates could describe the basic elements of each of the bubble, merge and insertion sort, although some had difficulty remembering insertion sort and confused it with quick sort.

Many candidates struggled to show accurate knowledge of the Big O time complexity values for best/average/worst case for the three algorithms, which is a learnt response. Some did give very good examples of where bubble and insertion sort would give best/worst case times.

Very few candidates could cite space complexity, and where they did a number thought bubble/insertion were space complexity O(n) because there are n elements rather than O(1) which means constant with no extra memory overhead.

There was some evaluation of the number of items being sorted in most cases.

Overall, many responses were clustered in Level 2, but a pleasing number of candidates achieved Level 3 with clear and detailed descriptions, accurate Big O values, and an evaluation of the size of the data set being sorted.

Question 4 (a) (i)

4 A programmer is developing an aeroplane simulator. The user will sit in a cockpit and the simulated environment will be displayed on screens around them.

(a) The programmer uses computational methods to design a solution for the program.

(i) Complete the table by writing a definition for each computational method.

Computational Method	Definition
Abstraction	
Decomposition	

[2]

While many candidates accurately recited definitions for abstraction and decomposition there were equally many that presented very vague and unqualified responses. E.g. For abstraction: 'Simplifying a problem' without specifying how it was simplified was insufficient. For decomposition: 'Breaking a problem into smaller parts' – without specifying that the 'smaller parts' are sub-problems rather than saying sequences of instructions or loops was insufficient.

Question 4 (ii)

(ii) Give **three** potential differences between the abstracted aeroplane simulator and reality.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

Many candidates gave detailed and relevant examples that were suitable within the context of a flight simulator. However, there were many unqualified responses that were too vague. Unqualified responses such as 'terrain' did not go far enough to explain how the terrain in the simulation would differ from reality. To gain marks candidates had to make it clear exactly how the difference identified differed between reality and the simulation.

Question 4 (iii)

(iii) Identify **two** reasons why abstraction is used when designing a solution to the problem.

1 .....

.....

2 .....

.....

[2]

It was noticeable that many candidates reiterated definitions of abstraction rather than identifying the actual reasons why abstraction is used, thus scoring no marks for not answering the question. Again, there were also many vague and unqualified responses. Responses such as 'focus on key aspects' did not identify the reason why abstraction is used in the design, whereas clearing specifying 'focus on key aspects allows coding/development time to be saved' would make the reason clear.

Question 4 (b)

(b) Describe how caching can be used in the aeroplane simulator.

.....

.....

.....

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

Responses to caching were better than those seen in recent papers and many candidates managed to either describe the concept of caching or give an example of something that would realistically be cached. Far fewer managed to do both in detail.

Question 5 (a) (i)

5 **Fig. 5** shows a graph data structure representing a small section of a parcel delivery network. Each node represents an address where deliveries need to be made. The edges show the possible routes and distances between these deliveries.

