PAPER 2 ADT

COMPUTER SCIENCE

Topic: Paper 2 ADT

INSTRUCTIONS

- Carry out every instruction in each task.
- Answer **all** questions.
- Use a black or dark blue pen.
- You may use an HB pencil for any diagram, graphs or rough working.
- **Calculator Not Allowed**.
- Show your workings if relevant.

INFORMATION

- The number of marks for each question or part question is shown in brackets [ ].

bishaltwr@gmail.com

3 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.

(a) (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]

(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
```

[4]

3 A program processes data using a stack. The data is copied to a text file before the program ends.

(a) The following diagram shows the current state of the stack.

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 grows upwards when items are added.

Stack		Pointer
Memory location	**Value**	
506		
505	WWW	← `TopOfStack`
504	YYY	
503	XXX	
502	ZZZ	
501	NNN	
500	PPP	← `BottomOfStack`

(i) An error will be generated if an attempt is made to POP a value when the stack is empty.

State the maximum number of consecutive POP operations that could be performed on the stack shown above **before** an error is generated.

..... [1]

(ii) The following operations are performed:

1. POP and store value in variable `Data1`
2. POP and store value in variable `Data2`
3. PUSH value AAA
4. PUSH value BBB
5. POP and discard value
6. POP and store value in variable `Data2`

Complete the diagram to show the state of the stack and the variables **after** the given operations have been performed.

Stack

Memory location	Value
506	
505	
504	
503	
502	
501	
500	

Pointer

Variable	Value
`Data1`	
`Data2`	

[4]

(b) The data is copied to a text file before the program ends.

(i) State an advantage of writing the data from the stack to a text file before the program ends.

.....

..... [1]

(ii) A module `SaveStack()` will write the data from the stack to a text file.

