1(a).
A programmer needs to sort an array of numeric data using an insertion sort.

i. The following, incomplete, algorithm performs an insertion sort.

Complete the algorithm.

```
procedure sortit(dataArray, lastIndex)
  for x = 1 to lastIndex
    currentData = dataArray[……]
    position = x
    while (position > 0 AND dataArray[position-1] > currentData)
      dataArray[position]  = dataArray[……]
      position = position - 1
    endwhile

    dataArray[position] = ……
  next x
endprocedure
```

currentData = dataArray

```
ray[] dataArray[pos] = dataArray[pos
```

ii. Show how an insertion sort would sort the following data: **[6]**

6	**1**	**15**	**12**	**5**	**6**	**9**

(c). The number of data items in the array is continually increasing.

Insertion sort has a worst case time complexity of $O(n^2)$ and space complexity of O(1).

An alternative sorting algorithm that could be used is bubble sort which also has a worst case time complexity of $O(n^2)$ and space complexity of O(1).

Briefly outline how the bubble sort algorithm works. Discuss the relationship between the complexities and the two sorting algorithms and justify which of the two algorithms is best suited to sorting the array. **[9]**

2(a). A programmer is developing an ordering system for a fast food restaurant. When a member of staff inputs an order, it is added to a linked list for completion by the chefs.

Explain why a linked list is being used for the ordering system.

[2]

(b). Each element in a linked list has:

- a pointer, `nodeNo`, which gives the number of that node
- the order number, `orderNo`
- a pointer, `next`, that points to the next node in the list

Fig. 2.1 shows the current contents of the linked list, `orders`.

nodeNo	orderNo	next
0	154	1
1	157	2
2	155	3
3	156	∅

Fig. 2.1

∅ represents a null pointer.

i. Order 158 has been made, and needs adding to the end of the linked list.

Add the order, 158, to the linked list as shown in Fig. 2.1. Show the contents of the linked list in the following table.

nodeNo	orderNo	next

ii.

iii. Order 159 has been made. This order has a high priority and needs to be the second order in the linked list.
Add the order, 159, to the original linked list as shown in Fig. 2.1. Show the contents of the linked list in the following table.

nodeNo	orderNo	next

iv.

(c). The linked list is implemented using a 2D array, `theOrders`:

- Row 0 stores `orderNo`
- Row 1 stores `next`

The data now stored in `theOrders` is shown in Fig. 2.2.

184	186	185	187
1	2	3	

Fig. 2.2

`theOrders [1,0]` would return `1`
The following algorithm is written:

```
procedure x()
    finished = false
    count = 0
    while NOT(finished)
        if theOrders[1,count] == null then
            finished = true
        else
            output = theOrders[0,count]
            print(output)
            count = theOrders[1,count]
        endif
    endwhile
    output = theOrders[0,count]
    print(output)

endprocedure
```

i. Outline why `nodeNo` does not need to be stored in the array.

[1]

ii. Complete the trace table for procedure `x`, for the data shown in Fig. 2.2.

finished	count	output

iii.

iv. Describe the purpose of procedure `x`.

[2]

v. A new order, 190, is to be added to `theOrders`. It needs to be the third element in the list.
The current contents of the array are repeated here for reference:

184	186	185	187		
1	2	3			

Describe how the new order, 190, can be added to the array, so the linked list is read in the correct order, without rearranging the array elements.

[4]

(d). The user needs to be able to search for, and find, a specific order number.

State an appropriate search algorithm that could be used, and justify your choice against an alternative Search algorithm.
Appropriate Search
Algorithm
Justification

[3]

(e). The programmer is writing the program using an IDE.

Identify **three** features of an IDE that the programmer would use when writing the code and describe how the features benefit the programmer.
1

2

3

[6]

(f). 

The programmer is considering using concurrent programming.

Discuss how concurrent programming can be applied to the food ordering system and the benefits and limitations of doing so. **[9]**

3(a). An encryption routine reads a line of text from a file, reverses the order of the characters in the string and subtracts 10 from the ASCII value of each letter, then saves the new string into the same file.

