QUESTION 4.

6 A queue Abstract Data Type (ADT) has these associated operations:

- create queue
- add item to queue
- remove item from queue

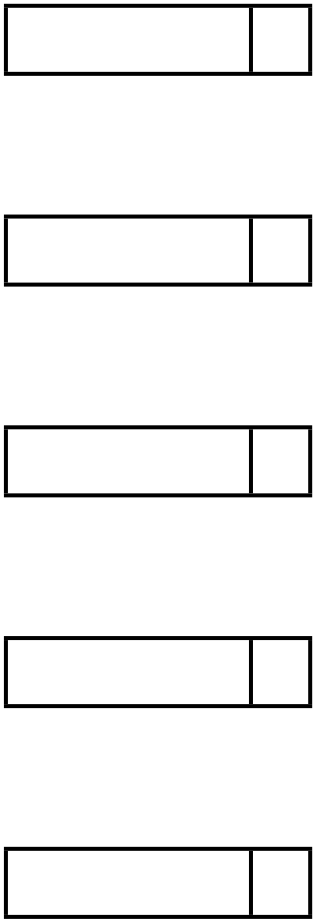
The queue ADT is to be implemented as a linked list of nodes.

Each node consists of data and a pointer to the next node.

(a) The following operations are carried out:

```
CreateQueue
AddName("Ali")
AddName("Jack")
AddName("Ben")
AddName("Ahmed")
RemoveName
AddName("Jatinder")
RemoveName
```

Add appropriate labels to the diagram to show the final state of the queue. Use the space on the left as a workspace. Show your final answer in the node shapes on the right:



[3]

(ii) Rewrite the algorithm in **part (b)**, using **pseudocode**, to reduce the [illegible] unnecessary comparisons. Use the same variable names where appropriate.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

QUESTION 5.

6 A recursively defined procedure X is defined below:

```
PROCEDURE X(BYVALUE n : INTEGER)
   IF (n = 0) OR (n = 1)
      THEN
         OUTPUT n
      ELSE
         CALL X(n DIV 2)
         OUTPUT (n MOD 2)
   ENDIF
ENDPROCEDURE
```

(a) Explain what is meant by recursively defined.

.....

..... [1]

(b) Explain how a stack is used during the execution of a recursive procedure.

.....

.....

.....

..... [2]

(c) Dry run the procedure X by completing the trace table for the procedure call:

```
CALL X(40)
```

Call number	n	(n = 0) OR (n = 1)	n DIV 2	n MOD 2
1	40	FALSE	20	
2				
3				
4				
5				
6				

OUTPUT ..... [6]

Complete the **pseudocode** for the procedure `Pop`. Use the variables listed in the identifier

```
PROCEDURE Pop()
  // Report error if Stack is empty
.....
.....
.....
.....
   OUTPUT Stack [.....].Name
   // take a copy of the current top of stack pointer
.....
   // update the top of stack pointer
.....
   // link released node to free list
.....
.....
.....
ENDPROCEDURE
```

[5]

QUESTION 6.

4 A binary tree Abstract Data Type (ADT) has these associated operations:

- create the tree (`CreateTree`)
- add an item to tree (`Add`)
- output items in ascending order (`TraverseTree`)

(a) Show the final state of the binary tree after the following operations are carried out.

```
CreateTree
Add("Dodi")
Add("Farai")
Add("Elli")
Add("George")
Add("Ben")
Add("Celine")
Add("Ali")
```

[4]

(b) The binary tree ADT is to be implemented as an array of nodes. Each node and two pointers.

