AS · Cambridge (CIE) · Computer Science

57 mins 8 questions

Exam Questions

Introduction to Abstract Data Types (ADT)

ADT overview

1 (a) A program uses a stack to hold up to 60 numeric values.
The stack is implemented using two integer variables and a 1D array.

The array is declared in pseudocode as shown:

DECLARE ThisStack : ARRAY[1:60] OF REAL

The stack operates as follows:

- Global variable SP acts as a stack pointer that points to the next available stack location. The value of SP represents an array index.
- Global variable OnStack represents the number of values currently on the stack.
- The stack grows upwards from array element index 1.

(i) Give the initial values that should be assigned to the **two** variables.

SP .....

OnStack .....

[1]

(ii) Explain why it is not necessary to initialise the array elements before the stack is used.

[2]

Answer

Mark Scheme and Guidance

(i)

SP: 1
OnStack: 0

One mark for both correct values

(ii)

MP1 Unused values cannot be popped / taken off the stack // initialised values would never be used / unused elements cannot be accessed

MP2 ... until a value has first been pushed / written // overwrites previous value

(3 marks)

(b) A function to add a value to ThisStack is expressed in pseudocode as shown. The function will return a value to indicate whether the operation was successful or not.

Complete the pseudocode by filling in the gaps.

```
FUNCTION Push(ThisValue : REAL) RETURNS BOOLEAN

DECLARE ReturnValue : BOOLEAN

IF ..... THEN

RETURN ..... // stack is already full

ENDIF

..... ← ThisValue

SP ← .....

OnStack ← OnStack + 1

RETURN TRUE

ENDFUNCTION
```

Answer

Mark Scheme and Guidance

Example solution:

```
FUNCTION Push(ThisValue : REAL) RETURNS BOOLEAN
DECLARE ReturnValue : BOOLEAN
```

```
IF OnStack = 60 / >59 // SP = 61 / SP > 60 // SP outside the range 1 to 60 THEN
RETURN FALSE // Stack is already full
ENDIF
ThisStack[SP] ← ThisValue
SP ← SP + 1
OnStack ← OnStack + 1
RETURN TRUE
ENDFUNCTION1
```

Mark as follows:
One mark per gap

(4 marks)

2 (a) The implementation of a linked list uses an integer variable and a 1D array List of type Node.

Record type Node is declared in pseudocode as follows:

```
TYPE Node
DECLARE Data : STRING
DECLARE Pointer : INTEGER
ENDTYPE
```

The array List is declared in pseudocode as follows:

```
DECLARE List : ARRAY[1:200] OF Node
```

The linked list will operate as follows:

- Integer variable HeadPointer will store the array index for the first node in the linked list.
- The Pointer field of a node contains the index value of the next array element (the next node) in the linked list.
- The value 0 is used as a null pointer.

State why the value 0 has been selected as the null pointer.

Answer

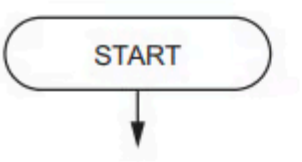
Mark Scheme and Guidance

0 is not a valid array / List index value // No 0^{th} element in the array

(1 mark)