The program is split into sub-procedures. Three sub-procedures are described as follows:

- Read string from file
- Push each character of the string onto a stack
- Read and encrypt each character message

i. Identify **one** further sub-procedure that could be used in the program.

[1]

ii. Describe **two** advantages of splitting the problem into sub-procedures.

1

2

iii.

(b). A function, `readMessage`:

- takes the file name as a parameter
- reads and returns the line of text

Complete the pseudocode algorithm for `readMessage`:

```
function ……(fileName)
    messageFile = openRead(……)
    message = messageFile.readLine()
    messageFile. ……
    return ……
endfunction
```

[4]

(c). A function, `push`, can be used to add a character to a stack. For example:

```
theStack.push("H")
```

places the character `H` onto the stack, `theStack`.

A procedure, `pushToStack`, takes a string as a parameter and pushes each character of the message onto the stack, `messageStack`.

Complete the procedure below.

Add comments to explain how your code works.

```
procedure pushToStack(message)

endprocedure
```

[5]

(d). Describe the steps that the program would have to take in order to encrypt the characters stored in the stack, and save them in a single variable.

[5]

4(a). A data structure is shown below in Fig. 4.1.

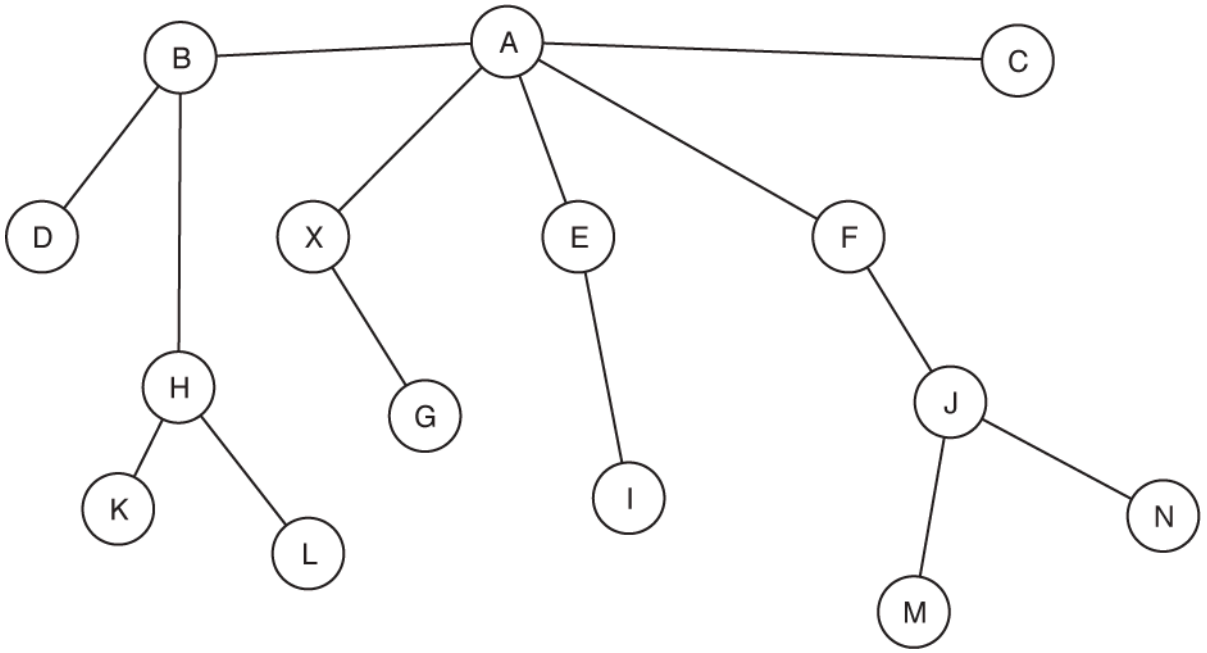


Fig. 4.1

Identify the data structure shown in Fig. 4.1.

[1]

(b). The programmer is considering using a depth-first (post-order) traversal, or a breadth-first traversal to find the path between node A and node X.

i. Explain the difference between a depth-first (post-order) and breadth-first traversal.

[4]

ii. Show how a depth-first (post-order) traversal would find the path between node A and node X for the structure shown in Fig. 4.1.

