6 **(a)** Artificial Intelligence (AI) can be aided by the use of different techniques.

Draw a line from each technique to the correct description.

Technique	**Description**
	A structure used to model relationships between objects.
Artificial Neural Network	A computer system modelled on a brain.
A* Algorithm	A computer program that improves its performance at certain tasks with experience.
Graph	An abstract data type with a hierarchical structure.
Machine Learning	A computer method used to find the optimal path between two mapped locations.

[4]

(b) Describe **two** categories of machine learning.

1 ....

....

....

....

2 ....

....

....

....

[4]

7 An ordered binary tree Abstract Data Type (ADT) has these associated operations:

- create tree
- add new item to tree
- traverse tree

A student is designing a program that will implement a binary tree ADT as a linked list of **ten** nodes.

Each node consists of data, a left pointer and a right pointer.

A program is to be written to implement the tree ADT. The variables and procedures to be used are listed below:

Identifier	**Data type**	**Description**
`Node`	`RECORD`	Data structure to store node data and associated pointers.
`LeftPointer`	`INTEGER`	Stores index of start of left subtree.
`RightPointer`	`INTEGER`	Stores index of start of right subtree.
`Data`	`STRING`	Data item stored in node.
`Tree`	`ARRAY`	Array to store nodes.
`NewDataItem`	`STRING`	Stores data to be added.
`FreePointer`	`INTEGER`	Stores index of start of free list.
`RootPointer`	`INTEGER`	Stores index of root node.
`NewNodePointer`	`INTEGER`	Stores index of node to be added.
`CreateTree()`		Procedure initialises the root pointer and free pointer and links all nodes together into the free list.
`AddToTree()`		Procedure to add a new data item in the correct position in the binary tree.
`FindInsertionPoint()`		Procedure that finds the node where a new node is to be added. Procedure takes the parameter `NewDataItem` and returns two parameters: • `Index`, whose value is the index of the node where the new node is to be added • `Direction`, whose value is the direction of the pointer (“`Left`” or “`Right`”).

These pseudocode declarations and this procedure can be used to create an empty tree with ten nodes.

```
TYPE Node
   DECLARE LeftPointer : INTEGER
   DECLARE RightPointer: INTEGER
   DECLARE Data : STRING
ENDTYPE
DECLARE Tree  : ARRAY[0 : 9] OF Node
DECLARE FreePointer : INTEGER
DECLARE RootPointer : INTEGER

PROCEDURE CreateTree()
   DECLARE Index : INTEGER
   RootPointer ← -1
   FreePointer ← 0
   FOR Index ← 0 TO 9  // link nodes
      Tree[Index].LeftPointer ← Index + 1
      Tree[Index].RightPointer ← -1
   NEXT
   Tree[9].LeftPointer ← -1
ENDPROCEDURE
```

(a) Complete the pseudocode to add a data item to the tree.

```
PROCEDURE AddToTree(BYVALUE NewDataItem : STRING)
// if no free node report an error

   IF FreePointer .................................................................
      THEN
         OUTPUT "No free space left"
      ELSE
         // add new data item to first node in the free list
         NewNodePointer ← FreePointer

         ..............................................................................
         // adjust free pointer

         FreePointer ← ..................................................................
         // clear left pointer

         Tree[NewNodePointer].LeftPointer ← ..............................
         // is tree currently empty?

         IF ..........................................................................
            THEN // make new node the root node

               ......................................................................
            ELSE // find position where new node is to be added
               Index ← RootPointer
               CALL FindInsertionPoint(NewDataItem, Index, Direction)
               IF Direction = "Left"
                  THEN // add new node on left

                     ........................................................
                  ELSE // add new node on right

                     ........................................................
               ENDIF
         ENDIF
   ENDIF
ENDPROCEDURE
```

[8]

(b) The traverse tree operation outputs the data items in alphabetical order.
This can be written as a recursive solution.