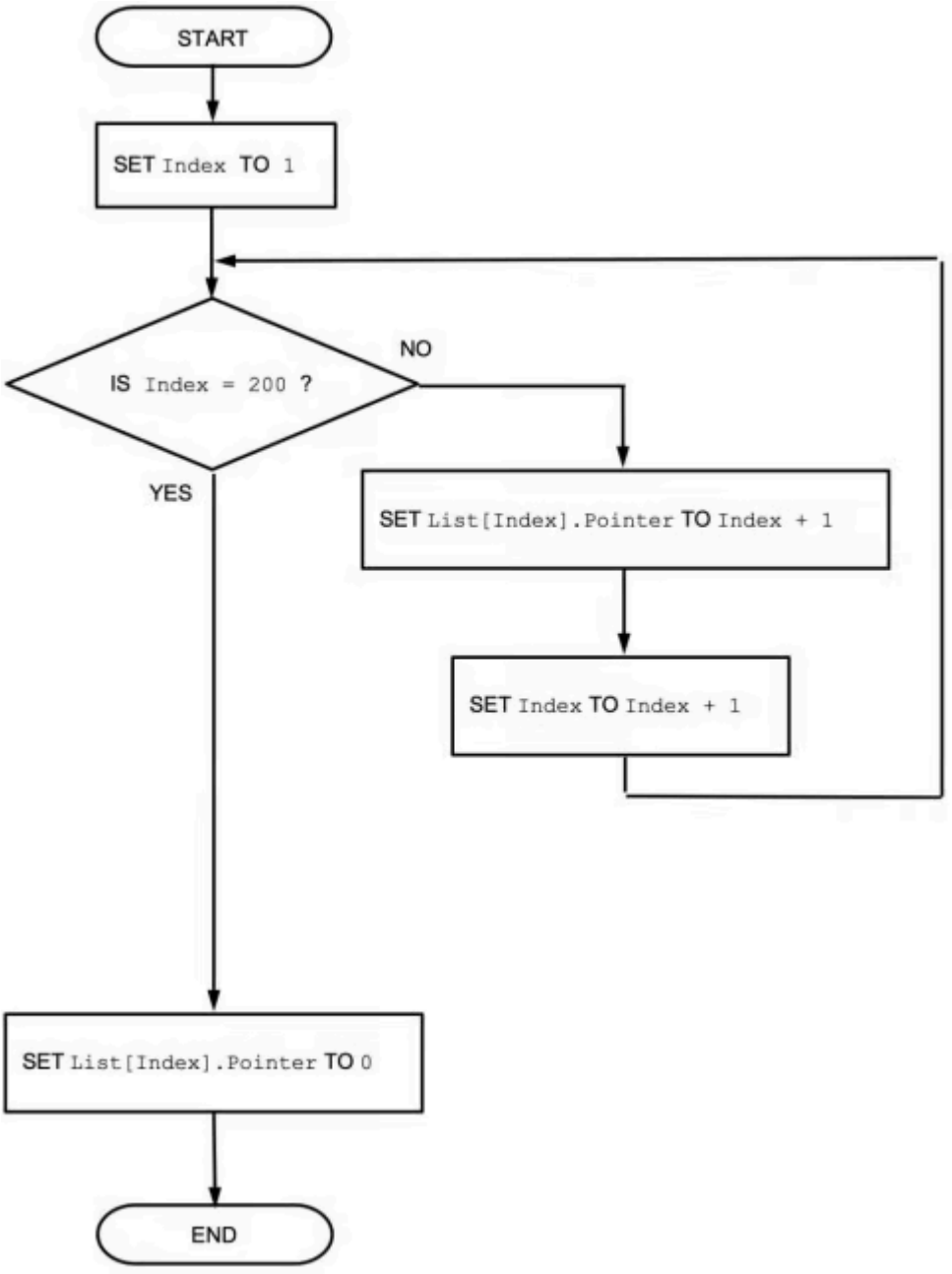
(b) The array List will be initialised so that each node points to the following node. The last node will contain a null pointer.

Complete the program flowchart to represent the algorithm for this operation.



Answer

Mark Scheme and Guidance



Mark as follows:
MP1 Use of variable as array index **and** initialisation to 1 / to first element of array
MP2 Loop for 199 / 200 iterations

MP3 Assign Index+1 to each pointer field in a loop
MP4 Assign 0 to 200th pointer field

(4 marks)

(c) An algorithm outputs the Data field from all nodes in the array List. The order the Data is output should be the same order it is stored in the linked list.

Describe the algorithm in **four** steps.

Do **not** use pseudocode statements in your answer.

Step 1 .....

Step 2 .....

Step 3 .....

Step 4 .....