[6]

iii. Explain how you used backtracking in your answer to part **(b)(ii)**.

[3]

5(a). A recursive function, `calculate`, is shown below:

```
01 function calculate(num1, num2)
02    if num1 == num2 then
03        return num1
04    elseif num1 < num2 then
05        return calculate(num1, (num2-num1))
06    else
07        return calculate(num2, (num1-num2))
08    endif
09 endfunction
```

Identify the lines where recursion is used.

[1]

(b). Trace the algorithm, showing the steps and result when the following line is run:

```
print(calculate(4,10))
```

[5]

(c). Re-write the function so it uses iteration instead of recursion.

[4]

6(a). A software developer is creating a Virtual Pet game.

The user can choose the type of animal they would like as their pet, give it a name and then they are responsible for caring for that animal. The user will need to feed, play with, and educate their pet.

The aim is to keep the animal alive and happy, for example if the animal is not fed over a set period of time then the pet will die.

- The game tells the user how hungry or bored the animal is as a percentage (%) and the animal's intelligence is ranked as a number between 0 and 150 (inclusive).
- Hunger and boredom increase by 1% with every tick of a timer.
- When the feed option is selected, hunger is reduced to 0.
- When the play option is selected, bored is reduced to 0.
- When the read option is selected, the intelligence is increased by 0.6% of its current value.

An example of the game is shown:

```
What type of pet would you like? Fox or Elephant?
Fox
What would you like to name your Fox?
Joanne
Joanne's stats are
Hunger: 56%
Bored: 85%
Intelligence: 20
What would you like to do with your pet? Play, Read or Feed?
```

Fig. 1.1

Identify **three** inputs that the user will have to enter to start, and / or play the game.

1

2

3

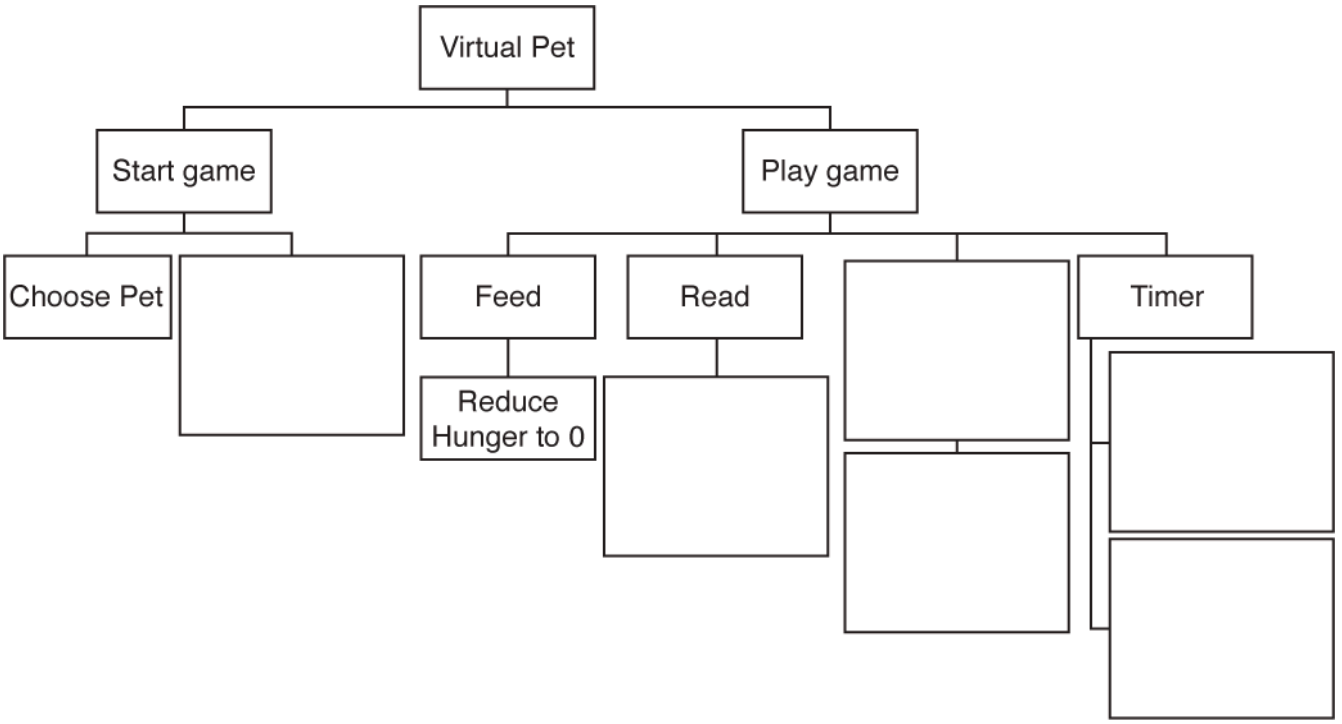
[3]

