2 Numbers are stored in a computer system using binary floating-point representation with:

- 10 bits for the mantissa
- 6 bits for the exponent
- two's complement form for both the mantissa and the exponent.

(a) Calculate the denary value of the given normalised binary floating-point number.
Show your working.

Mantissa										Exponent					
0	1	1	1	1	0	0	1	0	1	0	0	1	0	1	1

Working .....

.....

.....

.....

.....

Denary value .....

[3]

(b) Calculate the normalised binary floating-point representation of +26.6875 in this system.
Show your working.

Mantissa										Exponent					

Working .....

.....

.....

.....

.....

.....

[3]

7 A program is being developed to implement a customer loyalty scheme for a coffee shop.

Each customer has a unique customer ID starting at `10001` with this value increasing by one each time a new customer joins the loyalty scheme.

For example, the third customer who joins the loyalty scheme is given the customer ID `10003`

The loyalty scheme is limited to 1000 customers.

A customer is awarded a loyalty point every time they buy a coffee.

The programmer has decided to use a global 2D array `Loyalty` of type `INTEGER`. The array `Loyalty` is made up of 1000 rows and 2 columns. Each row relates to one customer; column 1 contains the unique customer ID and column 2 contains the number of customer loyalty points.

Rows in the array `Loyalty` that are not currently being used have the value of Column 1 set to `99999`

The array is sorted in ascending order by customer ID.

The programmer has defined a program module:

Module	Description
`FindCustomer()`	• called with parameter of type `INTEGER` representing a customer ID • searches the `Loyalty` array for this customer ID • the search will stop as soon as the customer ID is found • the search should efficiently deal with the situation when the customer ID is **not** stored in the `Loyalty` array • if the customer ID is found, return an integer value representing the loyalty points, otherwise return `-1`

(a) Write efficient pseudocode for module `FindCustomer()`

[8]

(b) A customer can claim a free coffee for every 11 loyalty points.

The programmer has defined a second program module:

Module	Description
`PointsReport()`	• output the customer ID for each customer who has 11 or more loyalty points • output the average loyalty points for all customers in the `Loyalty` array along with an appropriate message

Write efficient pseudocode for module `PointsReport()`

Assume the array contains the data for at least **one** customer.

.....

.....

.....

.....

.....

.....

.....

..... [7]

(c) The programmer decides to amend the customer ID; it will be stored as a `STRING` instead of an `INTEGER`. This means that the 2D array `Loyalty` can no longer be used.

(i) Explain why the 2D array `Loyalty` can no longer be used.

.....

..... [1]

(ii) Explain how a 1D array could be used to store both the loyalty points and the amended customer ID.

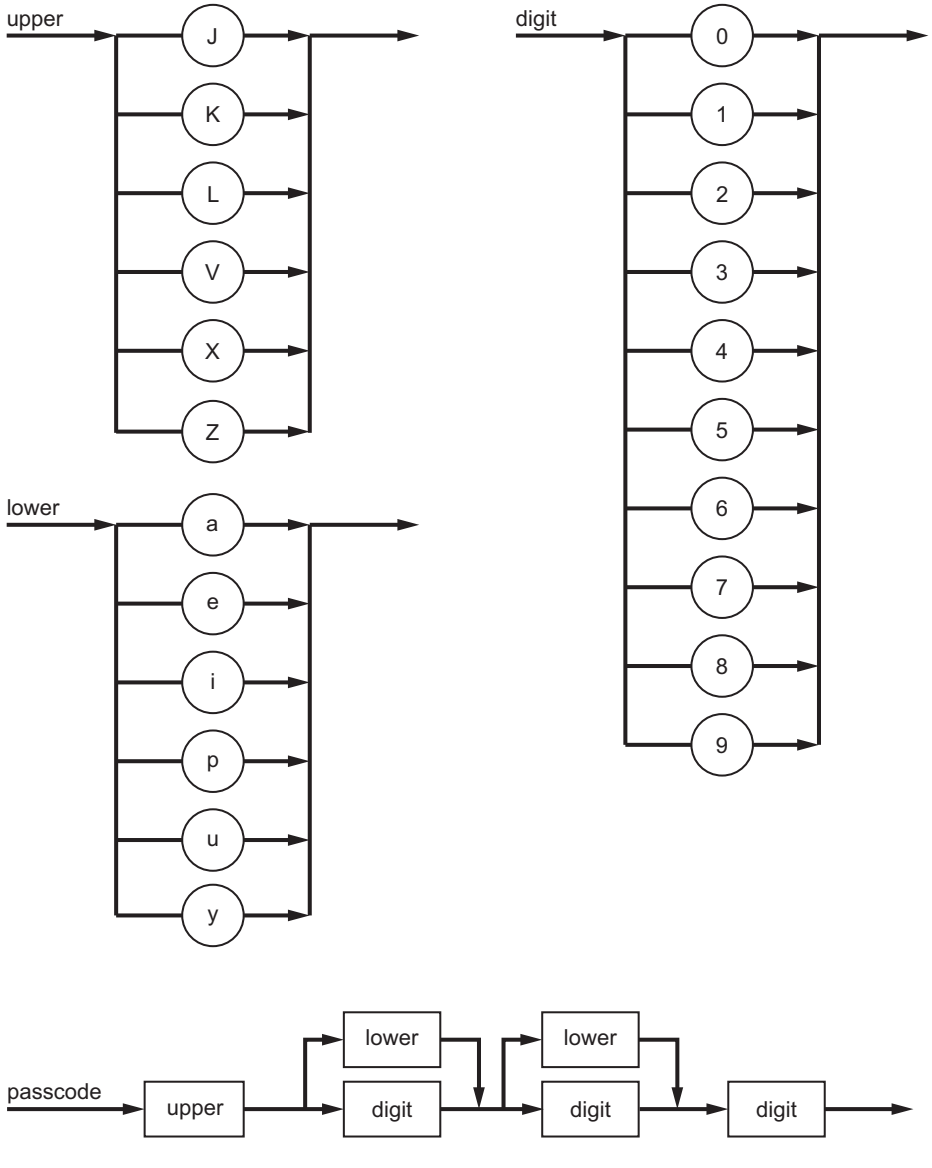
.....