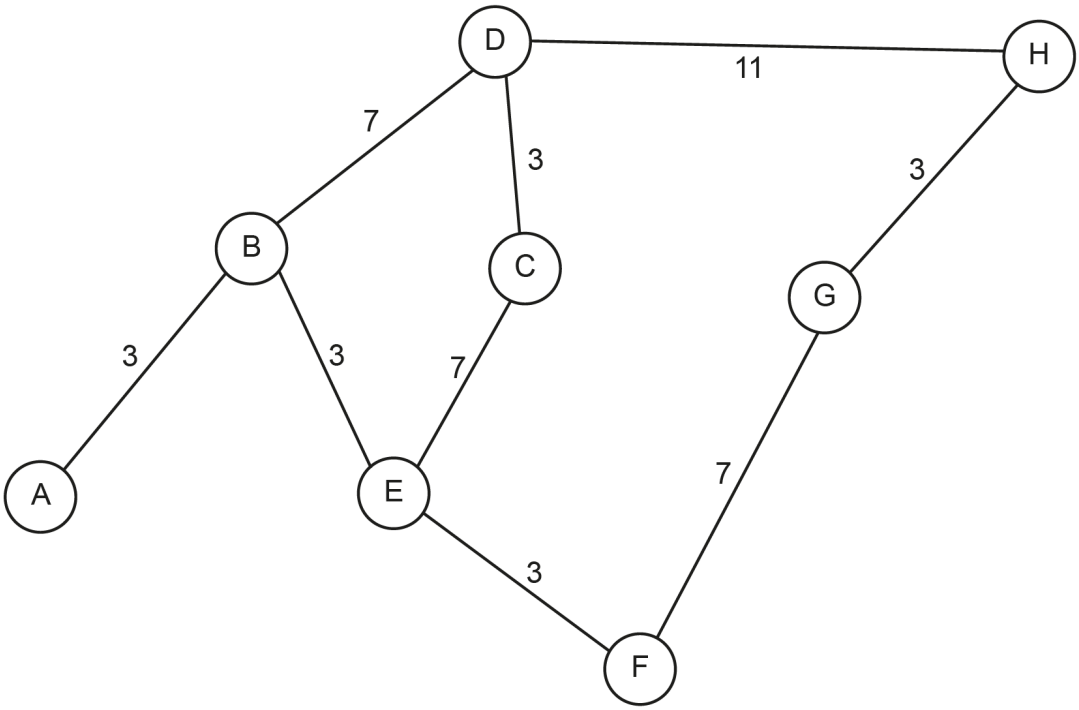


Fig. 5

(a) **(i)** State **one** way a directed graph is different to an undirected graph.

.....

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

Very few candidates had a clear grasp of the relevant terminology to clearly identify that an edge/arc is given a specific direction in a directed graph. Many responses either erroneously talked about paths between nodes (too vague as there can be many routes between two nodes located in different parts of a graph) or graphs being weighted.

Question 5 (a) (ii)

(ii) State **one** way a graph data structure is different to a tree data structure.

.....

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

Most candidates gave clear responses that indicated either that a tree has a root whereas a graph does not, or graphs can contain cycles whereas trees do not. There were a surprising number of candidates who thought trees were limited to a maximum of two child nodes, showing a lack of understanding of generalised tree structures with hierarchies (e.g. directory tree).

Question 5 (b)

(b) Give **one** reason why the graph is a visualisation of the problem.

.....

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

Very few candidates were able to give suitable answers within the context of the problem. The question was asking why the graph in Fig 5 was a visualisation. Few candidates identified that it was because the letters at the nodes represented delivery addresses, while the weights on the edges represented the road distances between the addresses. Most candidates gave descriptions of visualisation in general rather than answering in context.

Question 5 (c) (i)

(c) (i) Show how Dijkstra's algorithm can be used on the graph shown in **Fig. 5** to find the shortest path from the start node A and the end node H.

You should state the nodes on the final path and the overall distance. Show your working.

You may choose to use the table below to give your answer.

.....

.....

.....

.....

.....

Node	Distance travelled	Previous node

Final path: .....

Distance: .....

[6]

Most candidates gave the final path and the total distance correctly by inspection if nothing else. All nodes are initially set to infinity, so A is updated to 0 and has a distance 0 from A as the start node and many candidates missed this. Those who gave an answer by inspection wrote ABEFGH without knowing Dijkstra's algorithm but gained some marks by giving the distances to BEFGH along the way, but distances to nodes C and D were omitted. Few candidates clearly showed that the initial calculation for the path distance to H (from D, distance 21) was later updated and overwritten with the more optimal path length from G with distance 19.

Question 5 (c) (ii)

(ii) Give a similarity and difference between the performance of Dijkstra's algorithm and the performance of A* algorithm.

Similarity .....

.....

Difference .....

.....

[2]

Those candidates who scored well in c(i) frequently gained full marks for describing the similarities and differences between Dijkstra's algorithm and the A* algorithm. The most common responses were that both give the shortest path and that A* uses heuristics.

Question 5 (d) (i)

(d) (i) State why performance modelling is used to test a system.

.....