Complete the pseudocode for the recursive procedure `TraverseTree`.

```
PROCEDURE TraverseTree(BYVALUE Pointer : INTEGER)
```

....

....

....

....

....

....

....

....

....

....

....

```
ENDPROCEDURE
```
[5]

2 Draw **one** line from each programming paradigm to its **most appropriate** description.

Programming paradigm	Description
	Programs using the instruction set of a processor
Declarative	Programs based on events such as user actions or sensor outputs
Imperative	Programs using the concepts of class, inheritance, encapsulation and polymorphism
Low-level	Programs with an explicit sequence of commands that update the program state, with or without procedure calls
Object-oriented	Programs that specify the desired result rather than how to get to it

[4]

3 Enumerated and pointer are two non-composite data types.

(a) Write **pseudocode** to create an enumerated type called `Parts` to include these parts sold in a computer shop:

`Monitor, CPU, SSD, HDD, LaserPrinter, Keyboard, Mouse`

DECLARE TYPE

Parts (Monitor, CPU, SSD, HDD, LaserPrinter, Keyboard, Mouse)

END TYPE [2]

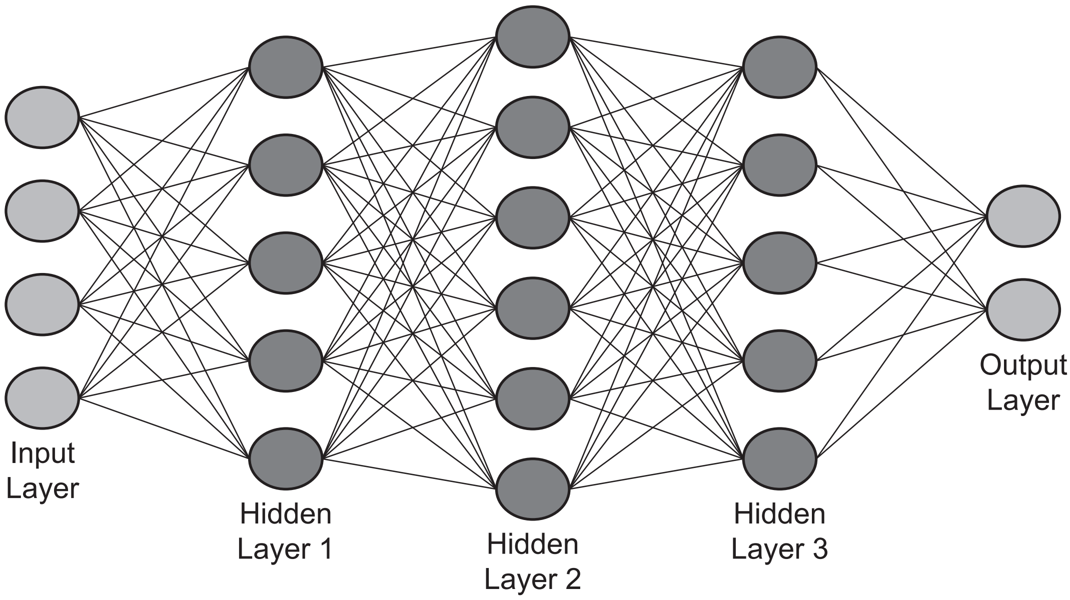
(b) Write **pseudocode** to create a pointer type called `SelectParts` that will reference the memory location in which the current part name is stored.

....

....

.... [2]

9 **(a)** The diagram shown represents an artificial neural network.



(i) State the reason for having multiple hidden layers in an artificial neural network.

....

.... [1]

(ii) Explain how artificial neural networks enable machine learning.

....

....

....

....

....

....

....