Using pseudocode, a record type, `Node`, is declared as follows:

```
TYPE Node
   DECLARE Name : STRING
   DECLARE LeftPointer : INTEGER
   DECLARE RightPointer : INTEGER
ENDTYPE
```

The statement

```
DECLARE Tree : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array `Tree`.

The `CreateTree` operation links all nodes into a linked list of free nodes. It also initialises the `RootPointer` and `FreePointer`.

Show the contents of the `Tree` array and the values of the two pointers, `RootPointer` and `FreePointer`, after the operations given in **part (a)** have been carried out.

`RootPointer`

`FreePointer`

`Tree`

	Name	LeftPointer	RightPointer
[1]			
[2]			
[3]			
[4]			
[5]			
[6]			
[7]			
[8]			
[9]			
[10]			

[7]

(c) A programmer needs an algorithm for outputting items in ascending order. To
programmer writes a recursive procedure in pseudocode.

(i) Complete the pseudocode:

```
01 PROCEDURE TraverseTree(BYVALUE Root: INTEGER)
02     IF Tree[Root].LeftPointer ..........
03         THEN
04             TraverseTree(..........)
05     ENDIF
06     OUTPUT ..........Name
07     IF ..........<> 0
08         THEN
09             TraverseTree(..........)
10     ENDIF
11 ENDPROCEDURE
```

[5]

(ii) Explain what is meant by a recursive procedure. Give a line number from the code above that shows procedure `TraverseTree` is recursive.

.....

.....

.....

Line number .....[2]

(iii) Write the pseudocode call required to output all names stored in `Tree`.

.....

.....[1]

Question 5 begins on page 14.

QUESTION 7.

1 A linked list abstract data type (ADT) is to be used to store and organise surname

This will be implemented with a 1D array and a start pointer. Elements of the array c
user-defined type. The user-defined type consists of a data value and a link pointer.

Identifier	Data type	Description
`LinkedList`	`RECORD`	User-defined type
`Surname`	`STRING`	Surname string
`Ptr`	`INTEGER`	Link pointers for the linked list

(a) (i) Write **pseudocode** to declare the type `LinkedList`.

.....

.....

.....

.....[3]

(ii) The 1D array is implemented with an array `SurnameList` of type `LinkedList`.

Write the **pseudocode** declaration statement for `SurnameList`. The lower and upper bounds of the array are 1 and 5000 respectively.

.....[2]

(b) The following surnames are organised as a linked list with a start pointer `StartPtr`.

`StartPtr: 3`

	1	2	3	4	5	6	...	5000
Surname	Liu	Yang	Chan	Wu	Zhao	Huang	...	
Ptr	4	5	6	2	0	1	...	

State the value of the following:

(i) `SurnameList[4].Surname` .....[1]

(ii) `SurnameList[StartPtr].Ptr` .....[1]

(c) Pseudocode is to be written to search the linked list for a surname input by t

Identifier	Data type	Description
ThisSurname	STRING	The surname to search for
Current	INTEGER	Index to array SurnameList
StartPtr	INTEGER	Index to array SurnameList. Points to the element at the start of the linked list

(i) Study the pseudocode in **part (c)(ii)**.

Complete the table above by adding the missing identifier details. [2]

(ii) Complete the pseudocode.

```
01  Current ← .....
02  IF Current = 0
03     THEN
04        OUTPUT .....
05     ELSE
06        IsFound ← .....
07        INPUT ThisSurname
08        REPEAT
09           IF ..... = ThisSurname
10              THEN
11                 IsFound ← TRUE
12                 OUTPUT "Surname found at position ", Current
13              ELSE
14                 // move to the next list item
15                 .....
16           ENDIF
17        UNTIL IsFound = TRUE OR .....
18        IF IsFound = FALSE
19           THEN
20              OUTPUT "Not Found"
21        ENDIF
22  ENDIF
```

[6]

QUESTION 8.

4 An ordered linked list Abstract Data Type (ADT) has these associated operations:

- create list
- add item to list
- output list to console

The ADT is to be implemented using object-oriented programming as a linked list of nodes.

Each node consists of data and a pointer.

(a) There are two classes, `LinkedList` and `Node`.

(i) State the term used to describe the relationship between these classes.

.....[1]

(ii) Draw the appropriate diagram to represent this relationship. Do not list the attributes and methods of the classes.

[2]

3 `NameList` is a 1D array that stores a sorted list of names. A programmer declares the array in pseudocode as follows:

