Unit 7: Data Structures
(AS Content)

Marks: /42

Answer **all** the questions.

1. Data structures may be described as static or dynamic.

(i) State the meaning of the term static.

--- **[1]**

(ii) State **one** type of data structure that is always considered to be static.

--- **[1]**

(iii) State the meaning of the term dynamic.

--- **[1]**

(iv) Give **one disadvantage** of using a dynamic data structure.

--- **[1]**

2(a). The organisers of an international football competition are planning to use a large electronic score board to display information to spectators in the stadium. The board can display three lines of text of 15 characters each.

The program stores the text to be displayed in an array called Board, so that

- Board(1,1) contains the letter in the top left corner of the display board
- Board(3,15) contains the letter in the bottom right corner of the display board.

A module in the program updates the display every time the contents of this array are changed.

State the identifier, number of dimensions and most appropriate data type of the array Board.

Identifier ____________________________________

Number of dimensions ____________________________________

Most appropriate data type _________________________________

[3]

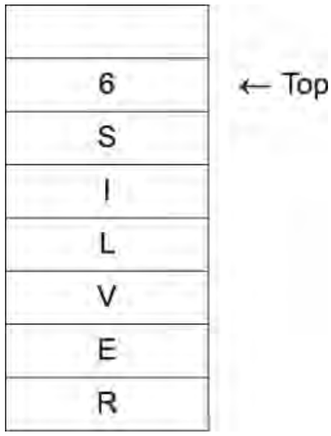
(b). The program contains a module which clears the display using a routine to insert a space in each element of the array using the following algorithm.

Complete this algorithm by filling in the blanks.

```
PROCEDURE ClearDisplay
  FOR Row = 1 TO 3
    FOR Column = 1 to ...........
      Board( Row, ...........) = " "
    NEXT Column
  NEXT ..........
END PROCEDURE
```

[3]

3. A stack, in shared memory, is being used to pass a single variable length ASCII string between two sub-systems. The string is placed in the stack one character at a time in reverse order with the last byte holding the number of characters pushed i.e. the text “SILVER” would be held in the stack as:



Use pseudocode to write a procedure that will take a text string passed to it and push it to the stack in the format defined above. You may assume any given input will fit in the stack.

[6]

4(a). A program stores a queue of mathematical questions to be asked to a user. The questions are asked in the order they are added. Once a question has been asked it cannot be asked again. New questions are continually added to the end of the queue.

The program will use a non-circular queue, `questions,` (implemented using an array) to store the questions.
The pointer, `head`, stores the index of the first element in the queue.
The pointer, `tail`, stores the index of the last element in the queue.

Fig. 4.1 shows an example of the data in the queue. `head` is currently 0, `tail` is currently 4.

"2*3"	"1+4"	"3–1"	"10/2"	"3+6"			

Fig. 4.1

(i) Show the contents of the queue shown in Fig. 4.1, after the following code is run.

```
add("6+1")
```


[2]

(ii) State the values stored in `head` and `tail` after the code in **part (i)** has run.

`head` ....

`tail` ....

[2]

(b). Complete the following algorithm, to ask the user to input a new question and then either add it to the queue, or report that the queue is full.

```
procedure add()
    maxElements = 10
```

```
endprocedure
```

[4]

(c). Describe why a queue is a suitable structure for this program.

--- [3]

(d). Complete the following algorithm, to remove, and output, the first element in the queue.

```
procedure remove()
```

```
endprocedure
```

[4]

5(a). Four in a Row is a game where two players drop coloured discs into a grid, with the aim to get four of their own colour in a row.

Each player is given a set of coloured discs, red (R) or yellow (Y). The players take it in turns to drop their disc into a column in the grid. The disc drops down to the lowest available space in that column.

(i) * The programmer is writing a new version of the game, where each player removes one disc from the bottom row of the grid before a new move is made.

In the example below, player R removes one disc from column 2 (Before) and places one in column 4 (After).