Answer

Mark Scheme and Guidance

One mark per step:

MP1 Assign HeadPointer to ThisPointer / Current Pointer // Identify first node using headpointer

MP2 Loop the following until ThisPointer value is 0 (zero) / null pointer reached

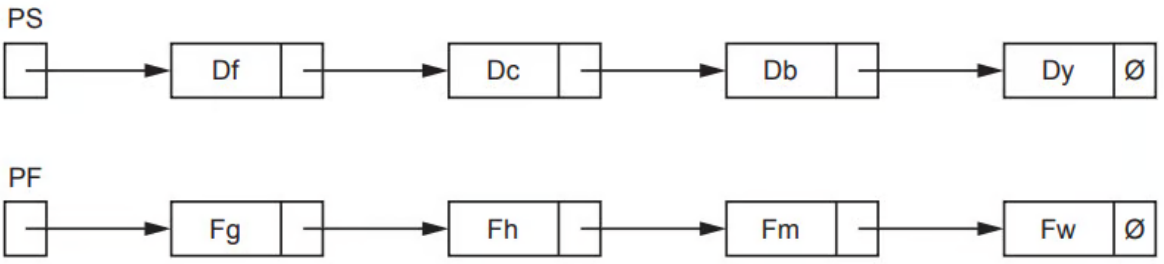
MP3 ... Output data (field) from array element at index ThisPointer // Output data (field) of the current node / array element

MP4 ... Assign pointer field from current array element / index / to ThisPointer / Current Pointer // Assign pointer field from current node to ThisPointer / Current Pointer

(4 marks)

3 The diagram shows an Abstract Data Type (ADT) representation of a linked list after data items have been added.

- PS is the start pointer.
- PF is the free list pointer.
- Labels Df, Dc, Db and Dy represent the data items of nodes in the list.
- Labels Fg, Fh, Fm and Fw represent the data items of nodes in the free list.
- The symbol Ø represents a null pointer.



Describe the linked list immediately after initialisation, before **any** data items are added.

Answer

Mark Scheme and Guidance

One mark per point:

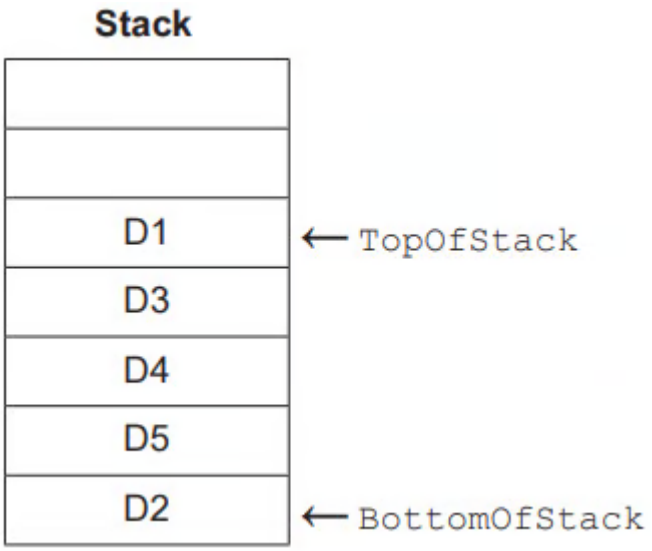
1. The PS contains a null pointer
2. The PF points to the first element on the free list
3. All the nodes are on the free list

(3 marks)

4 (a) The diagram represents an Abstract Data Type (ADT).

The operation of this stack may be summarised as follows:

- The TopOfStack pointer points to the last item added to the stack.
- The BottomOfStack pointer points to the first item on the stack.