Express an algorithm for `SaveStack()` as five steps that could be used to produce pseudocode.

Write the **five** steps.

Step 1 .....

.....

Step 2 .....

.....

Step 3 .....

.....

Step 4 .....

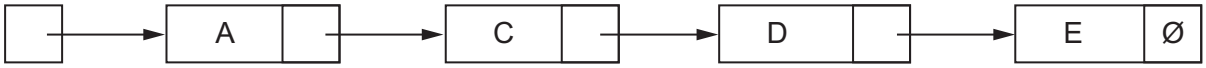
.....

Step 5 .....

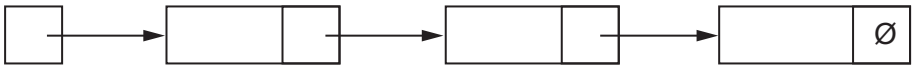
.....

[5]

6 The following diagram represents an Abstract Data Type (ADT) for a linked list.



The free list is as follows:



(a) Explain how a node containing data value B is added to the list in alphabetic sequence.

.....

.....

.....

.....

.....

.....

..... [4]

(b) Describe how the linked list in **part (a)** may be implemented using variables and arrays.

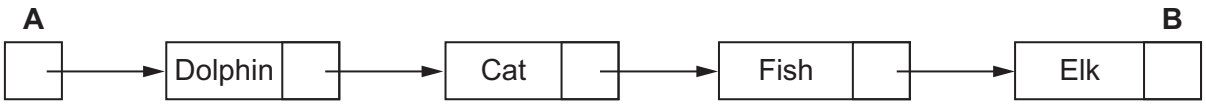
.....

.....

.....

..... [2]

3 The following diagram represents an Abstract Data Type (ADT).



(a) Identify this type of ADT.

..... [1]

(b) Give the technical term for the item labelled **A** in the diagram.

..... [1]

(c) Give the technical term for the item labelled **B** in the diagram.

Explain the meaning of the value given to this item.

Term .....

Meaning .....

.....

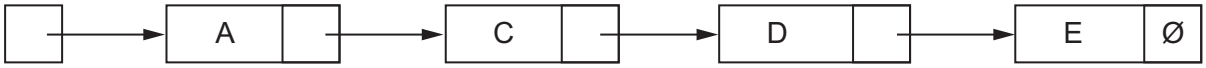
.....

[2]

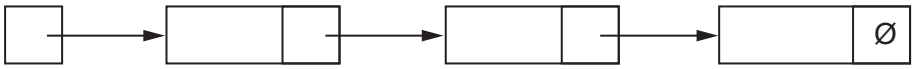
(d) Complete the diagram to show the ADT after the data has been sorted in alphabetical order.

[2]

6 The following diagram represents an Abstract Data Type (ADT) for a linked list.



The free list is as follows:



(a) Explain how a node containing data value B is added to the list in alphabetic sequence.

.....

.....

.....

.....

.....

.....

..... [4]

(b) Describe how the linked list in **part (a)** may be implemented using variables and arrays.

.....

.....

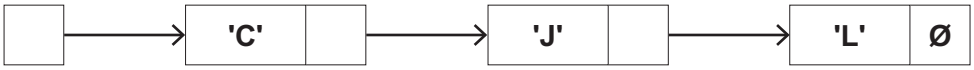
.....

..... [2]

4 **(a)** The following diagram shows an Abstract Data Type (ADT) representation of an ordered linked list. The data item stored in each node is a single character. The data will be accessed in alphabetical order.

The symbol Ø represents a null pointer.

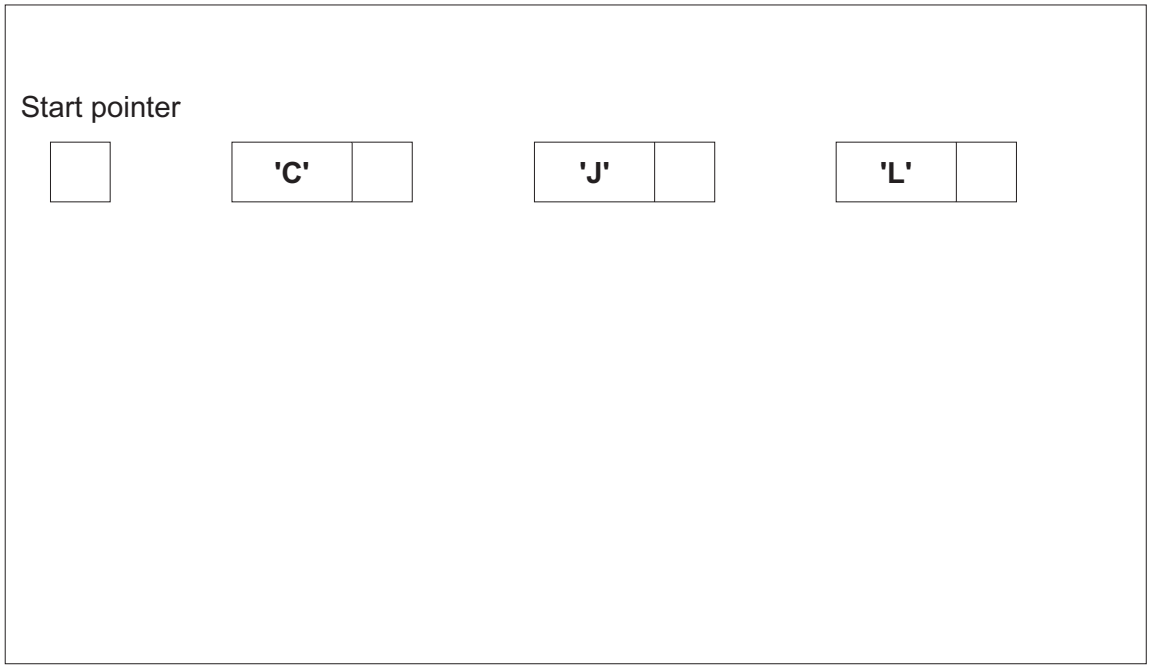
Start pointer



