1. A 1-dimensional array stores a set of numbered cards from 0 to 7. An example of this data is shown in Fig in 4.1

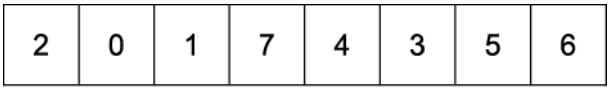


Fig 4.1

The programmer wants to search for a specific card in the array.

State whether a binary search or a linear search would be the most appropriate method to search for a specific card, and justify your answer.

Search method _____

Justification _____

_____

_____

_____ [3]

2. A programmer is developing an ordering system for a fast food restaurant. When a member of staff inputs an order, it is added to a linked list for completion by the chefs.

The user needs to be able to search for, and find, a specific order number.

State an appropriate search algorithm that could be used, and justify your choice against an alternative Search algorithm.

Appropriate Search Algorithm ----------

Justification ----------

---

[3]

3. The target integer 8 exists in a list of integers 1, 4, 6, 9, 8, 12, 15 but is not found during a binary search. There are no errors in the code.

(i) Give the reason why the target integer 8 is **not** found.

---

[1]

(ii) Identify and describe an alternative search algorithm that could be used.

---

[3]

4.

A country's national rail operator provides an app for customers to purchase tickets. An array is used to store the names of the stations on the network. Customers must enter a departure station into the app.

The current contents of the array are shown:

Cavalry	Bridge	Walkway	Museum	Monument	Council House	Theatre	Cinema

A linear search is used to check if the entered departure station exists in the array.

(i) Identify **one** precondition that is needed before a binary search could be used with the station array.

---

[1]

(ii) A user enters the departure station 'Bridge Heights'

Explain how a linear search would check if the departure station exists in the array.

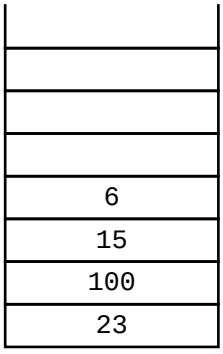
---

[4]

5. A computer program stores data input on a stack named `dataItems`. The stack has two subprograms to add and remove data items from the stack. The stack is implemented as a 1D array, `dataArray`.

Sub-program	Description
`push()`	The parameter is added to the top of the stack
`pop()`	The parameter is added to the top of the stack

The current contents of `dataItems` are shown:



As an array, the data in `dataArray` is sorted and then searched for a specific value.

(i) The data in `dataArray` is sorted into ascending order using an insertion sort.

The current contents of `dataArray` are shown:

100	22	5	36	999	12

Show the steps of an insertion sort on the current contents of the array `dataArray`.

---

[5]

(ii) The array `dataArray` can now be searched using a binary search.

Describe the stages of a binary search on an array of size n.

---

[7]

(iii) The array has 50 items.

The function, `searchItem()`, performs a linear search for a data item.

```
function searchItem(dataItem)
     for count = 0 to 49
          if dataArray[count] == dataItem then
               return(count)
          endif
     next count
     return(-1)
endfunction
```

Rewrite the function using a while loop.

---

[4]

6(a). Describe the steps involved in a binary search to find the value 47 in the list below.

4, 7, 8, 21, 46, 47, 51

[4]

(b). A programmer has been tasked with writing a function that uses a binary search to return a Boolean value. The function should return `true` if the target integer is found in a list of integers. Using pseudocode, write an algorithm for the function.