```
NameList : Array[0 : 100] OF STRING
```

The programmer wants to search the list using a binary search algorithm.

The programmer decides to write the search algorithm as a recursive function. The function, `Find`, takes three parameters:

- `Name`, the string to be searched for
- `Start`, the index of the first item in the list to be searched
- `Finish`, the index of the last item in the list to be searched

The function will return the position of the name in the list, or –1 if the name is not found.

Complete the **pseudocode** for the recursive function.

```
FUNCTION Find(BYVALUE Name : STRING, BYVALUE Start : INTEGER,
                                BYVALUE Finish : INTEGER) RETURNS INTEGER

   // base case

   IF .....

      THEN

         RETURN -1

      ELSE

         Middle ← .....

         IF .....

            THEN

               RETURN .....

            ELSE    // general case

               IF SearchItem > .....

                  THEN

                     .....

                  ELSE

                     .....

               ENDIF

         ENDIF

   ENDIF

ENDFUNCTION
```

[7]

QUESTION 9.

2 An ordered binary tree Abstract Data Type (ADT) has these associated operation

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

The binary tree ADT is to be implemented as a linked list of nodes.

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

(a) A null pointer is shown as Ø.

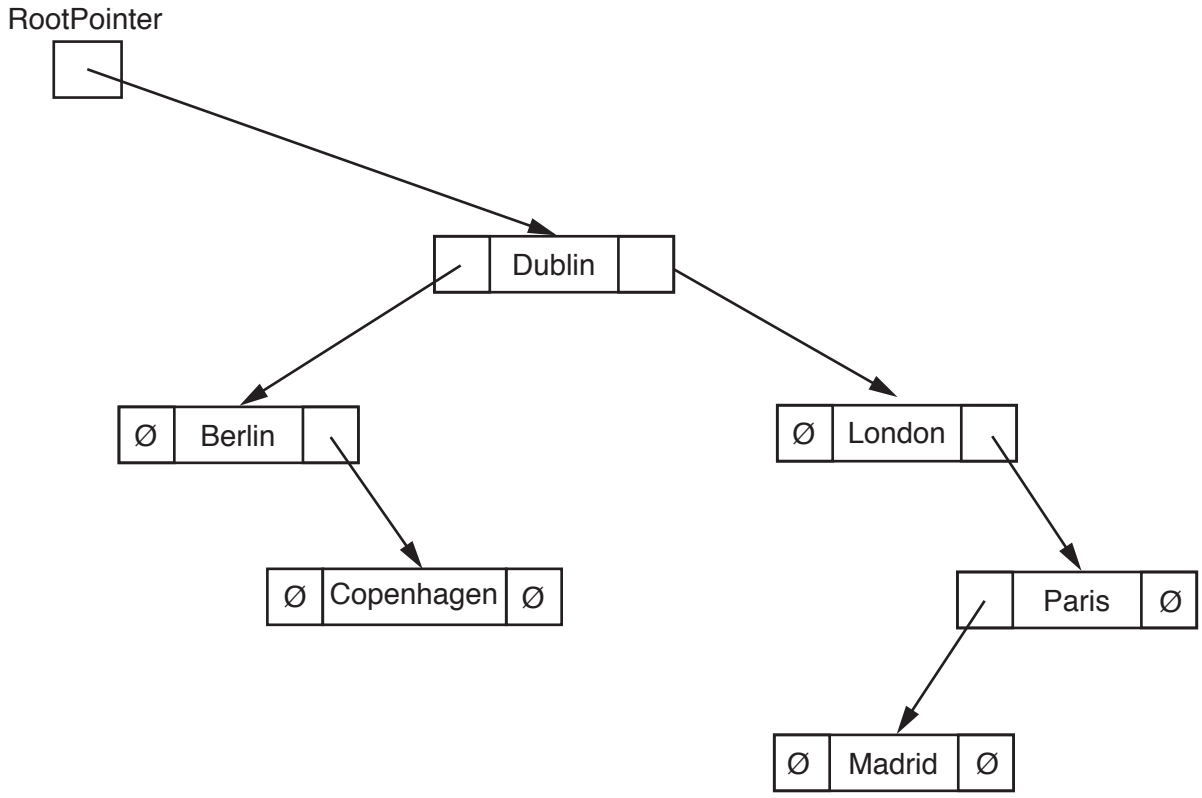
Explain the meaning of the term **null pointer**.

.....

.....[1]

(b) The following diagram shows an ordered binary tree after the following data have been added:

Dublin, London, Berlin, Paris, Madrid, Copenhagen



Another data item to be added is Athens.

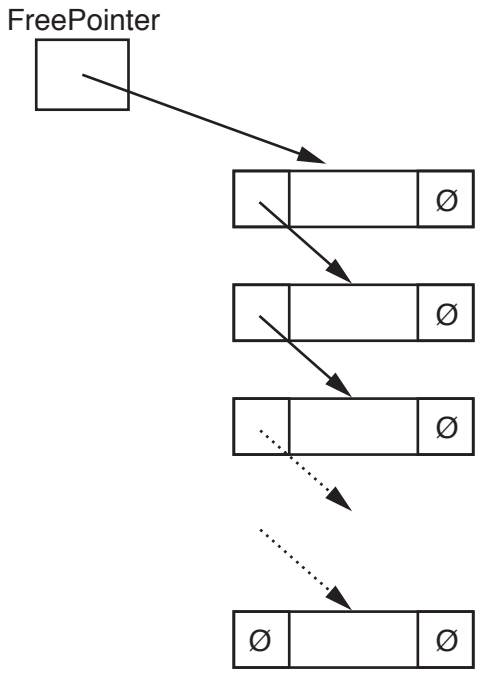
Make the required changes to the diagram when this data item is added. [2]

(c) A tree without any nodes is represented as:

RootPointer

Ø

Unused nodes are linked together as shown:



The following diagram shows an array of records that stores the tree shown in **part (b)**.

(i) Add the relevant pointer values to complete the diagram.

RootPointer

0

FreePointer

	LeftPointer	Tree data	RightPointer
[0]		Dublin	
[1]		London	
[2]		Berlin	
[3]		Paris	
[4]		Madrid	
[5]		Copenhagen	
[6]		Athens	
[7]			
[8]			
[9]			

[5]

(ii) Give an appropriate numerical value to represent the null pointer for th
your answer.

.....

.....

.....

.....[2]

(d) 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`”).

(i) Complete the pseudocode to create an empty tree.