(i) Nodes with data 'A' and 'K' are added to the linked list. Nodes with data 'J' and 'L' are deleted.

After the changes, the data items still need to be accessed in alphabetical order.

Complete the diagram to show the new state of the linked list.



[4]

(ii) The original data could have been stored in a 1D array in which each element stores a character.

For example:

'C'	'J'	'L'		

Explain the advantages of making the changes described in **part (a)(i)** when the data is stored in the linked list instead of an array.

.....

.....

.....

..... [2]

(iii) Explain the disadvantages of making the changes described in **part (a)(i)** when the data is stored in the linked list instead of an array.

.....

.....

.....

..... [2]

(b) A program will store data using a linked list like the one shown in **part (a)**.

Explain how the linked list can be implemented.

.....

.....

.....

.....

.....

.....

.....

..... [4]

3 A program stores data in a text file. When data is read from the file, it is placed in a queue.

(a) The diagram below represents an Abstract Data Type (ADT) implementation of the queue. Each data item is stored in a separate location in the data structure. During initial design, the queue is limited to holding a maximum of 10 data items.

The operation of this queue may be summarised as follows:

- The Front of Queue Pointer points to the next data item to be removed.
- The End of Queue Pointer points to the last data item added.
- The queue is circular so that locations can be reused.

0		
1		
2		
3		
4		
5	Red	← Front of Queue Pointer
6	Green	
7	Blue	
8	Pink	← End of Queue Pointer
9		

(i) Describe how the data items Orange and Yellow are added to the queue shown in the diagram.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(ii) The following diagram shows the state of the queue after several operations have been performed. All queue locations have been used at least once.

0	D4	
1	D3	← End of Queue Pointer
2	D27	
3	D8	
4	D33	
5	D17	← Front of Queue Pointer
6	D2	
7	D1	