[8]

7. Show how a binary search would be performed on the array shown in Fig. 4.2 to find the value ‘duck’.

wolf	monkey	lion	iguana	goat	giraffe	frog	elephant	duck

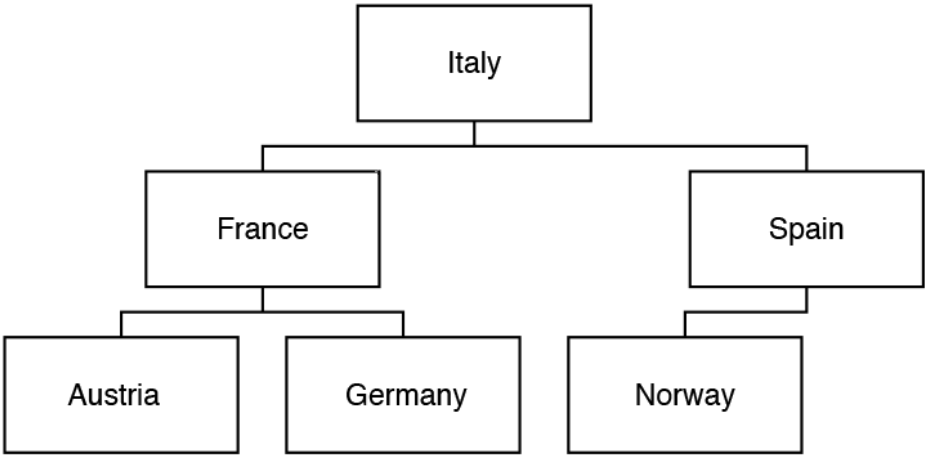
Fig. 4.2

---

[3]

8. A program stores entered data in a binary search tree.

The current contents of the tree are shown:



A pseudocode algorithm is written to search the tree to determine if the data item “Sweden‘ is in the tree.

The function `currentNode.left()`returns the node positioned to the left of `currentNode.`

The function `currentNode.right()`returns the node positioned to the right of `currentNode.`

```
function searchForData(currentNode:byVal, searchValue:byVal)
     thisNode = getData (.................................)
     if thisNode == ..................................... then
          return .............................................
     elseif thisNode < searchValue then
          if currentNode.left() != null then
               return (searchForData(currentNode.left(), searchValue))
          else
               return .................................
          endif
     else
          if ..................................... != null then
               return (searchForData(currentNode.right(), searchValue))
          else
               return false
          endif
     endif
endfunction
```

(i) Complete the algorithm.

[5]

(ii) The algorithm needs to be used in different scenarios, with a range of different trees.

Identify **two** preconditions needed of a tree for this algorithm to work.

1 ----------

2 ----------

[2]

END OF QUESTION PAPER

Question			Answer/Indicative content	Marks	Guidance
1			1 mark for linear search, 2 for justification Justification: • The array is not sorted (1) • Linear does not need ordered / linear goes through all elements from beginning / binary needs a sorted array (1)	3	
			Total	3	
2			Algorithm, max 1 • linear Justification, 1 mark per bullet to max 2 • Items do not have to be in a specific order • Binary needs items in order	3 AO1.1 (1) AO2.1 (2)	No marks for justification if linear has not been identified Examiner's Comment: Many candidates correctly identified a linear search and could justify the need for it. However, a lot of candidates did answer binary search without appreciating that the data set needed to be in order first.
			Total	3	
3		i	• The integers in the list are unsorted (1)	1	Examiner's Comments Most candidates correctly identified that a list needs to be in order for a binary search to be applied. However, a worrying number of candidates thought that it could not be applied because there was an even number of items in the list, and hence no clear mid-point.

Question			Answer/Indicative content	Marks	Guidance
		ii	Identification (Max 1) • Perform a linear search Description (Max 2) • starting at the first element / each item is checked... • until value is found • or end of list reached and not found	3	Accept serial Examiner's Comments Many candidates identified a linear search, but fewer could give a full description as to how a linear search operates. A number of candidates confused searching with sorting algorithms.
			Total	4	
4		i	The data needs to be sorted / in alphabetical order	1 AO2.1 (1)	Examiner's Comments Most candidates knew that the list had to be sorted before a binary search could be performed.
		ii	1 mark per bullet to max 4 • Start at the first item (Cavalry) • Compare with departure station 'Bridge Heights' • If matched, report found • Otherwise continue to the next item in list (Bridge) • Continue until item found, or end of list reached… • and then False returned	4 AO2.1 (2) AO2.2 (2)	Examiner's Comments The majority of candidates described a linear search, but some described a binary search by mistake. Where candidates did describe a linear search, they generally did so with sufficient precision.
			Total	5	