```
TYPE Node
    .....
    .....
    .....
ENDTYPE
DECLARE Tree : ARRAY[0 : 9] .....
DECLARE FreePointer : INTEGER
DECLARE RootPointer : INTEGER

PROCEDURE CreateTree()
   DECLARE Index : INTEGER
   .....
   .....
   FOR Index ← 0 TO 9   // link nodes
      .....
      .....
   ENDFOR
   .....
ENDPROCEDURE
```

[7]

(ii) 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]

(e) The traverse tree operation outputs the data items in alphabetical order. Th
as a recursive solution.

Complete the pseudocode for the recursive procedure `TraverseTree`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
ENDPROCEDURE
```

[5]

QUESTION 10.

4 **(a)** The array `Numbers[0 : Max]` stores numbers. An insertion sort can be u… numbers into ascending order.

Complete the following **pseudocode** for the insertion sort algorithm.

```
FOR Pointer ← 1 TO (Max - 1)
    ItemToInsert ← .....
    CurrentItem ← .....
    WHILE (CurrentItem > 0) AND (Numbers[CurrentItem - 1] > ItemToInsert)
        Numbers[.....] ← Numbers[CurrentItem - 1]
        CurrentItem ← CurrentItem - 1
    ENDWHILE
    Numbers[CurrentItem] ← .....
ENDFOR
```

[4]

(b) Identify **two** features of the array `Numbers` that would have an impact on the performance of this insertion sort algorithm.

