NAME : DATE:

June 2026 PAPER 33

10 (a) A program declares a 1D array globally as `PartName[1:1000000] OF STRING`, and sorts its contents in ascending alphabetical order. The function `PartSearch` performs a binary search on the array. The item to be found is passed as a parameter. If it is found, its location in the array is returned, otherwise, –1 is returned.

(i). Complete the following table to define the variables used in the function.

Identifier	Data type	Description
Left		
Right		
Mid		
Target		

(ii) Complete the pseudocode for the function to perform a binary search on `PartName` using all the variables from the table in **10(a)(i)**.

Some lines may be left blank.

```
FUNCTION PartSearch(          :          ) RETURN
    Left ←
    Right ← 1000000
    WHILE (          <=          )
        ← (Left + Right) DIV 2
        IF PartName[Mid] =          THEN
            RETURN
        ELSE
            IF PartName[          ] = Target THEN
                Left ←          + 1
            ELSE
                ← Mid –
            ENDIF
        
    
    RETURN -1
ENDFUNCTION
```

(b) Describe how the time efficiency of a binary search changes as the number of items in the search list increases. Refer to Big O notation in your answer.

JUNE 2025 / PAPER 33

12 (a) An array is an Abstract Data Type (ADT).

Identify **two** other ADTs.

1 .....

2 .....

[1]

(b) A 1D array `DataArray` holds up to 1000 elements of type integer and needs to be sorted in ascending order.

Write the **pseudocode** for an insertion sort to sort the array into ascending order.

Use the identifiers from the table in your algorithm.

You do **not** need to declare any arrays or variables for this algorithm. You may assume this has already been done.

Identifier	Data type	Description
`Index`	`INTEGER`	counter for outer loop
`Position`	`INTEGER`	counter for inner loop – insertion position
`DataArray`	`INTEGER`	1D array to store up to 1000 integers
`Value`	`INTEGER`	value to insert

```
FOR Index ← [ ] TO [ ]
    Value ← [ ] [ ]
    Position ← Index - [ ]
    IF DataArray[ [ ] ] > Value THEN
        WHILE [ ] >= 1 AND DataArray[ [ ] > [ ]
            DataArray[ Position + 1 ] ← [ ] [ ]
            Position ← Position [ ] 1
        ENDWHILE
        DataArray [ [ ] + 1 ] ← [ ]
NEXT [ ]
```

(c) Describe **two** ways in which the performance of a sort routine is affected by the data to be sorted.

JUNE 2023 PAPER 33

11 Pseudocode is to be written to implement a queue Abstract Data Type (ADT) with items of the string data type. This will be implemented using the information in the table.

Identifier	Data type	Description
FrontPointer	INTEGER	points to the start of the queue
RearPointer	INTEGER	points to the end of the queue
Length	INTEGER	the current size of the queue
Queue	STRING	1D array to implement the queue

A constant, with identifier `MaxSize`, limits the size of the queue to 60 items.

(a) Write the pseudocode to declare `MaxSize`, `FrontPointer`, `RearPointer`, `Length` and `Queue`.

...

...

...

...

...

[3]

(b) Complete the following pseudocode for the function `Dequeue` to remove the front item from the queue.

```
FUNCTION Dequeue RETURNS STRING
   DECLARE Item : STRING
   ..                                                .. > 0 THEN
      Item ←                                                .
                                                            .
      IF Length = 0 THEN
         CALL Initialise // reset the pointers
      ELSE
         IF FrontPointer > MaxSize THEN
                                                        . ← 1
         ENDIF
      ENDIF
   ELSE
      OUTPUT "The print queue was empty - error!"
      Item ← ""
   ENDIF
   RETURN Item
ENDFUNCTION
```

[4]

(c) Explain how a new element can be added to the queue if it is implemented using two stacks.

12 (a) Describe what is meant by recursion.

(b) A Fibonacci sequence is a series of numbers formed by adding together the two preceding numbers, for example:

0, 1, 1, 2, ...

This function calculates and returns values in the Fibonacci sequence and uses recursion.

```
FUNCTION Fib(Number : INTEGER) RETURNS INTEGER
   IF Number <= 1 THEN
      Result ← Number
   ELSE
      Result ← Fib(Number - 1) + Fib(Number - 2)
   ENDIF
   RETURN Result
ENDFUNCTION
```

Complete the trace table for the function when it is called as `Fib(5)`.

[5]

Call number	Function call	Number	Result	Return value