0 1 The algorithm in **Figure 2** is a sorting algorithm.

- Array indexing starts at 0.
- Line numbers are included but are not part of the algorithm.

Figure 2

```
1   arr ← [4, 1, 6]
2   sorted ← false
3   WHILE sorted = false
4      sorted ← true
5      i ← 0
6      WHILE i < 2
7         IF arr[i+1] < arr[i] THEN
8            t ← arr[i]
9            arr[i] ← arr[i+1]
10           arr[i+1] ← t
11           sorted ← false
12        ENDIF
13        i ← i + 1
14     ENDWHILE
15  ENDWHILE
```

0 1 . 1 State the data type of the variable `sorted` in the algorithm shown in **Figure 2**.

[1 mark]

0 1 . 2 The identifier `sorted` is used in the algorithm shown in **Figure 2**.

Explain why this is a better choice than using the identifier `s`.

[2 marks]

0 1 . 3 Shade **one** lozenge to show which of the following contains the **false** statement about the algorithm in **Figure 2**.

[1 mark]

A The algorithm uses a named constant ⬭

B The algorithm uses indefinite iteration ⬭

C The algorithm uses nested iteration ⬭

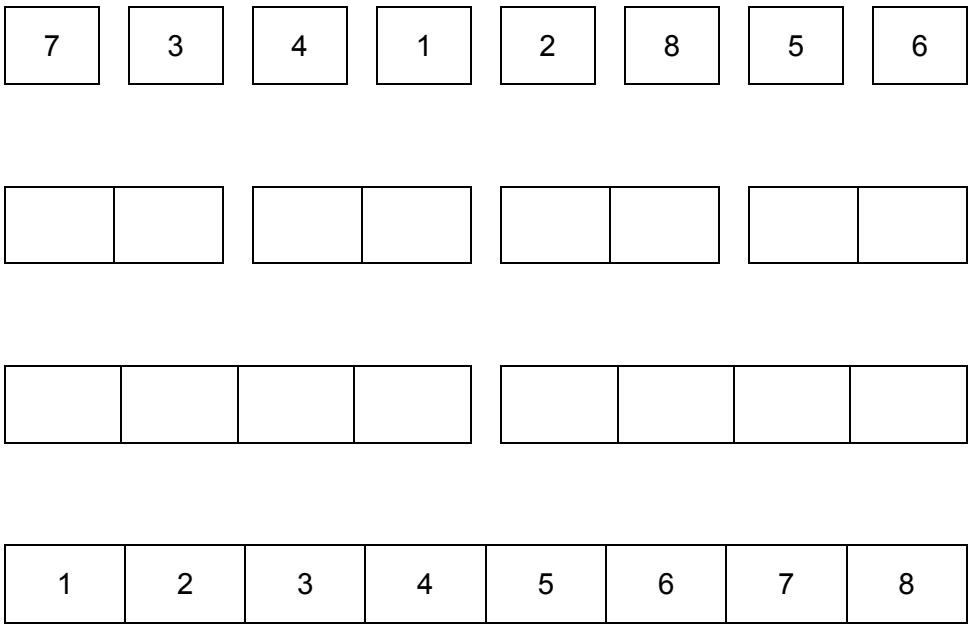
0 1 . 4 Complete the trace table for the algorithm shown in **Figure 2**. Some values have already been entered.

[6 marks]

arr			sorted	i	t
[0]	[1]	[2]			
4	1	6	false		

0 1 . 5 Fill in the values in the boxes to show how the merge part of the merge sort algorithm operates. The first and last rows have been completed for you.

[3 marks]



0 1 . 6 State **one** advantage of the merge sort algorithm compared to the sorting algorithm in **Figure 2**.

[1 mark]

0 1 . 7 A programmer implementing the algorithm in **Figure 2** decided to create it as a subroutine. Line 1 was removed and the array `arr` was made a parameter of the subroutine.

State **two** reasons why the programmer decided to implement the algorithm as a subroutine.

[2 marks]

Reason 1:

Reason 2:

0 2 Show the steps involved, for either the bubble sort algorithm **or** the merge sort algorithm, to sort the array shown in **Figure 6** so the result is `[1, 4, 5, 8]`

Figure 6

`[8, 4, 1, 5]`

Circle the algorithm you have chosen:

Bubble sort Merge sort

[4 marks]

Steps:

0 3 **Figure 13** shows an algorithm represented in pseudo-code. A developer wants to check the algorithm works correctly.

- Line numbers are included but are not part of the algorithm.

Figure 13

```
1   arr[0] ← 'c'
2   arr[1] ← 'b'
3   arr[2] ← 'a'
4   FOR i ← 0 TO 1
5       FOR j ← 0 TO 1
6           IF arr[j + 1] < arr[j] THEN
7               temp ← arr[j]
8               arr[j] ← arr[j + 1]
9               arr[j + 1] ← temp
10          ENDIF
11      ENDFOR
12  ENDFOR
```

0 3 . 1 Complete the trace table for the algorithm shown in **Figure 13**.

Some values have already been entered. You may not need to use all the rows in the table.

[6 marks]

arr			i	j	temp
[0]	[1]	[2]			
c	b	a			

0 3 . 2 State the purpose of the algorithm.

[1 mark]

0 3 . 3 **Figure 13** has been included again below.

Figure 13

```
1   arr[0] ← 'c'
2   arr[1] ← 'b'
3   arr[2] ← 'a'
4   FOR i ← 0 TO 1
5      FOR j ← 0 TO 1
6         IF arr[j + 1] < arr[j] THEN
7            temp ← arr[j]
8            arr[j] ← arr[j + 1]
9            arr[j + 1] ← temp
10        ENDIF
11     ENDFOR
12  ENDFOR
```

An earlier attempt at writing the algorithm in **Figure 13** had different code for **lines 4** and **5**.

Lines 4 and **5** of the pseudo-code were:

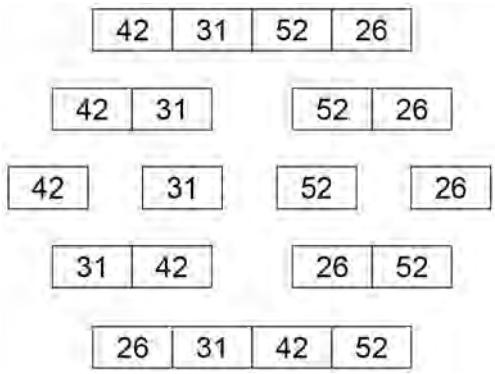
```
FOR i ← 0 TO 2
   FOR j ← 0 TO 2
```

Explain why the algorithm did not work when the value 2 was used instead of the value 1 on these two lines.

[1 mark]

0 4 **Figure 7** shows a merge sort being carried out on a list.

Figure 7



Explain how the merge sort algorithm works.

[4 marks]

0 5 Fill in the blank arrays to show the steps involved in applying the bubble sort algorithm to the array `[3, 5, 1, 4, 2]`. You only need to show the missing steps where a change is applied to the array.

[5 marks]

3	5	1	4	2
1	2	3	4	5