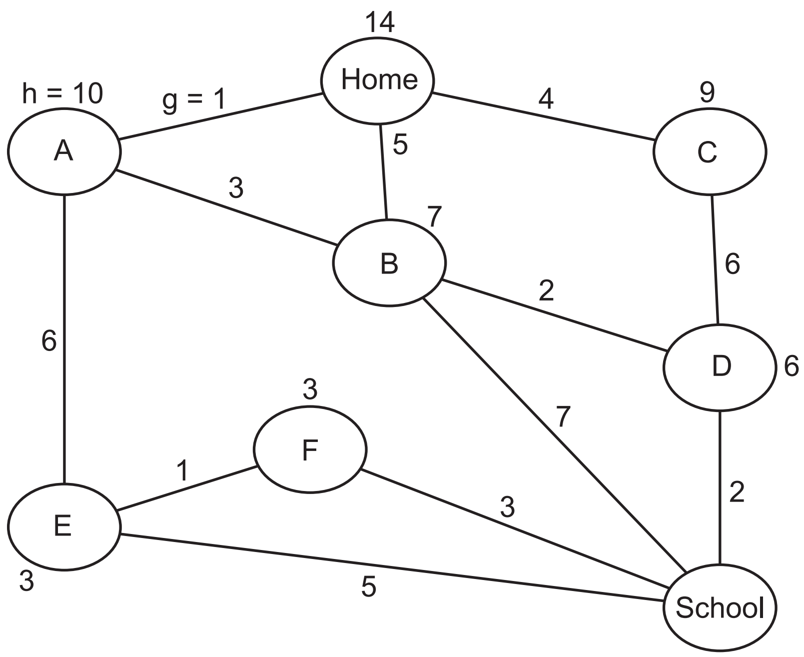
....

....

.... [4]

(b) Find the shortest path between the Home and School nodes using the A* algorithm. Show your working in the table provided.

The first two rows in the table have been completed.



Node	Cost from Home node (g)	Heuristic (h)	Total (f = g + h)
Home	0	14	14
A	1	10	11

Final path	

[5]

10 **(a)** State **three** essential features of **recursion**.

1 ....

....

2 ....

....

3 ....

....
[3]

(b) Explain the reasons why a stack is a suitable Abstract Data Type (ADT) to implement recursion.

....

....

....

....

....

.... [3]

(c) Identify **two** ADTs other than a stack.

1 ....

2 ....
[2]

(d) The function `StackFull()` checks whether a stack is full.

The function uses the variable `TopOfStack` to represent the pointer to the most recent position used on the stack, and the variable `Max` to represent the maximum size of the stack. Assume `TopOfStack` and `Max` are global variables.

```
FUNCTION StackFull() RETURNS BOOLEAN
    IF TopOfStack = Max THEN
       RETURN TRUE
    ELSE
       RETURN FALSE
    ENDIF
ENDFUNCTION
```

An algorithm `AddInteger` is required to add a new integer data element to a stack.

The stack is implemented as an array `ArrayStack`.

The function `AddInteger()` calls `StackFull()` and returns an appropriate message.

Complete the pseudocode for the function `AddInteger()`.

```
FUNCTION AddInteger(NewInteger : INTEGER) RETURNS STRING
```

....

....

....

....

....

....

....

....

....

....

....