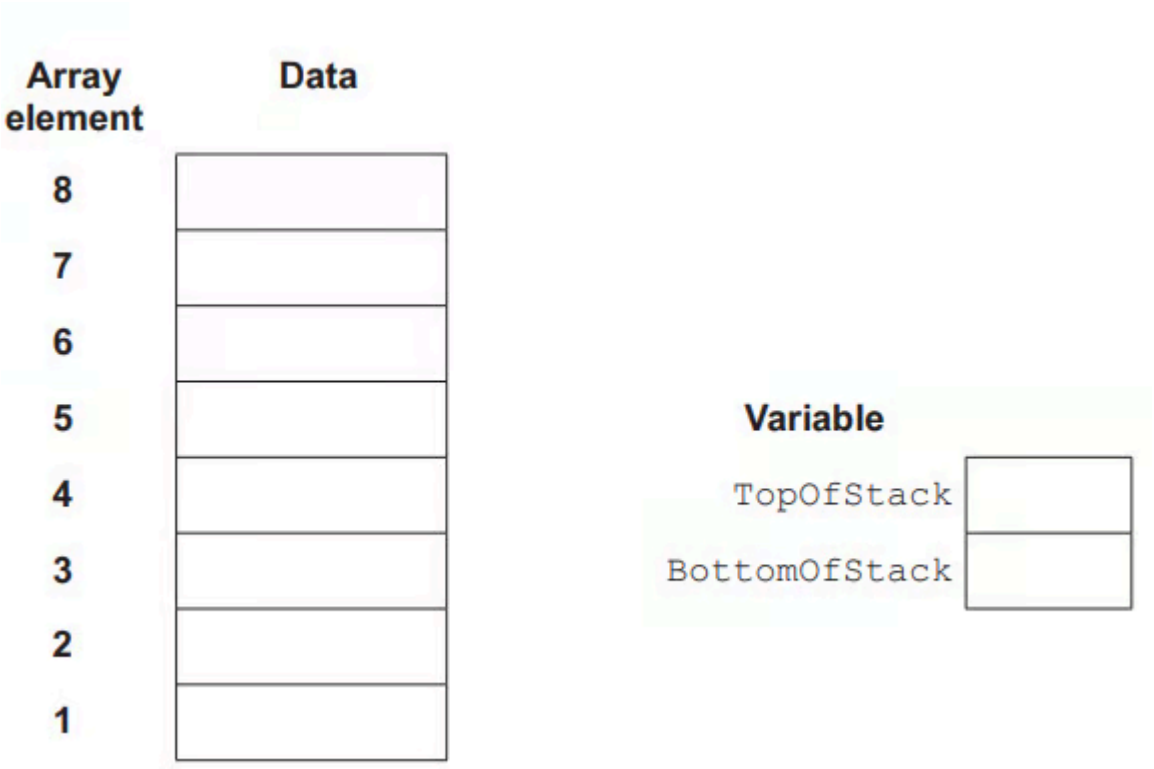


The stack is implemented using two variables and a 1D array of 8 elements as shown.

The variables are used to reference individual elements of the array, in such a way that:

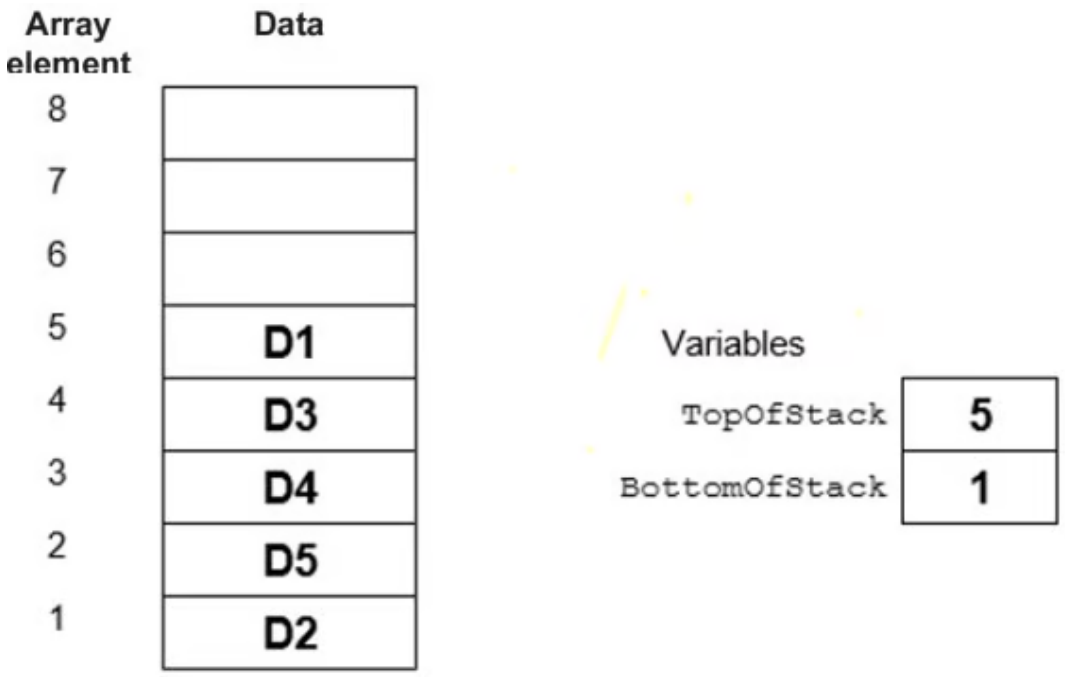
- the array is filled from the lowest indexed element towards the highest
- all the elements of the array are available for the stack.