1 .....

2 .....

[2]

QUESTION 11.

3 Joseph is taking a toy apart. Each time he removes an item from the toy, he wr
the item at the bottom of a paper list. When he rebuilds the toy, he puts the items
working from the bottom of the list.

Joseph writes a computer program to create the list using a stack, `Parts`.

(a) Describe a stack structure.

.....

..... [1]

(b) The stack is represented as an array in the program, the first element in the array is `[0]`.

The current contents of the stack, `Parts`, and its pointer, `StackPointer` are shown.

StackPointer 5

	StackContents
0	"Screw 1"
1	"Screw 2"
2	"Back case"
3	"Screw 3"
4	"Engine outer"
5	
6	
7	

(i) Describe the purpose of the variable `StackPointer`.

.....

..... [1]

(ii) The procedure `POP()` removes an item from the stack. The procedure `PUSH(<identifier>)` adds an item to the stack.

The current contents of the stack, `Parts`, and its pointer, `StackPointer` are s

StackPointer 5

StackContents

0	"Screw 1"
1	"Screw 2"
2	"Back case"
3	"Screw 3"
4	"Engine outer"
5	
6	
7	

Use the table below to show the contents of the stack, `Parts`, and its pointer after the following code is run.

```
POP()
POP()
PUSH("Light 1")
PUSH("Light 2")
PUSH("Wheel 1")
POP()
POP()
```

StackPointer

StackContents

0	
1	
2	
3	
4	
5	
6	
7	

[2]

(c) A 1D array, `Parts`, is used to implement the stack. `Parts` is declared as:

```
DECLARE Parts : ARRAY[0 : 19] OF STRING
```

(i) The procedure `POP` outputs the last element that has been pushed onto the stac
replaces it with a `'*'`.

Complete the **pseudocode** for the procedure `POP`.

```
PROCEDURE POP

    IF ..... = .....

        THEN

            OUTPUT "The stack is empty"

        ELSE

            StackPointer ← .....

            OUTPUT .....

            Parts[StackPointer] ← .....

    ENDIF

ENDPROCEDURE
```

[5]

(ii) The procedure `PUSH()` puts the parameter onto the stack.

Complete the **pseudocode** for the procedure `PUSH()`.

```
PROCEDURE PUSH(BYVALUE Value : String)

    IF StackPointer > .....

        THEN

            OUTPUT "Stack full"

        ELSE

            ..... ← .....

            StackPointer ← .....

    ENDIF

ENDPROCEDURE
```

[4]

QUESTION 12.

6 A linked list abstract data type (ADT) is created. This is implemented as an array records are of type `ListElement`.

An example of a record of `ListElement` is shown in the following table.

Data Item	Value
`Country`	`"Scotland"`
`Pointer`	`1`

(a) (i) Use **pseudocode** to write a definition for the record type, `ListElement`.

.....

.....

.....

.....

..... [3]

(ii) Use **pseudocode** to write an array declaration to reserve space for only 15 nodes of type `ListElement` in an array, `CountryList`. The lower bound element is `1`.

..... [2]

(b) The program stores the position of the last node in the linked list in `LastNode`. The last node always has a `Pointer` value of `-1`. The position of the node at the head of the list is stored in `ListHead`.

After some processing, the array and variables are in the following state.

ListHead
1
LastNode
3

CountryList

	Country	**Pointer**
1	"Wales"	2
2	"Scotland"	4
3		-1
4	"England"	5
5	"Brazil"	6
6	"Canada"	7
7	"Mexico"	8
8	"Peru"	9
9	"China"	10
10		11
11		12
12		13
13		14
14		15
15		3

A **recursive** algorithm searches the list for a value, deletes that value, a
required pointers. When a node value is deleted, it is set to empty "" and the
to the end of the list.