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

The concept of stress testing a system by simulating heavy loads seemed to be poorly understood by most candidates. Many candidates often confused this with the general testing of a system to find bugs.

Question 5 (d) (ii)

(ii) Describe how performance modelling can be used in the delivery system.

.....

.....

.....

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

There was much confusion over testing a system in general to determine whether an individual route was effective in the graph given, and the actual performance of the system as a whole in terms of the overall time taken to actually perform the calculations required. The majority of candidates did not appreciate that performance modelling looks at the effect of escalating loads on a system.

Question 6 (a)

6 A card game uses a set of 52 standard playing cards. There are four suits; hearts, diamonds, clubs and spades. Each suit has a card with a number from; 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13.

The card game randomly gives 2 players 7 cards each. The unallocated cards become known as the deck.

The players then take it in turns to turn over a card. A valid move is a card of the same suit or the same number as the last card played.

The winner is the first player to play all of their cards.

(a) One component of the game is checking if a move is valid.

Identify **three** other components of the game.

1 .....

2 .....

3 .....

[3]

The specification requires candidates to be able to identify elements of computational thinking. As such, candidates are expected to be able to think procedurally and to be able to identify the components of a problem. While analysis of the problem given in context led most candidates to identify valid components, many struggled to read the scenario and to give relevant points. For instance, many reiterated aspects of checking if a move was valid, which was already given in the question.

Question 6 (b) (i)

(b) A function, `checkValid()`, takes the card the player has selected, and the last card played as parameters.

It returns `true` if the player's move is valid and returns `false` if the player's move is not valid.

(i) State the reason why `checkValid()` is a function and not a procedure.

.....

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

Most candidate correctly identified that `checkValid()` was a function because it returned a value. Some candidates erroneously stated 'because it has an output' which did not differentiate a function from a procedure since both can print an output as a side effect.

Question 6 (b) (ii)

(ii) The programmer will use a branching (selection) construct to make decisions.

Describe the decisions that will be made in the `checkValid()` function and how these change the return values.

.....

.....

.....

.....

.....

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

Many candidates were able to think computationally to identify the points in a solution where a decision had to be taken. Occasionally some candidates did not specify the specific result that would be returned from the function after determining if a card was valid or not. There were also occurrences of incorrect logic such as returning valid using an AND condition on the same number and same suit clauses, rather than executing them in sequence.

Question 6 (c)

(c) The cards are held in the 2D array `cards`. The first index stores the card number and the second index stores the suit, both as strings.

Write a pseudocode statement or program code to declare the array `cards`.

.....

.....

.....

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

Many candidates gained a mark by initialising the identifier `cards`, but fewer gained the second mark for correctly setting it to be a two-dimensional structure. Many obscure forms of syntax were observed, but marks was given if it was clear that the structure was two-dimensional, however, for many responses, it was clear that a one-dimensional list had been initialised.

Question 7 (a) (i)

7 A program uses the recursive function `calculate()`. The function is written in pseudocode.

```
1.  function calculate(number : byVal)
2.      if number == 1 then
3.          return number
4.      else
5.           return number + calculate (number - 1)
6.      endif
7.  endfunction
```

(a) (i) Give the line number in the algorithm `calculate()` where a recursive call is made.

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

Nearly all candidates gave the correct answer line `5`.

Question 7 (a) (ii)

(ii) State **two** features of any recursive algorithm.

Feature 1 .....

.....

Feature 2 .....

.....

[2]

Most candidates knew that a recursive algorithm is self-referential and calls itself, but some candidates were too vague specifying that it 'calls a function'. Candidates often found it harder to gain the second mark and some gave answers not related to the question such as explanations of how recursion uses stack frames during execution. Technical vocabulary is important, and some candidates did not make it clear that recursion has a base case. Those that stated a stopping/terminating condition needed to qualify their response to say that these conditions stopped/halted the recursion. Where candidates just wrote 'stopping condition' it was too vague as it was unqualified since they could have been talking about any conditional loop.