8	D45	
9	D60	

State the number of data items in the queue.

..... [1]

(b) The design of the queue is completed and the number of locations is increased.

A function `AddToQueue()` has been written. It takes a string as a parameter and adds this to the queue. The function will return `TRUE` if the string was added successfully.

A procedure `FileToQueue()` will add each line from the file to the queue. This procedure will end when all lines have been added or when the queue is full.

Describe the algorithm for procedure `FileToQueue()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

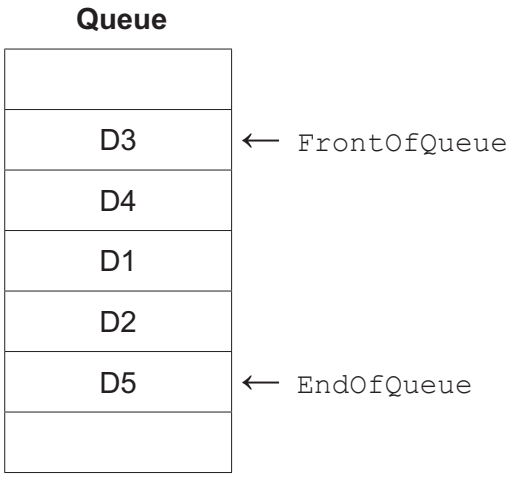
.....

..... [5]

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

The organisation of this queue may be summarised as follows:

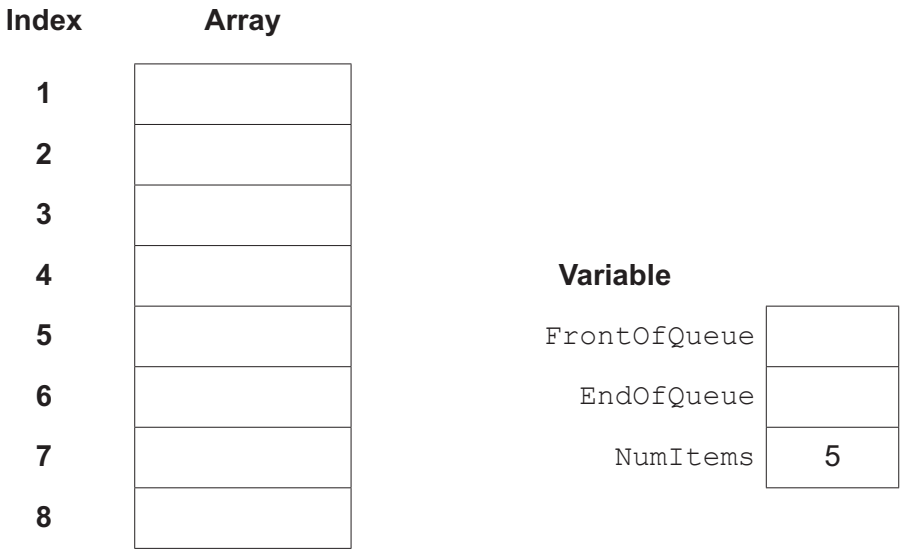
- The `FrontOfQueue` pointer points to the next data item to be removed.
- The `EndOfQueue` pointer points to the last data item added.



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.

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



[3]

(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.

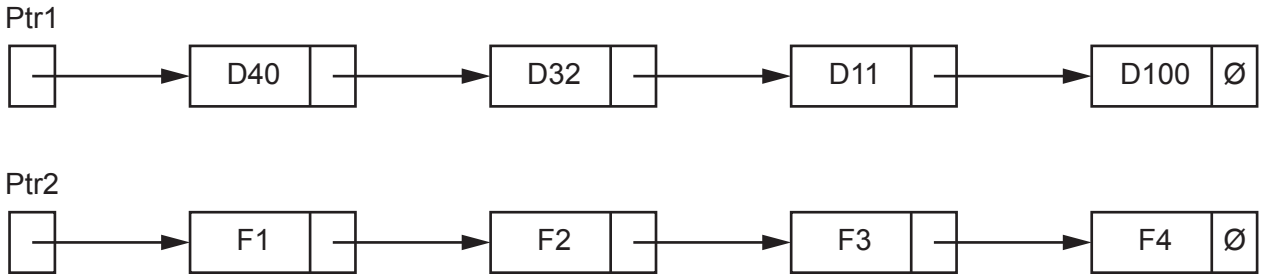
Complete the steps.

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.

[6]

3 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.



(a) 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.

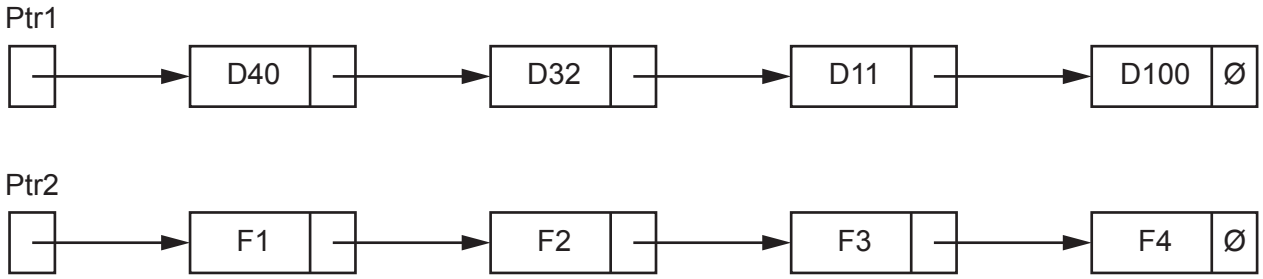
Complete the diagram to show how the linked list as shown above may be represented using the variables and arrays.

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		

[5]

(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.

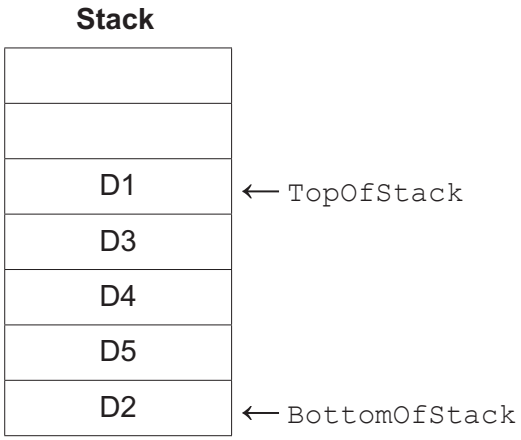
1. Assign the data item ..... to ..... .
2. Set the ..... of this node to point to ..... .
3. Set Ptr2 to point to ..... .
4. Set pointer of ..... to point to ..... .

[4]

3 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.

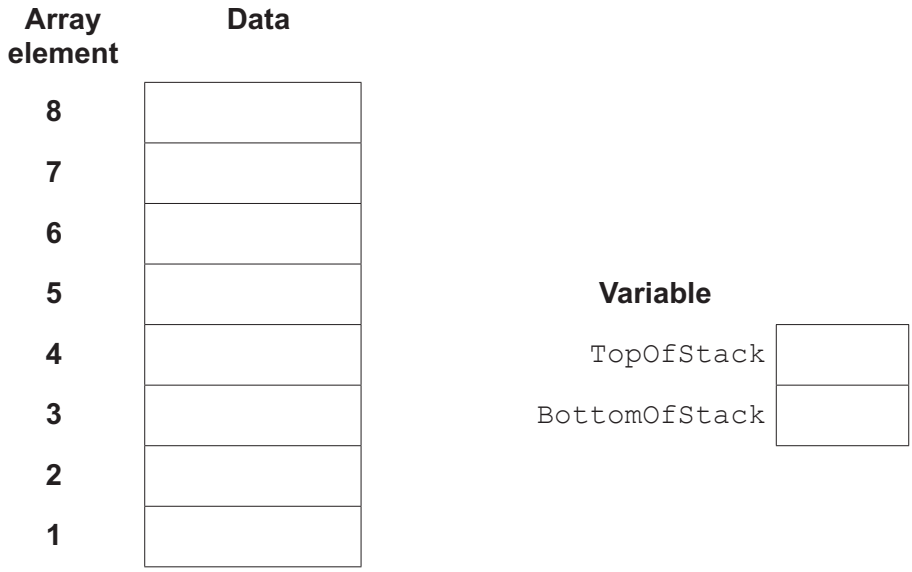


(a) The stack is implemented using two variables and a 1D array of 8 elements as shown.