A node value is deleted using the pseudocode statement

```
CALL DeleteNode("England", 1, 0)
```

Complete the following **pseudocode** to implement the `DeleteNode` procedure.

```
PROCEDURE DeleteNode(NodeValue: STRING, ThisPointer : INTEGER,
                                                PreviousPointer : INTEGER)

 IF CountryList[ThisPointer].Value = NodeValue

   THEN

     CountryList[ThisPointer].Value ← ""

     IF ListHead = ……

       THEN

         ListHead ← ……

       ELSE

         CountryList[PreviousPointer].Pointer ← CountryList[ThisPointer].Pointer

     ENDIF

     CountryList[LastNode].Pointer ← ……

     LastNode ← ThisPointer

     ……

   ELSE

     IF CountryList[ThisPointer].Pointer <> -1

       THEN

         CALL DeleteNode(NodeValue, ……,
                                                              ThisPointer)

       ELSE

         OUTPUT "DOES NOT EXIST"

     ENDIF

 ENDIF

ENDPROCEDURE
```

[5]

QUESTION 13.

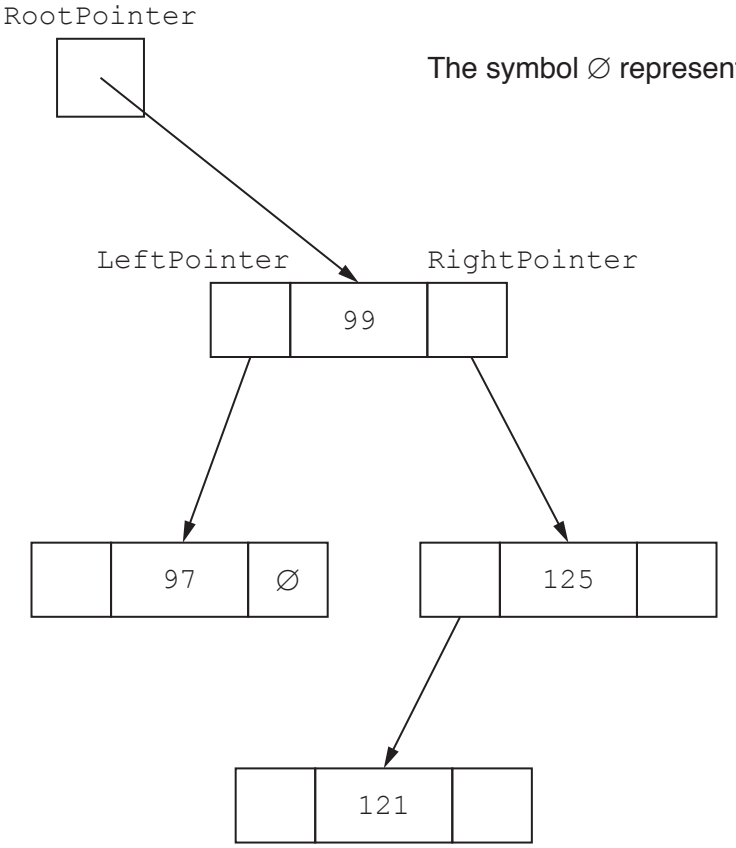
2 A computer games club wants to run a competition. The club needs a system to achieved in the competition.

A selection of score data is as follows:

99, 125, 121, 97, 109, 95, 135, 149

(a) A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.

(i) Complete the binary tree to show how the score data above will be organised.



[5]

(ii) The following diagram shows a 2D array that stores the nodes of the bi
list.

Add the correct pointer values to complete the diagram, using your answ
part (a)(i).

RootPointer

0

FreePointer

Index	LeftPointer	Data	RightPointer
0		99	
1		125	
2		121	
3		97	
4		109	
5		95	
6		135	
7		149	
8			

[6]

(b) The club also considers storing the data in the order in which it receives linked list in a 1D array of records.