Question 7 (b)

(b) Trace the recursive function `calculate()` and give the final return value, when the following function call is run:

```
calculate(5)
```

You may choose to use the table below to give your answer.

.....

.....

.....

.....

.....

Function call	number	return
`calculate(5)`		

[5]

Many candidates continue to struggle with recursion as a concept and so had little idea how to trace and unwind a call to a recursive function. Some got to the last call and the base case and could return 1 or unwind one step further to gain the first 2 marks. Fewer achieved all 5 marks.

Question 7 (c)

(c) Give the pseudocode function call that would return 55 from the recursive function `calculate()`.

.....

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

Those candidates who scored full marks in 7b had little difficulty giving the correct call `calculate(10)`. Some candidates just wrote 10, which did not answer the question. Those who showed little understanding of recursion in 7b rarely gave the correct answer in 7c.

Section B overview

Section B requires candidates to tackle a longer scenario-based question that will require reference to Object Oriented Programming. Candidates will be expected to be able to both complete code and to provide pseudocode / programming code solutions to a number of problems. Prior practical experience is essential for candidates to tackle this section of the paper confidently.

Assessment for learning

Candidates tackling Section B benefit from having extensive practical experience programming using the Object Oriented Programming (OOP) paradigm.

Implementing stack and queue classes is a useful exercise to reinforce theoretical knowledge of associated operations alongside discussion of OOP concepts.

Past paper questions provide a suitable source of stimulus material and candidates could implement a stack class to simulate all parts of Questions 8a and 8b in this paper.

Question 8 (a) (i)

8 A computer uses a stack data structure, implemented using an array, to store numbers entered by the user.

The array is zero based and has 100 locations.

(a) **Fig. 8** shows the current contents of the stack and the first 9 locations of the array.

pointerValue	5

Index	Data
8	
7	
6	
5	
4	1
3	23
2	6
1	5
0	10

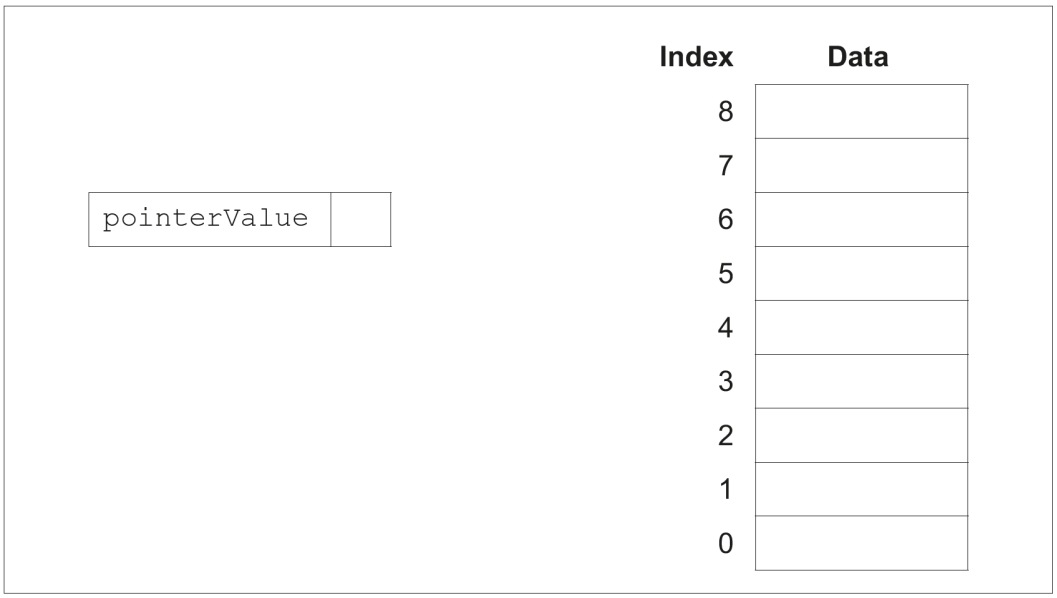
Fig. 8

(i) The function `pop()` removes an item from the stack.

The function `push()` adds an item to the stack that is passed in as a parameter.