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>5</td><td></td><td>i</td><td>1 mark per row (after first row)

<table><tr><td>100</td><td>22</td><td>5</td><td>36</td><td>999</td><td>12</td><td></td></tr><tr><td>22</td><td>100</td><td>5</td><td>36</td><td>999</td><td>12</td><td>1 mark</td></tr><tr><td>5</td><td>22</td><td>100</td><td>36</td><td>999</td><td>12</td><td>1 mark</td></tr><tr><td>5</td><td>22</td><td>100</td><td>36</td><td>999</td><td>12</td><td>1 mark</td></tr><tr><td>5</td><td>12</td><td>22</td><td>36</td><td>100</td><td>999</td><td>1 mark</td></tr></table></td><td>5
AO2.2
(5)</td><td><u>Examiner's Comments</u>

Candidates should be encouraged to demonstrate sorting algorithms through the use of clear diagrams that show the steps/passes in the sorting algorithm. Where candidates used verbose text, it often made it far harder to follow whether or not the correct sorting algorithm had been applied. Most candidates did implement an insertion sort, but some did describe bubble or merge sorts instead.</td></tr>
<tr><td></td><td></td><td>ii</td><td>1 mark per bullet to max 7

• Repeat
• Calculating an array midpoint…
• …by adding the array lower bound to the array upper bound, dividing by 2 and rounding
• Compare array midpoint with value to search for…
• …if equal set found flag to true
• …if array midpoint < value to search for, change lowerbound to equal midpoint + 1
• …if array midpoint > value to search for, change upperbound to equal midpoint - 1
• Until lowerbound is greater than or equal to upperbound
• Return/output found flag</td><td>7
AO1.1
(2)
AO1.2
(3)
AO2.1
(1)
AO2.2
(1)</td><td><u>Examiner's Comments</u>

The paper title is 'Algorithms and programming'. Binary search is a standard algorithm that should be fully understood by candidates, and candidates should be able to program it. Many candidates produced very vague descriptions that were far too general to credit at this level. Candidates needed to be able to discuss how the upper and lower bound pointers are used in this algorithm (or equivalent for recursive solutions).</td></tr>
</table>

Question			Answer/Indicative content	Marks	Guidance

		iii	1 mark per bullet
			
			• Setting variable to start at 0
			• Suitable while structure (endwhile or clear indentation)
			• looping 50 times
			• Incrementing the variable within the loop
			
			e.g. 1
			`function searchItem(dataItem)`
			`  count = 0`
			`  while count < 50`
			`  if dataArray(count) == dataItem then`
			`  return(count)`
			`  endif`
			`  count = count + 1`
			`  endwhile`
			`  return(-1)`
			`endfunction`
			
			e.g. 2
			`function searchItem(dataItem)`
			`  count = 0`
			`  while count < 50 and dataArray[count]!=dataItem`
			`  count = count + 1`
			`  endwhile`
			`  if count==50`
			`  count=-1`
			`  endif`
			`  return(count)`
			`endfunction`
				4
				AO1.2 (1)
				AO3.1 (1)
				AO3.2 (2)
					Examiner's Comments
					
					It was a little disappointing to see a number of candidates using variations on for loops rather than rewriting the code using a while loop as required. A significant number of candidates still struggled to demonstrate a coherent logical response that would work for something that is relatively simplistic, thus showing a lack of proficiency in coding practice.