The following pseudocode algorithm searches for an element in the linked list.

Complete the **six** missing sections in the algorithm.

```
FUNCTION FindElement(Item : INTEGER) RETURNS ..........

    .......... ← RootPointer

    WHILE CurrentPointer .......... NullPointer

        IF List[CurrentPointer].Data <> ..........

            THEN

                CurrentPointer ← List[..........].Pointer

            ELSE

                RETURN CurrentPointer

        ENDIF

    ENDWHILE

    CurrentPointer ← NullPointer

    .......... CurrentPointer

ENDFUNCTION
```

[6]

(c) The games club is looking at two programming paradigms: imperative and
programming paradigms.

Describe what is meant by the **imperative programming paradigm** and the **object-**
programming paradigm.

(i) Imperative .....

.....

.....

.....

.....

..... [3]

(ii) Object-oriented .....

.....

.....

.....

.....

..... [3]

(d) Players complete one game to place them into a category for the competition. wants to implement a program to place players into the correct category. The has decided to use object-oriented programming (OOP).

The highest score that can be achieved in the game is 150. Any score less than 50 w qualify for the competition. Players will be placed in a category based on their score.

The following diagram shows the design for the class `Player`. This includes the properties and methods.

Player

```
Score    : INTEGER // initialised to 0
Category : STRING  // "Beginner", "Intermediate",
                   // "Advanced" or "Not Qualified", initialised
                   // to "Not Qualified"
PlayerID : STRING  // initialised with the parameter InputPlayerID
```

```
Create()          // method to create and initialise an object using
                  // language-appropriate constructor
SetScore()        // checks that the Score parameter has a valid value
                  // if so, assigns it to Score
SetCategory()     // sets Category based on player's Score
SetPlayerID()     // allows a player to change their PlayerID
                  // validates the new PlayerID
GetScore()        // returns Score
GetCategory()     // returns Category
GetPlayerID()     // returns PlayerID
```

(i) The constructor receives the parameter `InputPlayerID` to create a
Other properties are initialised as instructed in the class diagram.

Write **program code** for the `Create()` constructor method.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(ii) Write **program code** for the following **three** get methods.

Programming language .....

`GetScore()`

Program code

.....

.....

.....

.....

`GetCategory()`

Program code

.....

.....

.....

.....

`GetPlayerID()`

Program code

.....

.....

.....

.....

[4]

(iii) The method `SetPlayerID()` asks the user to input the new player ID
value.

It checks that the length of the `PlayerID` is less than or equal to 15 charac
greater than or equal to 4 characters. If the input is valid, it sets this as the `Play`
otherwise it loops until the player inputs a valid `PlayerID`.

Use suitable input and output messages.

Write **program code** for `SetPlayerID()`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(iv) The method `SetScore()` checks that its `INTEGER` parameter `ScoreI` it is valid, it is then set as `Score`. A valid `ScoreInput` is greater than or e less than or equal to 150.

If the `ScoreInput` is valid, the method sets `Score` and returns `TRUE`.

If the `ScoreInput` is not valid, the method does not set `Score`, displays an erro message, and it returns `FALSE`.

Write **program code** for `SetScore(ScoreInput : INTEGER)`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(v) Write **program code** for the method `SetCategory()`. Use the properti
in the original class definition.

Players will be placed in one of the following categories.

Category	Criteria
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(vi) Joanne has played the first game to place her in a category for the com

The procedure `CreatePlayer()` performs the following tasks.

- allows the player ID and score to be input with suitable prompts
- creates an instance of `Player` with the identifier `JoannePlayer`
- sets the score for the object
- sets the category for the object
- outputs the category for the object

Write **program code** for the `CreatePlayer()` procedure.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(e) The programmer wants to test that the correct category is set for a player's s

As stated in **part (d)(v)**, players will be placed in one of the following categories.

Category	Criteria
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Complete the table to provide test data for each category.