.....

.....

..... [2]

9 Several syntax diagrams are shown.



(a) State why `JJ90` is **not** a valid passcode for the given syntax diagrams.

.....

..... [1]

(b) Complete the Backus-Naur Form (BNF) for `<upper>` and `<passcode>`.

`<upper> ::=` .....

.....

`<passcode> ::=` .....

.....

.....

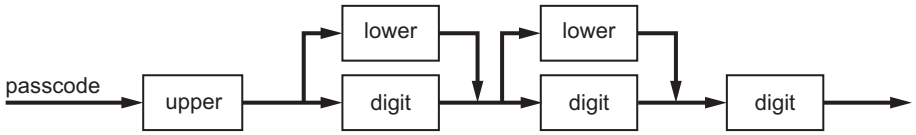
.....

[3]

(c) A character can be an upper, a lower or a digit.

The rules for passcode have been changed so that the third character may also be selected from upper and the final character may be repeated one or more times.

Complete the syntax diagram for passcode to show these changes.



[2]

2 A linear queue data structure is designed using a record structure.

The record structure `Queue` has the following fields:

- `QueueArray`, a 1D array of up to 100 integer values
- `Headpointer`, a variable that stores the index of the first data item in `QueueArray`
- `Tailpointer`, a variable that stores the index of the next free location in `QueueArray`.

(a) Write program code to declare the record structure `Queue` and its fields.

If your programming language does **not** support record structures, a class can be declared instead.

If you are writing in Python, use comments to declare the appropriate data types.

Save your program as **Question2_N24**.

Copy and paste the program code into part **2(a)** in the evidence document.

[3]

(b) The main program creates a new `Queue` record with the identifier `TheQueue`. The head pointer is initialised to –1. The tail pointer is initialised to 0. Each element in the array is initialised with –1.

Write program code for the main program.

Save your program.

Copy and paste the program code into part **2(b)** in the evidence document.

[3]

(c) The pseudocode function `Enqueue()` inserts an integer value into the queue.

The function is incomplete. There are **three** incomplete statements.

```
FUNCTION Enqueue(BYREF AQueue : Queue, BYVAL TheData : INTEGER)
                 RETURNS INTEGER

   IF AQueue.Headpointer = -1 THEN

      AQueue.QueueArray[AQueue.Tailpointer] ⟵ .....

      AQueue.Headpointer ⟵ 0

      AQueue.Tailpointer ⟵ AQueue.Tailpointer + 1

      RETURN 1

   ELSE

      IF AQueue.Tailpointer > ..... THEN

         RETURN -1

      ELSE

         AQueue.QueueArray[AQueue.Tailpointer] ⟵ TheData

         AQueue.Tailpointer ⟵ AQueue.Tailpointer .....

         RETURN 1

      ENDIF

   ENDIF

ENDFUNCTION
```

Write program code for `Enqueue()`.