			Total	16	

Question			Answer/Indicative content	Marks	Guidance
6	a		• Find the middle point in the list / 21 / element 4 • Compare it to the value 47, false • Is 47 greater than middle point, true New subset is 46-51 / change lower bound to 46 / element 5 • Find the middle of the new subset / 47 / element 6 Is this value equal to 47, true Search finishes	4	Some marks such as the comparison may be by implication if the candidate's logic works Must refer to the list given in the question i.e. not a generic description Examiner's Comments In general, many candidates had an understanding of how a binary search operated. Unfortunately, some gave a generic description rather than answering the specific question which required the candidate to illustrate how a binary search would operate on a specific data set. Some candidates drew concise and elegant diagrams with appropriate annotations that made their answers much clearer than those who wrote prose at great length.

Question			Answer/Indicative content	Marks	Guidance

	b		• Finding `midPoint` and correctly checking if `midPoint` value is target value ...
			• ... and if so returning `true`
			• Correctly checking that all elements have been checked ...
			• ... and if so returning `false`
			• Identify top or bottom of list ...
			• ... if top then `leftPtr` set/passed as `midPoint + 1` ...
			• ... if bottom then `rightPtr` set/passed as `midPoint - 1`
			• Correct use of indentation (AO2.1)
			
			Example iterative example
			
			`function findItem (numberArray integer[2000], targetNumber:integer, leftPtr:integer, rightPtr:integer): boolean`
			`  while (leftPtr <= rightPtr)`
			`    midPoint = (leftPtr + rightPtr) DIV 2`
			`    if (numberArray[midPoint] == targetNumber)`
			`      return true`
			`    else if (numberArray[midPoint] < targetNumber)`
			`      leftPtr = midPoint + 1`
			`    else`
			`      rightPtr = midPoint - 1`
			`    endif`
			`  endwhile`
			`  return false`
			`endfunction`
				8	Max 8 marks
					
					Note: candidates may have given a recursive algorithm and this should is perfectly acceptable.
					
					Examiner's Comments
					
					The Binary Search is one of the algorithms specifically identified in the specification that candidates need to be able to program and understand. It is a difficult algorithm to code correctly and only the most able candidates managed to produce a strong response to give the degree of accuracy required. Many candidates wrote in structured English which was not acceptable – the question specifically required a pseudocode solution. Candidates are not expected to be able to write pseudocode in the form given by OCR in the specification appendix, and variations from various programming languages were taken into account. However, the overall ability to write pseudocode proved to be a key differentiator and many candidates should aim to improve their ability to write pseudocode before the A2 examination.

			Total	**12**	
7			1 mark for each bullet • Duck is smaller than goat • Duck is less than frog/elephant • Duck is equal to duck/less than elephant so only duck left	3	
			Total	**3**	

Question			Answer/Indicative content	Marks	Guidance

8		i	1 mark per bullet to max 5
			
			`function searchForData(currentNode:byVal, searchValue:byVal)`
			`  thisNode = getData(currentNode)`
			`  if thisNode == searchValue then`
			`  return true`
			`  elseif thisNode < searchValue then`
			`  if currentNode.left () != null then`
			`  return (searchForData(currentNode.left (), searchValue))`
			`  else`
			`  return false`
			`  endif`
			`  else`
			`  if currentNode.right() != null then`
			`  return (searchForData(currentNode.right (), searchValue))`
			`  else`
			`  return false`
			`  endif`
			`  endif`
			`endfunction`
				5
				AO2.2
				(2)
				AO3.2
				(3)
					The line **`elseif thisNode < searchValue then`** should have read **`elseif thisNode > searchValue then`**
					
					If candidates attempt to correct the code and their answers are consistent with, and work with their amendment, such answers should be credited.
					
					<u>Examiner's Comments</u>
					
					Some candidates found the first two marks more difficult to access than the last three, because they did not fully understand that the parameters to the function were to be used.

		ii	• It's a binary tree • It's ordered / sorted	2 AO2.2 (2)	<u>Examiner's Comments</u> Many candidates recognised that a binary search tree was required, but some lost marks when they specified that 2 children were always required for each parent node, when it is a maximum of two children that are allowed, so that there can be 0, 1 or 2 children.
			Total	7	