Complete the diagram to represent the state of the stack as shown above.



Answer

Mark Scheme and Guidance



MP1 all values in the order and location shown
MP2 TopOfStack value is index of element containing D1
MP3 BottomOfStack value is index of element containing D2

(3 marks)

(b) A function Push() will add a value onto the stack by manipulating the array and variables in **part (a)**.

Before adding a value onto the stack, the algorithm will check that space is available.

If the value is added to the stack, the function will return TRUE, otherwise it will return FALSE.

The algorithm is expressed in five steps.

Complete the steps.

1. If ..... then return FALSE

2. Otherwise ..... TopOfStack

3. Use TopOfStack as an ..... to the array.

4. Set the element at this ..... to the ..... being added.

5. Return ..... .

Answer

Mark Scheme and Guidance

MP1 If **TopOfStack = 8 // (stack) full** then return FALSE

MP2 Otherwise, **increment** TopOfStack

MP3 Use TopOfStack as an **index** to the Array

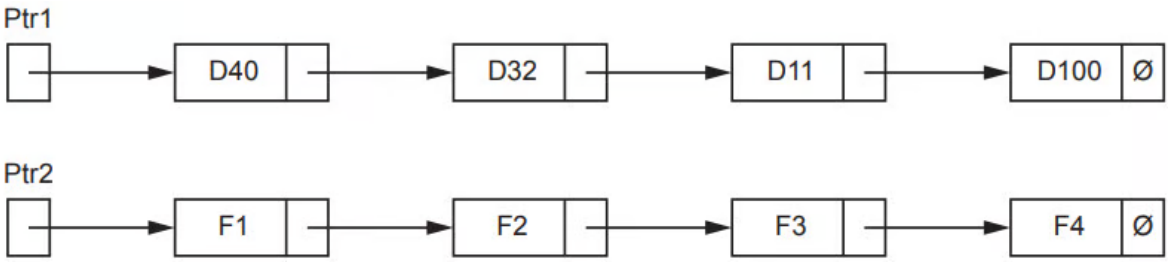
MP4 Set the element at this **index / location / position** to the **value / data / item** being added

MP5 Return **TRUE**

(5 marks)

5 (a) The diagram represents a linked list Abstract Data Type (ADT).

- Ptr1 is the start pointer. Ptr2 is the free list pointer.
- Labels D40, D32, D11 and D100 represent the data items of nodes in the list.
- Labels F1, F2, F3 and F4 represent the data items of nodes in the free list.
- The symbol Ø represents a null pointer.



The linked list is implemented using two variables and two 1D arrays as shown.

The pointer variables and the elements of the Pointer array store the indices (index numbers) of elements in the Data array.

Complete the diagram to show how the linked list as shown above may be represented using the variables and arrays.