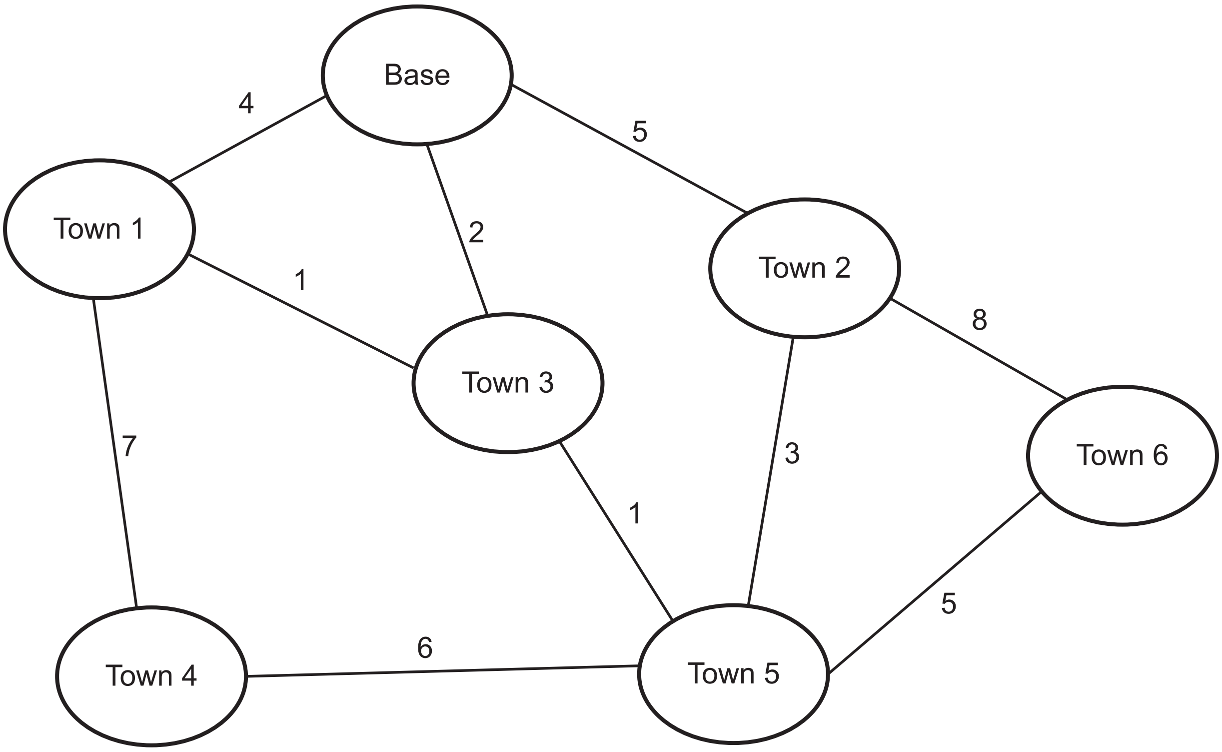
....

```
ENDFUNCTION
```

[5]

5 **(a)** Calculate the shortest distance between the base and each of the other towns in the diagram using Dijkstra's algorithm.

Show your working **and** write your answers in the table provided.



Working ....

....

....

....

....

....

....

....

Answers

Town 1	Town 2	Town 3	Town 4	Town 5	Town 6

[5]

(b) Explain the use of graphs to aid Artificial Intelligence (AI).

....

....

....

....

....

.... [3]

6 Give **two** benefits **and two** drawbacks of packet switching.

Benefit 1 ....

....

Benefit 2 ....

....

Drawback 1 ....

....

Drawback 2 ....

....
[4]

8 (a) State **two** factors that may affect the performance of a sorting algorithm.

....

....

....

.... [2]

(b) The given algorithm is a simple bubble sort that arranges a set of scores stored in a one-dimensional array into **descending** order, and orders the corresponding students' names stored into a two-dimensional array in the same order as the scores. All the arrays are indexed from 1.

The contents of both arrays after sorting are shown.

	Score
1	98
2	97
...	
248	5
249	3

	Name	
	1	**2**
1	Smithfield	Tom
2	Johnson	Jane
...		
248	Peters	Jade
249	Allen	John

```
YearSize ← 249
Flag ← TRUE
WHILE Flag = TRUE
    Flag ← FALSE
    FOR Student ← 1 TO YearSize - 1
        IF Score[Student] < Score[Student + 1] THEN
           Temp1 ← Score[Student]
           Temp2 ← Name[Student,1]
           Temp3 ← Name[Student,2]
           Score[Student] ← Score[Student + 1]
           Name[Student,1] ← Name[Student + 1,1]
           Name[Student,2] ← Name[Student + 1,2]
           Score[Student + 1] ← Temp1
           Name[Student + 1,1] ← Temp2
           Name[Student + 1,2] ← Temp3
           Flag ← TRUE
        ENDIF
    NEXT Student
ENDWHILE
```

Write an algorithm, using pseudocode, that will perform the same task using an insertion sort.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

9 **(a)** Describe what is meant by **an imperative (procedural)** programming language.