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.



[3]

(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.

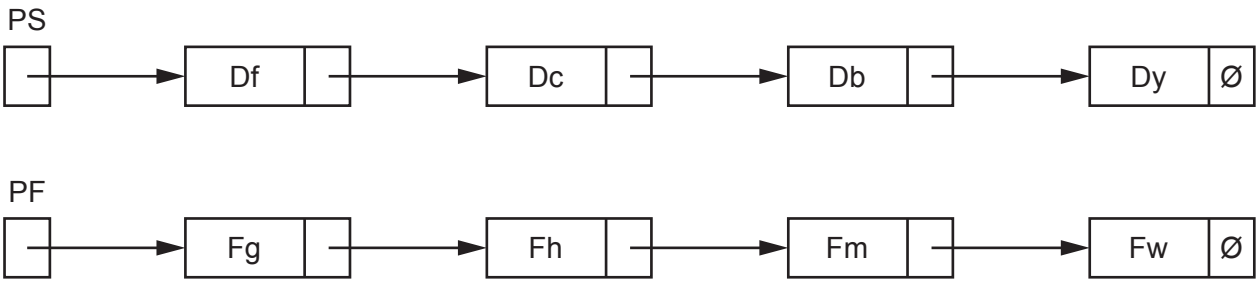
Complete the steps.

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 ..... .

[5]

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.



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

.....

.....

.....

.....

.....

..... [3]

(b) A program will be written to include a linked list to store alphanumeric user IDs.

The design uses two variables and two 1D arrays to implement the linked list.
Each array element contains data of a single data type and **not** a record.

The statements below describe the design.

Complete the statements.

The two variables will be of type ..... .

The two variables will be used as ..... to the arrays.

The values stored in the two variables will indicate .....

..... .

The first 1D array will be of type ..... .

The first 1D array will be used to ..... .

The second 1D array will be of type ..... .

The second 1D array will be used to ..... .

[5]

2 An algorithm will:

1. prompt and input a sequence of 100 integer values, one at a time
2. sum the positive integers
3. output the result of the sum.

(a) Write pseudocode for the algorithm.

Assume the value zero is neither positive nor negative.

You must declare all variables used in the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) The algorithm requires the use of basic constructs. One of these is sequence.

Identify **one other** basic construct required by the algorithm **and** describe how it is used.

Construct .....

Use .....

.....

[2]

3 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.

(a) (i) State why the value 0 has been selected as the null pointer.

.....

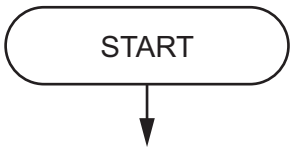
..... [1]

(ii) Give the range of valid values that could be assigned to variable `HeadPointer`.

..... [1]

(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.



[4]