Before

	0	1	2	3	4	5	6
0							
1							
2							
3		R	Y	Y			
4		Y	R	R	Y		
5	R	Y	R	R	Y	R	

Fig 7.4

After

	0	1	2	3	4	5	6
0							
1							
2							
3		R		Y	R		
4		Y	Y	R	Y		
5	R	Y	R	R	Y	R	

Fig 7.5

The programmer has to decide whether to continue to use a 2D array, or produce an array of queues.

Evaluate the use of a 2D array versus an array of queues to perform this action.

[9]

(ii) Explain why a stack would **not** be an appropriate data structure for this revised game.

[2]

END OF QUESTION PAPER

Question			Answer/Indicative content	Marks	Guidance
1		i	Size is fixed when structure created / size cannot change **during processing**	1	**Examiner's Comments** These four questions were marked as a group. These questions were good differentiators and allowed for a clear distinction between candidates. The more able got four marks the majority managed two marks.
		ii	array	1	
		iii	Size can change **during processing**	1	
		iv	Storage required is unknown initially / more difficult to program	1	**Examiner's Comments** These four questions were marked as a group. These questions were good differentiators and allowed for a clear distinction between candidates. The more able got four marks the majority managed two marks.
			Total	4	
2	a		• Identifier: Board / board • Number of dimensions: 2 • Most appropriate data type: Character / String	3	Allow suitable name Allow 2D Do not except alphanumeric **Examiner's Comments** Generally well answered, the most common error was with the dimensions where candidates gave (3, 15) or 45 as the answer.
	b		*In order* • 15 • Column • Row	3	cao **Examiner's Comments** This question was well answered with most candidates gaining full marks
			Total	6	

Question			Answer/Indicative content	Marks	Guidance

3			• String length calculated (1)
			• Correct number of characters from passed string taken ... (1)
			• ... in reverse order (1)
			• Characters placed in stack in correct order (1)
			• String length placed in stack at correct point (1)
			• Meaningful variable names used (1) (AO2.1)
			
			Example program
			
			`procedure passToStack(passString)`
			`   stringLen = passString.Length()`
			`   if stringLen == 0 then`
			`      stack[0]=0`
			`   else`
			`      stackPtr = 0`
			`      stringPtr = stringLen - 1`
			`      for i = 1 TO stringLen`
			`         stack[stackPtr] =`
			`passString[stringPtr]`
			`         stackPtr = stackPtr + 1`
			`         stringPtr = stringPtr -1`
			`      next i`
			`      stack[stackPtr] = stringLen`
			`   endif`
			`endprocedure`
				6	Allow `StackPtr` to be used instead of `i` in loop, as we would not expect them to know that some compilers do not always increment “loop counter” when they exit loops (i.e. loop counter on exit is undefined)
					
					Accept candidates using built-in stack methods e.g. `stack.push(word.substring(i,1))`
					
					Do not penalise for syntax errors if the logic can clearly be followed.
					
					Max 6 mark
					
					Examiner's Comments
					
					Many of the same comments regarding pseudocode as in 4b once again applied in 4d. An encouraging number of able candidates produced quite elegant solutions.

			Total	6	

Question			Answer/Indicative content	Marks	Guidance
4	a	i	"6+1" in the correct box. [1] Speech marks present [1] <table><tr><td>"2*3"</td><td>"1+4"</td><td>"3-1"</td><td>"10/2"</td><td>"3+6"</td><td>"6+1"</td><td></td><td></td></tr></table>	2 AO2.1 (2)	**Examiner's Comment:** Many candidates would have scored well on this question if they understood that a queue is FIFO. Those who did not understand the basic properties of a queue struggled with the question.
		ii	1 mark for head, 1 for tail `head` = 0 [1] `tail` = 5 [1]	2 AO2.1 (2)	**Examiner's Comment:** Many candidates would have scored well on this question if they understood that a queue is FIFO. Those who did not understand the basic properties of a queue struggled with the question.