....

....

....

.... [2]

(b) Describe what is meant by **a declarative** programming language.

....

....

....

.... [2]

(c) Identify the programming paradigm for each of these program code examples.

<table>
<tr><th>Program code example</th><th>Programming paradigm</th></tr>
<tr><td><pre>male(john).
female(ethel).
parent(john, ethel).</pre></td><td>declarative</td></tr>
<tr><td><pre>FOR Counter = 1 TO 20
    X = X * Counter
NEXT Counter</pre></td><td>imperative</td></tr>
<tr><td><pre>Start: LDD Counter
       INC ACC
       STO Counter</pre></td><td>low level</td></tr>
<tr><td><pre>public class Vehicle
{
    private speed;
    public Vehicle()
    {
        speed = 0;
    }
}</pre></td><td>object oriented</td></tr>
</table>

[4]

3 A zoo reptile house has sixteen tanks which accommodate its reptiles. Each tank has to have its own microclimate where the appropriate levels of heat and humidity are crucial. The zoo implements a computer system which supplies the conditions in each of the tanks to a terminal in a central area. Warning messages are flashed up on the screen if any condition arises which requires the intervention of a zoo-keeper.

(a) State the name of the type of computing system described.

.... [1]

(b) State **two** items of hardware which need to be present in the tanks for this system to function correctly.

1 ....

2 .... [2]

(c) This is the polling routine which is used to run the system indefinitely.

```
REPEAT
  FOR i ← 1 TO ..........
    READ Condition1, Condition2 in tank(i)
    IF Condition1 < Extreme[i,1] OR Condition1 > Extreme[i,2]
      THEN
        OUTPUT "Warning! Problem in Tank ", i
    ENDIF
    IF Condition2 < Extreme[i,3] OR Condition2 > Extreme[i,4]
      THEN
        OUTPUT "Warning! Problem in Tank ", i
    ENDIF
  ENDFOR

  FOR i ← 1 TO 999999
  ENDFOR
UNTIL ..............................
```

(i) Fill in the gaps in the pseudocode. [2]

(ii) Explain what is stored in the array `Extreme`.

....

....

....

.... [2]

(iii) Explain what happens in lines `04` to `11`.

....

....

....

....

....

.... [3]

(iv) Explain the purpose of the loop in lines `14` to `15`.

....

.... [1]

[6]

6 Raz and Tan wish to exchange some sensitive information via a message in an email. Initially, Raz wants to send the message to Tan in such a way that Tan can be assured that the message did come from Raz.

(a) The steps are as follows.

1. Raz creates a **<answer 1>** using a **<answer 2>** function on the message.
2. Raz encrypts the **<answer 1>** using his **<answer 3>** key. This is the digital **<answer 4>** for the message.
3. Raz sends both the message and the digital **<answer 4>** to Tan.
4. Tan decrypts the digital **<answer 4>** using Raz's **<answer 5>** key.
5. Tan repeats what Raz did in Step 1 to the message.

Select from the list of terms to complete the five statements.

signature hash message-digest encryption private public email

<answer 1> ....

<answer 2> ....

<answer 3> ....

<answer 4> ....

<answer 5> .... [5]

(b) Tan finds that her results in Step 5 do not match her results in Step 4.

Give **two** possible reasons for this.

1 ....

....

2 ....

.... [2]

(c) Even though Tan's results in Step 5 match the results in Step 4, she is still concerned that anybody receiving the message can actually read the contents.

Explain what Raz and Tan need to do so that only Tan can read the message.

....

....

....

....

....

....

....

....