Variable	Value
Start_Pointer	
Free_List_Pointer	5

Index	Data array	Pointer array
1	D32	2
2		3
3		
4	D40	
5		
6	F2	7
7		
8		

Answer

Mark Scheme and Guidance

Variable	Value
Start_Pointer	4
Free_List_Pointer	5

Index	Data Array	Pointer Array
1	D32	2
2	**D11**	3
3	**D100**	**0**
4	D40	**1**
5	**F1**	**6**
6	F2	7
7	**F3**	**8**
8	**F4**	**0**

Mark as follows:

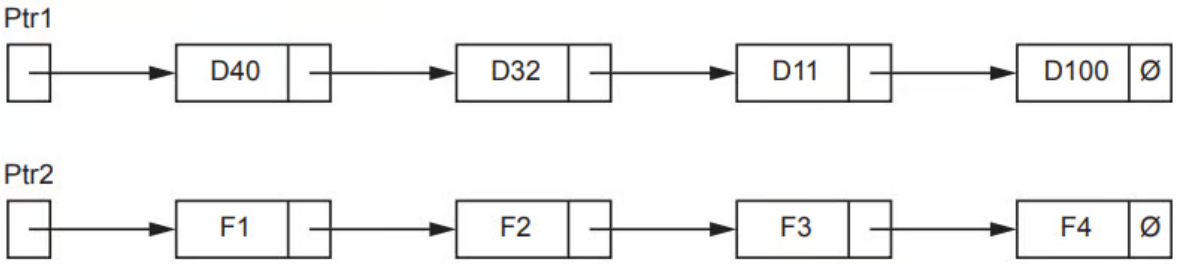
- One mark for Start_Pointer value
- One mark each group of row(s):
- 2
- 3 and 4
- 5
- 7 and 8

For null pointer: accept 0 / ∅ / an out-of-bound index value less than 1, greater than 8

(5 marks)

(b) The original linked list is to be modified. A new node D6 is inserted between nodes D32 and

D11.



The algorithm required is expressed in four steps as shown.

Complete the steps.

1. Assign the data item ..... to ..... .

2. Set the ..... of this node to point to ..... .

3. Set Ptr2 to point to ..... .

4. Set pointer of ..... to point to ..... .

Answer

Mark Scheme and Guidance

One mark per step:

1. Assign the data item **D6** to **F1**
2. Set the **pointer** of this node to point to **D11**
3. Set Ptr2 to point to **F2**
4. Set pointer of **D32** to point to **D6**

(4 marks)

6 The program contains a module GetFile() which receives text files sent from another computer.

Lines from the file are sent one at a time. Each message contains one line and ProcessMsg() from part **(b)** adds each message as it is received onto stack 1.

Module GetFile() removes messages from stack 1 and writes the data to a text file.

There is a problem. Under certain circumstances, the received file does not appear as expected.

Assume that while a file is being received ProcessMsg() receives only messages containing lines from the file.

(i) Describe the circumstances and explain the problem.

Circumstances .....

Explanation .....

[3]

(ii) Suggest a more appropriate Abstract Data Type that could be used to store the messages that would not have the same problem.

[1]

Answer

Mark Scheme and Guidance

(i)

One mark per point **Max 3** marks:

Decide on scenario and mark accordingly.
Scenario one:

- If more than one line is / all lines are stored on the stack (before line(s) are removed)

- The stack operates as a FILO device // Last item added to stack will be in first item out
- So **lines in the file** appear out of sequence

Scenario two:

- Stack is Full
- Not all lines can be stored on the stack
- so resulting **file** will not contain **all the original lines**

Scenario three:

- (All) the data in a line read can't be stored on the stack
- Stack elements have not been allocated enough memory
- so only **part of each line** is stored in the **file**

Scenario four:

- Stack is empty
- The stack is being read faster than it is being written to
- so **blank lines** may be inserted into the **file**

(ii)