Show the contents of the stack and pointer from **Fig. 8** after the following subroutines calls have run.

```
pop()
pop()
push(3)
push(6)
push(7)
```



[2]

Most candidates scored full marks for this question, but the most common error was the omission of the `pointerValue` or the setting of the `pointerValue` to 5 to point to the last element in the stack instead of the next free location at the top of the stack.

Question 8 (a) (ii)

(ii) State the purpose of `pointerValue`.

.....

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

Many candidates answered the question clearly specifying a pointer to the next free index in the stack, but some candidates gave vague responses such as 'location of next item'.

Question 8 (b) (i)

(b) The stack is programmed as an object using object-oriented programming. The design for the class, its attributes and methods are shown:

class: stack
attributes: private stackArray : Array of integer private pointerValue : integer
methods: new() function pop() function push(value)

(i) The method `pop()` returns the next value in the stack, or `-1` if the stack is empty.

Complete the pseudocode method `pop()`.

```
public function pop()
  if pointerValue == ..... then
    return .....
  else
    pointerValue  = pointerValue .....
    returnValue = stackArray[.....]
    return .....
  endif
endfunction
```

[5]

Most candidates gained some marks with many gaining full marks. The most common errors were writing identifiers for `pointerValue` and `returnValue` with spaces in, returning values as strings, or returning True/False instead of -1 as required in the question.

Question 8 (b) (ii)

(ii) The method `push()` accepts an integer as a parameter and adds it to the top of the stack unless the stack is already full.

If the push is successful the method returns `true`.

If the push is unsuccessful due to the stack being full the method returns `false`.

Write the method `push()` using either pseudocode or program code.

.....

.....

.....

.....

.....

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

Candidates produced a higher standard of pseudocode this session and many scored most if not full marks for a standard stack push routine.

Common errors included omitting a parameter and/or getting a user input as the value to place into the stack, returning strings or printing the return values, and off-by-one errors when testing to see if the `stackPointer` was at the top of the stack to determine if the stack was full.

It was noticeable that a number of students who were only familiar with Python gave list append type solutions rather than using the array and pointers as per the implementation given.

Another common error was incrementing the value of the stackPointer before the parameter was assigned to `stackArray` at the `stackPointer` index, which was frequently seen when candidates did not know that the `stackPointer` actually pointed to the index of the next free space in the stack.

Exemplar 3

```
function push(num)
    if pointerValue == stackArray.length then
        return false
    else
        stackArray[pointerValue] = num
        pointerValue++
        return true
    endif
endfunction.
```

Candidates are encouraged to present pseudocode solutions with clear indentation to aid readability. No specific language/syntax is expected, but the logic of the solution must be clear.

Question 8 (b) (iii)

(iii) The main program initialises a new object of type stack with the identifier `mathsStack`.

Write pseudocode or program code to declare the object.

.....

.....

.....

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

Many candidates struggled to answer this question because they lacked practical experience of OOP that showed a lack of familiarity in terms of creating instances of a class. Some candidates tried to declare `mathsStack` as a procedure or class, and many got it the wrong way round and actually declared a new identifier called `stack` as an instance of the class `mathStack`.

Question 8 (b) (iv)

(iv) The main program needs to:

- take numbers as input from the user
- push them onto the stack `mathsStack` until the stack is full
- output an appropriate message if the stack is full.

Complete the pseudocode algorithm to meet these requirements.

```
returnValue = true

while returnValue == ..........

     returnValue = mathsStack. ..........(input("Enter
     Number"))

     if returnValue == .......... then

          .......... ("Stack full")

     endif

endwhile
```

[4]

The majority of candidates scored some marks but relatively few gained full marks. Return was seen instead of print/output when the question did, on this occasion ask for the message to be output (as the code was specified as being in the main program rather than a subroutine).

Question 8 (b) (v)

(v) The main program also needs to:

- remove one item from the stack at a time and add this to a total
- output the total every time an item is removed
- stop removing items when either the stack is empty, or 20 items have been removed.