.... [3]

7 The following are the first few lines of a source code program written in a high-level language. The source code program is to be translated by the language compiler.

```
// program written on 15 June 2019

DECLARE IsFound : Boolean;
DECLARE NoOfChildren : Integer;
DECLARE Count : Integer;
Constant TaxRate = 15;

// start of main program
For Count = 1 to 50
...
...
...
```

(a) During the lexical analysis stage, the compiler will use a keyword table and a symbol table.

(i) Identify **two** types of data in the keyword table.

Type 1 ....

Type 2 .... [2]

(ii) Identify **two** types of data in the symbol table.

Type 1 ....

Type 2 .... [2]

(iii) Explain how the contents of the keyword and symbol tables are used to translate the source code program.

....

....

....

.... [2]

(iv) State **one** additional task completed at the lexical analysis stage that does not involve the use of a keyword or a symbol table.

....

.... [1]

(b) The final stage of compilation can be code optimisation.

Explain why code is optimised.

....

....

....

.... [2]

4 A compiler uses a keyword table and a symbol table. Part of the keyword table is shown.

- Tokens for keywords are shown in hexadecimal.
- All of the keyword tokens are in the range 00 – 5F.

Keyword	**Token**
←	01
+	02
=	03
<>	04
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
REPEAT	4E
UNTIL	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from 60 (hexadecimal).

Study the following piece of pseudocode.

```
Counter ← 0
INPUT Password
REPEAT
   IF Password <> "Cambridge"
      THEN
         INPUT Password
   ENDIF
   Counter ← Counter + 1
UNTIL Password = "Cambridge"
OUTPUT Counter
```

(a) Complete the symbol table to show its contents after the lexical analysis stage.

Symbol	**Token**	
	Value	**Type**
`Counter`	`60`	Variable

[3]

(b) The output from the lexical analysis stage is stored in the following table. Each cell stores one byte of the output.

Complete the output from the lexical analysis using the keyword table **and** your answer to **part (a)**.

60	01																							

[2]

6 The compilation process has a number of stages. The first stage is lexical analysis.

A compiler uses a keyword table and a symbol table. Part of the keyword table is shown.

- Tokens for keywords are shown in hexadecimal.
- All of the keyword tokens are in the range `00` – `5F`.

Keyword	**Token**
←	01
*	02
=	03
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
FOR	4E
STEP	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from 60 (hexadecimal).

Study the following code.

```
Start ← 1
INPUT Number
// Output values in a loop
FOR Counter ← Start TO 12
    OUTPUT Number * Counter
ENDFOR
```


(a) Complete the symbol table to show its contents after the lexical analysis stage.

Symbol	**Token**	
	Value	**Type**
`Start`	`60`	Variable
`1`	`61`	Constant

[3]

(b) The output from the lexical analysis stage is stored in the following table. Each cell stores one byte of the output.

Complete the output from the lexical analysis stage. Use the keyword table and your answer to **part (a)**.

`60`	`01`														

[2]

(c) The output of the lexical analysis stage is the input to the syntax analysis stage.

Identify **two** tasks in syntax analysis.

1 ....

....

2 ....

....
[2]

(d) The final stage of compilation is optimisation.

(i) Code optimisation produces code that minimises the amount of memory used.

Give **one** additional reason why code optimisation is performed.

....

....[1]

2 The following incomplete table shows descriptions and terms relating to malware.

(a) Complete the table with appropriate description and terms.

	Description	Term	
(i)	A standalone piece of malicious software that can replicate itself using a network.	....	[1]
(ii)	Use email to attempt to obtain an individual's confidential data.	....	[1]
(iii)	....	Virus	[2]

(b) State **two** vulnerabilities that the malware in **part (a)(i)** or **part (a)(iii)** can exploit.

Vulnerability 1 ....

....

Vulnerability 2 ....

.... [2]

Question 2 continues on the next page.