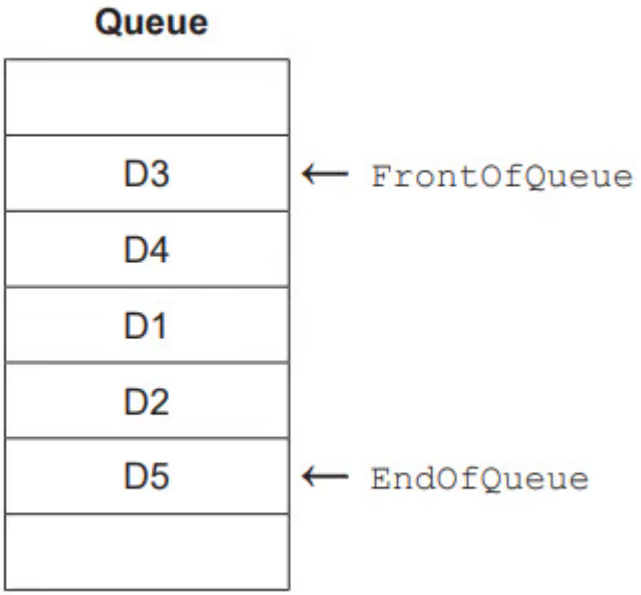
Queue

(3 marks)

7 (a) The diagram represents a queue Abstract Data Type (ADT).

The organisation of this queue may be summarised as follows:

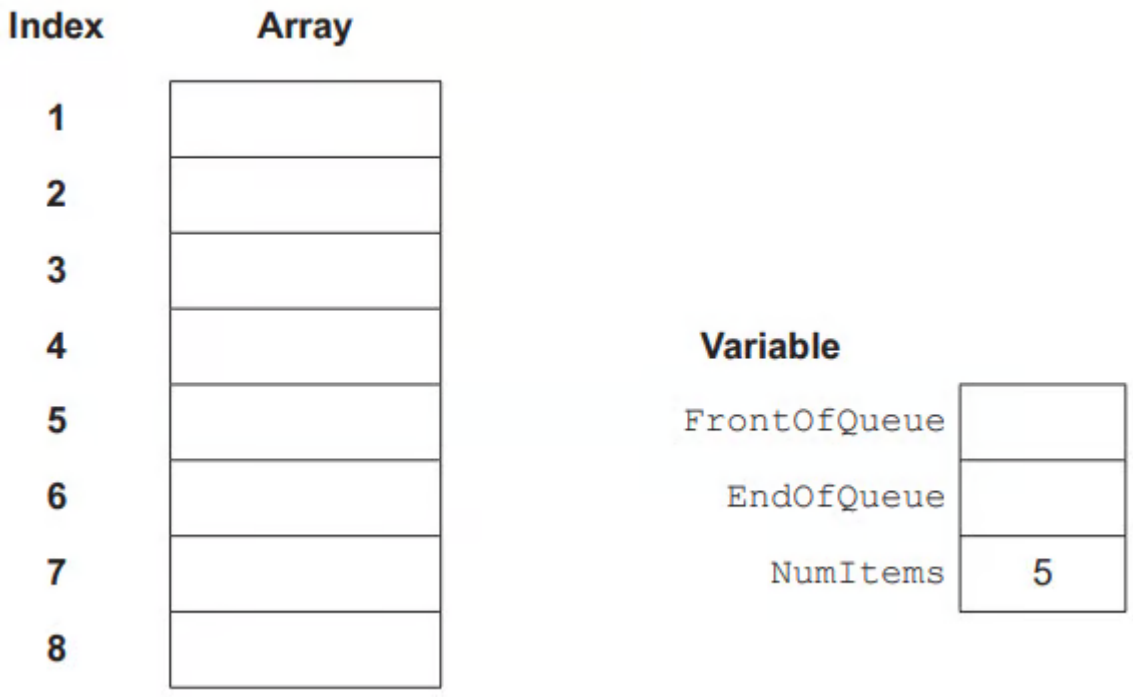
- The FrontOfQueue pointer points to the next data item to be removed.
- The EndOfQueue pointer points to the last data item added.



The queue is implemented using three variables and a 1D array of eight elements as shown. The variable NumItems stores the number of items in the queue.

The pointer variables store indices (index numbers) of the array.

Complete the diagram to represent the state of the queue as shown above.