Save your program.

Copy and paste the program code into part **2(c)** in the evidence document.

[5]

(d) The function `ReturnAllData()` accesses `TheQueue`. It concatenates all the integer values that have been inserted into the queue's array, starting from the value stored at `HeadPointer`, with a space between each integer value. The string of concatenated values is returned.

None of the integer values are removed from the queue.

Write program code for `ReturnAllData()`.

Save your program.

Copy and paste the program code into part **2(d)** in the evidence document.

[3]

(e) **(i)** The main program asks the user to enter 10 integers with values of 0 or greater. It reads each input repeatedly until a valid number is entered.

All 10 valid inputs are added to the queue, using `Enqueue()`.

If the value returned from `Enqueue()` is –1, a message is output to state that the queue is full, otherwise a message is output to state that the item has been added to the queue.

The function `ReturnAllData()` is called once all 10 integers have been entered and the return value from the function call is output.

Amend the main program to perform these actions.

Save your program.

Copy and paste the program code into part **2(e)(i)** in the evidence document.

[5]

(ii) Test your program with the following inputs in the order given:

10 9 –1 8 7 6 5 4 3 2 1

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **2(e)(ii)** in the evidence document.

[2]

(f) The function `Dequeue()` accesses `TheQueue`. The function returns –1 if the queue is empty. If the queue is **not** empty, the function returns the next item in the queue and updates the relevant pointer(s).

The data is **not** replaced or deleted from the queue.

Write program code for `Dequeue()`.

Save your program.

Copy and paste the program code into part **2(f)** in the evidence document.

[4]

(g) **(i)** The main program calls `Dequeue()` twice, and each time it either outputs 'Queue empty' if there is no data in the queue or outputs the return value.

The main program then calls `ReturnAllData()` a second time.

Amend the main program.

Save your program.

Copy and paste the program code into part **2(g)(i)** in the evidence document.

(ii) Test your program with the following inputs in the order given:

10 9 8 7 6 5 4 3 2 1

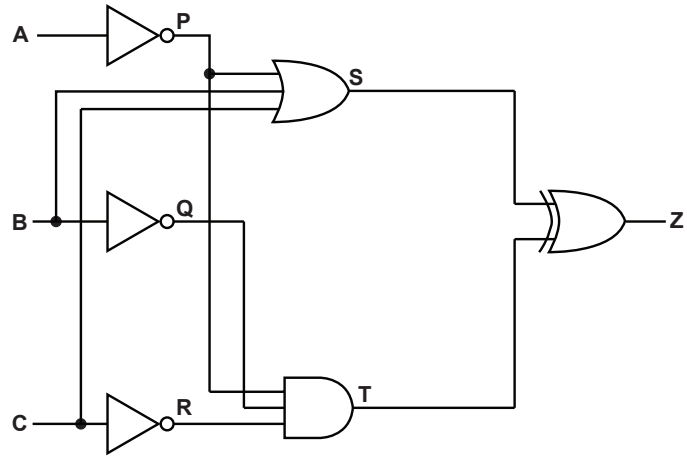
Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **2(g)(ii)** in the evidence document.

[1]

6 The diagram shows a logic circuit.



(a) Complete the truth table for the given logic circuit.
Show your working.

			Working space					
A	**B**	**C**	**P**	**Q**	**R**	**S**	**T**	**Z**
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

[3]

(b) Write the Boolean expression that corresponds to the logic circuit as a sum-of-products.

Z = ...

.....

.....

..... [2]

(c) **(i)** Complete the Karnaugh map (K-map) for this Boolean expression:

$\overline{A}.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.\overline{C} + A.B.\overline{C} + A.B.C$

A \ BC	00	01	11	10
0				
1				

[2]

(ii) Draw loop(s) around appropriate group(s) in the K-map to produce an optimal sum-of-products. [2]

(iii) Write the Boolean expression from your answer to part **c(ii)** as a simplified sum-of-products.

.....

..... [1]

2 The TCP/IP protocol suite has four layers:

Transport, Application, Link, Internet

(a) Complete the diagram to show the correct order for these layers.

[2]

(b) Describe the function of the Transport layer.

.....

.....

.....

..... [2]

(c) Outline **one** protocol that is associated with the Application layer.

.....

.....

.....

..... [2]