4 A bank has 95 000 customers. Each customer has a unique ID.

When a customer uses an Automated Teller Machine (ATM) to obtain cash, their current balance is checked. The balance is stored in a file which has the following fields:

- the customer ID (6-digit number in the range 100000 to 999999)
- an encrypted PIN
- the current balance

The file can store a maximum of 100 000 records.

(a) Give a reason why a random organisation would be appropriate for this file.

....

....[1]

(b) An algorithm for inserting a new record in this file uses the following hash function:

```
RecordKey ← CustomerID MOD 100000
```

where `RecordKey` is the record position in the file.

(i) Complete the table to show the values generated by the hash function for the given customer IDs.

CustomerID	RecordKey
802139	2139
700004	
689998	
102139	

[1]

(ii) State the range of possible values for `RecordKey`.

Minimum value of `RecordKey`: ....

Maximum value of `RecordKey`: ....
[2]

(iii) A procedure is written to insert a new record into the file.

Complete the algorithm for this procedure.

```
PROCEDURE InsertRecord(CustomerID : INTEGER)
   RecordKey ← CustomerID MOD 100000
   Success ← FALSE
   // Find position for new record and insert it
   REPEAT
      IF record at position RecordKey is ..........................
         THEN
            Insert new record at position RecordKey
            Success ← TRUE
         ELSE
            IF RecordKey = ..........................
               THEN
                  RecordKey ← ..........................
               ELSE
                  RecordKey ← .......................... + 1
            ENDIF
      ENDIF
   UNTIL Success = TRUE
ENDPROCEDURE
```

[4]

(c) (i) Explain why an encrypted version of the PIN is stored in the file.

....

....

....

....[2]

(ii) A customer attempts to withdraw cash from an ATM. An algorithm is used to check if the customer has entered the correct PIN.

Complete the algorithm.

```
1. Customer ID is read from card.

2. Customer enters PIN.

3. Customer PIN is ..........................................................

4. ...........................................................................

5. Customer record is located in file.

6. ...........................................................................

7. If match then transaction can proceed.
```

[3]

6 A large office building has many floors. On each floor there are security sensors and security cameras. There is the same number of sensors on each floor. The building has a single security room.

The images from the security cameras are output on monitors (one monitor for each floor) placed in the security room.

The data from the sensors are read and processed by a computer system. Sensor readings and warning messages can be displayed on the monitors.

(a) **(i)** State the name given to the type of system described.

....[1]

(ii) Explain your answer to **part (i)**.

....

....[1]

(iii) State **two** sensors that could be used in this system.

Sensor 1 ....

Sensor 2 ....
[2]

(b) A software routine:

- checks the readings from the sensors
- outputs readings and warning messages to the monitors
- loops continuously.

The routine uses the following pseudocode variables:

Identifier	Data type	Description
`FloorCounter`	`INTEGER`	Loop counter for number of floors
`SensorCounter`	`INTEGER`	Loop counter for number of sensors
`NumberOfFloors`	`INTEGER`	Stores the number of floors
`NumberOfSensors`	`INTEGER`	Stores the number of sensors
`ForEver`	`BOOLEAN`	Stores value that ensures continuous loop

(i) Complete the following pseudocode algorithm for the routine.

```
01 ForEver ← ......................................................
02 REPEAT
03   FOR FloorCounter ← 1 TO NumberOfFloors
04     FOR SensorCounter ← 1 TO ...................................
05       READ Sensor(SensorCounter)on Floor(FloorCounter)
06       IF Sensor value outside range
07          THEN
08             OUTPUT "Problem on Floor ", FloorCounter
09       ENDIF
10     ENDFOR
11   ENDFOR
12   //
13   // Delay loop
14   // Delay loop
15   //
16 UNTIL ..........................................................
```

[3]

(ii) A delay needs to be introduced before the loop is processed again.

Write a `FOR` loop, in pseudocode, to replace lines `13` and `14`.

....

....[1]

(iii) Give a reason for this delay in the system.

....

....[1]