Answer

Mark Scheme and Guidance

Index	Array
1	
2	D3
3	D4
4	D1
5	D2
6	D5
7	
8	

Variable	
FrontOfQueue	2
EndOfQueue	6
NumItems	5

MP1 D3, D4, D1, D2 and D5 in question order or reversed - in any five consecutive locations
MP2 FoQ value matches index with D3
MP3 EoQ value matches index with D5

(3 marks)

(b) A module AddTo() will add a value to the queue by manipulating the array and variables in part **(a)**.

The queue implementation is circular. When pointers reach the end of the queue, they will 'wrap around' to the beginning.

Before a value can be added to the queue, it is necessary to check the queue is not full.

The algorithm to add a value to the queue is expressed in six steps.

Complete the steps.

1. If NumItems ..... then jump to step 6.

2. Increment ..... .

3. If ..... then set EndOfQueue to ..... .

4. Increment ..... .

5. Set the ..... at the index stored in ..... to the ..... being added.

6. Stop.

Answer

Mark Scheme and Guidance

MP1 If NumItems **is / = 8 // (queue) full** then jump to step 6
MP2 Increment **EndOfQueue**
MP3 If **EndOfQueue = 9** then set EndOfQueue to **1**
MP4 Increment **NumItems**

MP5 Set the **Element** at the index ...
MP6 stored in **EndOfQueue** to **value/data/item** being added

Mark as follows:
Steps 1 to 4: One mark for gaps filled as shown

Step 5: One mark for 'element' and one mark for other two terms

(6 marks)

8 (a) A stack Abstract Data Type (ADT) is to be implemented using pseudocode, with procedures to initialise it and to push new items onto the stack.

A 1D array Stack stores the contents of the stack.

(i) Study the pseudocode in **part (a)(ii)** and complete the table of identifiers by writing the missing data types and descriptions.

Identifier	Data type	Description
BasePointer		
TopPointer		
Stack	REAL	

[2]

(ii) Complete the pseudocode.

```
CONSTANT MaxSize = 40
DECLARE BasePointer : INTEGER
DECLARE TopPointer : INTEGER
DECLARE Stack : ARRAY[1:40] OF REAL

// initialisation of stack
PROCEDURE Initialise()

..... ← 1 .

..... ← 0
ENDPROCEDURE

// push an item onto the stack
PROCEDURE Push(NewItem : REAL)

.....MaxSize THEN

.....
```

```
Stack[TopPointer] ← .....
ENDIF
ENDPROCEDURE
```

[5]

Answer

Mark Scheme and Guidance

(i)

Two marks for all five empty boxes correct
One mark for any three or four empty boxes correct

Identifier	Data type	Description
BasePointer	INTEGER	Points to the bottom of the stack
TopPointer	INTEGER	Points to the top of the stack
Stack	REAL	List of decimal numbers stored in the stack

(ii)

One mark for each correctly completed line **(Max 5)**

```
CONSTANT MaxSize = 40
DECLARE BasePointer : INTEGER
DECLARE TopPointer : INTEGER
DECLARE Stack : ARRAY[1:40] OF REAL

// initialisation of stack
PROCEDURE Initialise()
BasePointer ← 1
TopPointer ← 0
ENDPROCEDURE

// adding an item to the stack
PROCEDURE Push(NewItem)
```

```
IF TopPointer < MaxSize THEN
TopPointer ← TopPointer + 1
Stack[TopPointer] ← NewItem
ENDIF
ENDPROCEDURE
```

(7 marks)

(b) Justify the use of a linked list instead of an array to implement a stack.

Answer

Mark Scheme and Guidance

One mark for linked list and one mark for array **(Max 2)**

Linked list
MP1 A linked list is a dynamic data structure / not restricted in size
MP2 Has greater freedom to expand or contract by adding or removing nodes as necessary
MP3 Allows more efficient editing using pointers (instead of moving the data).

Array
MP4 An array is a static data structure1 generally fixed in size
MP5 When the array is full, the stack cannot be extended any further.

(2 marks)