(b) The developer is using decomposition to design the game.

i. Describe the process of decomposition.

[2]

ii. The developer has produced the following structure diagram for the game:



Complete the structure diagram for the Virtual Pet game by filling in the empty boxes.

(c). The developer needs to write procedures for the options play and read. Each of the options changes its corresponding value, and outputs the results to the screen.

i. Write a procedure, using pseudocode, to reset `bored` and output the new value in an appropriate message.

[3]

ii. Write a procedure, using pseudocode, to increase `intelligence` by 0.6% and output the new intelligence in an appropriate message.

[3]

(d). The developer is extending the game to allow users to have multiple pets of different types. The developer has written a class, `Pet`.

The attributes and methods in the class are described in the table:

Identifier	Attribute / Method	Description
`petName`	Attribute	Stores the pet's name
`bored`	Attribute	Stores the % bored
`hunger`	Attribute	Stores the % hunger
`intelligence`	Attribute	Stores the intelligence
`type`	Attribute	Stores the type of animal
`new`	Method	Creates a new instance of `pet`
`feed`	Method	Reduces `hunger` to 0 and outputs `hunger`
`play`	Method	Reduces `bored` to 0 and outputs `bored`
`read`	Method	Increases `intelligence` by a set value
`outputGreeting`	Method	Outputs a message to the user

Part of the class declaration is given:

```
class Pet
private petName
private bored
private hunger
```

```
private intelligence
private type
  ...
  ...
```

i. After a user enters the pet name, and chooses a type, the constructor method of Pet is called to create a new instance. The method needs to set `petName`, as well as `hunger, bored` and `intelligence` to starting values of 0.

Write, using pseudocode, the constructor method for this class.

[4]

ii. Write a line of code that creates a new instance of `Pet` for a Tiger called “Springy”.

[2]

iii. The method `outputGreeting` for the superclass is written as follows:

```
public procedure outputGreeting()
print("Hello, I'm " + petName + ", I'm a " + type)
endprocedure
```

iv.

A class is needed for Tiger. The class needs to:

- inherit the methods and attributes from `pet`
- in the constructor, set `type` to Tiger, `intelligence` to 10, `hunger` to 50 and `bore` 10
- extend the method `outputGreeting`, by outputting an additional line that says “I l eat meat and roar”

v.

Write, using pseudocode, the class `Tiger`.

vi.

vii.

viii.

ix.

x.

xi.

xii.

xiii.

xiv.

xv.

xvi.

xvii.

xviii.

xix.

xx.

xxi.

xxii.

xiii.

xxiv.

xxv.

xxvi.

xvii.

xxviii.

xix.

xxx.

xxi.

xxxii.

xiii.

xxxiv.

xxv.

xxxvi.

xvii.

xxxviii. **[5]**

(e). 

The developer made use of abstraction when creating the Virtual Pet game.

Discuss the need for and purpose of abstraction and how abstraction will be used in the development of the game. **[9]**

(f). The developer is storing the user's pets in a 1- dimensional array. At each timer interval, the array is searched, using a linear search, to check if any pets' hunger or bored values are greater than 90%.
If they are, an alert is displayed to the user.

i. State the complexity of searching the pets in Big-O notation.

[1]

ii. A given computer takes 4 milliseconds (ms) to search an array of 20 pets. Calculate an estimate of how long the computer will take to search an array of 100 pets.

Show your working.

[2]

END OF QUESTION PAPER