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td></td><td>b</td><td></td><td>1 mark for pseudocode / code that meets each bullet

• Input a question [1]
• Check if <code>tail</code> is full and outputs message / reports error [1]
• Increment <code>tail</code> [1]
• Adds question to <code>tail</code> of <code>questions</code> [1]

e.g.<pre>
procedure add()
    maxElements = 10
    item = input("Enter a question")
    if tail == maxElements - 1 then
        print("Queue is full")
    else
        tail = tail + 1
        questions[tail]=item

    endif
endprocedure
</pre></td><td>4

AO3.2
(4)</td><td>Examiner's Comment:
Again, the use of pseudocode posed problems for many candidates. Those who had a wider programming experience were apparent from the well-crafted solutions. Those who gained credit generally gained two marks for understanding how the pointers were updated and how data was added / removed. Fewer scored full marks by also performing error checking.</td></tr>
<tr><td></td><td>c</td><td></td><td>1 mark per bullet to max 3
e.g.

• A queue is First In First Out (FIFO) [1]
• The questions are retrieved in the order they are stored [1]
• Questions can be added to the end [1]
• Dynamic structure… [1]
• …expands to take more questions [1]</td><td>3
AO1.2

(2)
AO2.1
(1)</td><td>Examiner's Comment:
Many candidates understood that a queue was a FIFO structure, but fewer could then go on to explain in context why this would then be a suitable data structure for the problem in context.</td></tr>
</table>

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td></td><td>d</td><td></td><td>1 mark for pseudocode/code that meets each bullet
Checking if queue is empty [1]
…outputting message/reporting error [1]
Outputting element in <code>questions</code> at index head [1]
Increment head [1]
e.g.
<pre>procedure remove()
    if head == tail + 1 then
        print("No questions")
    else
        print(questions[head])
        head = head + 1
    endif
endprocedure</pre></td><td>4

AO3.2
(4)</td><td>Examiner's Comment:
Again, the use of pseudocode posed problems for many candidates. Those who had a wider programming experience were apparent from the well-crafted solutions. Those who gained credit generally gained two marks for understanding how the pointers were updated and how data was added/removed. Fewer scored full marks by also performing error checking.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>15</td><td></td></tr>
</table>

Question			Answer/Indicative content	Marks	Guidance
5		i	**Mark Band 3 – High level** **(7–9 marks)** The candidate demonstrates a thorough knowledge and understanding of queues and arrays; the material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence / examples will be explicitly relevant to the explanation. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2 – Mid level** **(4–6 marks)** The candidate demonstrates reasonable knoledge and understanding of queues and arrays; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence / examples are for the most part implicitly relevant to the explanation. The candidate provides a reasonable discussion, the majority of which is focused. Evaluative comments are, for the most part appropriate, although one or two opportunities for development are missed. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.* **Mark Band 1 – Low Level** **(1–3 marks)** The candidate demonstrates a basic knowledge of queues and arrays with limited understanding shown; the material is basic and contains some inaccuracies. The candidates makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus.	9	**AO1: Knowledge and Understanding Indicative content** • Arrays are static (size cannot change) • Queues are dynamic (size can change) • Queues use pointers to identify the first element (to be removed) **AO2: Application** • Array will need all elements to be moved 'down 1' each time a disc is removed • Queue will allow the front element to be taken out and then the pointer will move • Algorithms for queues can be more complex, especially as the language may use an array to implement the queue **AO3: Evaluation** Candidates will need to evaluate the benefits and drawbacks of using queues and arrays and suggest an appropriate solution e.g. • Size does not need to change (Static is needed as grid is fixed size) so that benefit of queues is not necessary • Programmer has already written a program using arrays, may be less time consuming to edit it for arrays • Language may need a queue to be programmed in an array, therefore an array may be more straight forward to use • Queue does not need to move all elements each time a counter is removed, only pointers change

Question			Answer/Indicative content	Marks	Guidance
			Judgements if made are weak and unsubstantiated. *The information is basic and comunicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* 0 marks No attempt to answer the question or response is not worthy of credit.		
		ii	Max 2 • Stack is last-in-first-out (1) • This game the first-in needs to be first-out (1)	2	
			Total	11	