Write pseudocode or program code to meet these requirements.

.....

.....

.....

.....

.....

..... **[8]**

The standard of pseudocode / programming code was better than in previous sessions and most candidates made a reasonable attempt to pop 20 values from the stack. Many candidates found it difficult to reference the class methods correctly or made assumptions about the existence of other methods that were not provided within the scenario (e.g. `.full()`, `.empty()`, `.length()`, `.remove()`) that should not be presumed to exist.

Those who had little understanding of encapsulation often tried to access class attributes such as `stackPointer` directly to manipulate `mathsStack`, rather than using methods to interact with the instance of the stack.

When candidates did use the `.pop()` method to retrieve a value from `mathsStack` they frequently did not store the result for later use to check whether then stack was empty.

Question 8 (c) (i)

(c) The program is amended to include the use of several queue data structures.

(i) Describe how an array can be used to implement a queue data structure.

.....

.....

.....

.....

.....

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

Many candidates identified the need to have a head/tail pointer but struggled to gain more than 1 mark by expanding on how enqueue and dequeue operations would be implemented.

Some candidates continued to talk about push/pop operations for a queue rather than enqueue/dequeue and often gave properties of a queue in general such as First In First Out rather than answering the question.

Question 8 (c) (ii)*

(ii)* Discuss the use of object-oriented programming and procedural programming to create and manipulate the queue data structures.

You should include the following in your answer:

- the features of object-oriented programming
- the features of procedural programming
- the benefits of using object-oriented instead of procedural programming when creating several queue structures.

.....

.....

.....

.....

.....

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

Many candidates were able to identify some elements of OOP and Procedural programming to achieve a Level 1 response or were able to describe features in detail for a Level 2 response. Far fewer were able to apply this to the specific context to achieve a Level 3 response.

Those with good knowledge of OOP stood out in terms of giving clear evaluations of multiple queues generated from instance of the class, encapsulation to reduce side effects and possibilities for inheritance for different types of queues.

Many did not describe the necessary creation of enqueue and dequeue subroutines in procedural programming for each separate queue or the need to pass queues to or returning queues from subroutines.

Misconception

There was a lot of confusion between inheritance and instantiation, e.g. "when creating several queues you can use inheritance, so all queues inherit attributes and methods".

Candidates need to be clear that each instance of a class is assigned the attributes of the class and has access to all associated methods.

Supporting you

Post-results services

If any of your students' results are not as expected, you may wish to consider one of our post-results services. For full information about the options available visit the OCR website.

Keep up-to-date

We send a weekly roundup to tell you about important updates. You can also sign up for your subject specific updates.
If you haven't already, sign up here.

OCR Professional Development

Attend one of our popular CPD courses to hear directly from a senior assessor or drop in to a Q&A session. Most of our courses are delivered live via an online platform, so you can attend from any location.

Please find details for all our courses on the relevant subject page on our website or visit OCR professional development.

Signed up for ExamBuilder?

ExamBuilder is the question builder platform for a range of our GCSE, A Level, Cambridge Nationals and Cambridge Technicals qualifications. Find out more.

ExamBuilder is **free for all OCR centres** with an Interchange account and gives you unlimited users per centre. We need an Interchange username to validate the identity of your centre's first user account for ExamBuilder.

If you do not have an Interchange account please contact your centre administrator (usually the Exams Officer) to request a username, or nominate an existing Interchange user in your department.

Active Results

Review students' exam performance with our free online results analysis tool. It is available for all GCSEs, AS and A Levels 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.

Find out more.

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
/ocrexams
/ocrexams
/company/ocr
/ocrexams

We really value your feedback

Click to send us an autogenerated email about this resource. Add comments if you want to. Let us know how we can improve this resource or what else you need. Your email address will not be used or shared for any marketing purposes.

I like this **I dislike this**

Please note – web links are correct at date of publication but other websites may change over time. If you have any problems with a link you may want to navigate to that organisation's website for a direct search.

CAMBRIDGE
UNIVERSITY PRESS & ASSESSMENT