Category	Type of test data	Example test data
Beginner	Normal	
	Abnormal	
	Boundary	
Intermediate	Normal	
	Abnormal	
	Boundary	
Advanced	Normal	
	Abnormal	
	Boundary	

[3]

(f) In **part (b)**, the club stored scores in a 1D array. This allows the club to sort t

The following is a sorting algorithm in pseudocode.

```
NumberOfScores ← 5
FOR Item ← 1 TO NumberOfScores - 1
   InsertScore ← ArrayData[Item]
   Index ← Item - 1
   WHILE (ArrayData[Index] > InsertScore) AND (Index >= 0)
      ArrayData[Index + 1] ← ArrayData[Index]
      Index ← Index - 1
   ENDWHILE
   ArrayData[Index + 1] ← InsertScore
ENDFOR
```

(i) Give the name of this algorithm.

..... [1]

(ii) State the name of **one** other sorting algorithm.

..... [1]

(iii) Complete a dry run of the algorithm using the following trace table.

Item	NumberOfScores	InsertScore	Index	ArrayData				
				0	1	2	3	
				99	125	121	109	115

[7]

QUESTION 14.

4 (a) A program has sorted some data in the array, `List`, in ascending order.

The following binary search algorithm is used to search for a value in the array.

```
ValueFound ← FALSE
UpperBound ← LengthOfList - 1
LowerBound ← 0
NotInList ← FALSE

WHILE ValueFound = FALSE AND NotInList = FALSE
   MidPoint ← ROUND((LowerBound + UpperBound) / 2)

   IF List[LowerBound] = SearchValue
      THEN
         ValueFound ← TRUE
      ELSE
         IF List[MidPoint] < SearchValue
            THEN
               UpperBound ← MidPoint + 1
            ELSE
               UpperBound ← MidPoint - 1
         ENDIF
         IF LowerBound > MidPoint
            THEN
               NotInList ← TRUE
         ENDIF
   ENDIF
ENDWHILE

IF ValueFound = FALSE
   THEN
      OUTPUT "The value is in the list"
   ELSE
      OUTPUT "The value is not found in the list"
ENDIF
```

Note:

The pseudocode function

```
ROUND(Reall : REAL) RETURNS INTEGER
```

rounds a number to the nearest integer value.

For example: `ROUND(4.5)` returns `5` and `ROUND(4.4)` returns `4`

(i) There are four errors in the algorithm.

Write the line of code where an error is present **and** write the correction in **ps**

Error 1 .....

Correction .....

Error 2 .....

Correction .....

Error 3 .....

Correction .....

Error 4 .....

Correction .....

[4]

(ii) A binary search is one algorithm that can be used to search an array.

Identify another searching algorithm.

..... [1]

(b) The following is an example of a sorting algorithm. It sorts the data in the ar

```
01  TempValue ← ""
02  REPEAT
03     Sorted ← TRUE
04     FOR Count ← 0 TO 4
05        IF ArrayData[Count] > ArrayData[Count + 1]
06           THEN
07              TempValue ← ArrayData[Count + 1]
08              ArrayData[Count + 1] ← ArrayData[Count]
09              ArrayData[Count] ← TempValue
10              Sorted ← FALSE
11        ENDIF
12     ENDFOR
13  UNTIL Sorted = TRUE
```

(i) Complete the trace table for the algorithm given in **part (b)**, for the `ArrayData` values given in the table.

Count	**TempValue**	**Sorted**	**ArrayData**					
			0	**1**	**2**	**3**	**4**	**5**
			5	20	12	25	32	29

[4]

(ii) Rewrite lines 4 to 12 of the algorithm in **part (b)** using a `WHILE` loop i
loop.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

(iii) Identify the algorithm shown in **part (b)**.

..... [1]

(iv) Identify another sorting algorithm.

..... [1]

QUESTION 15.

1 Each student at CIE University needs a printing account to print documents computers.

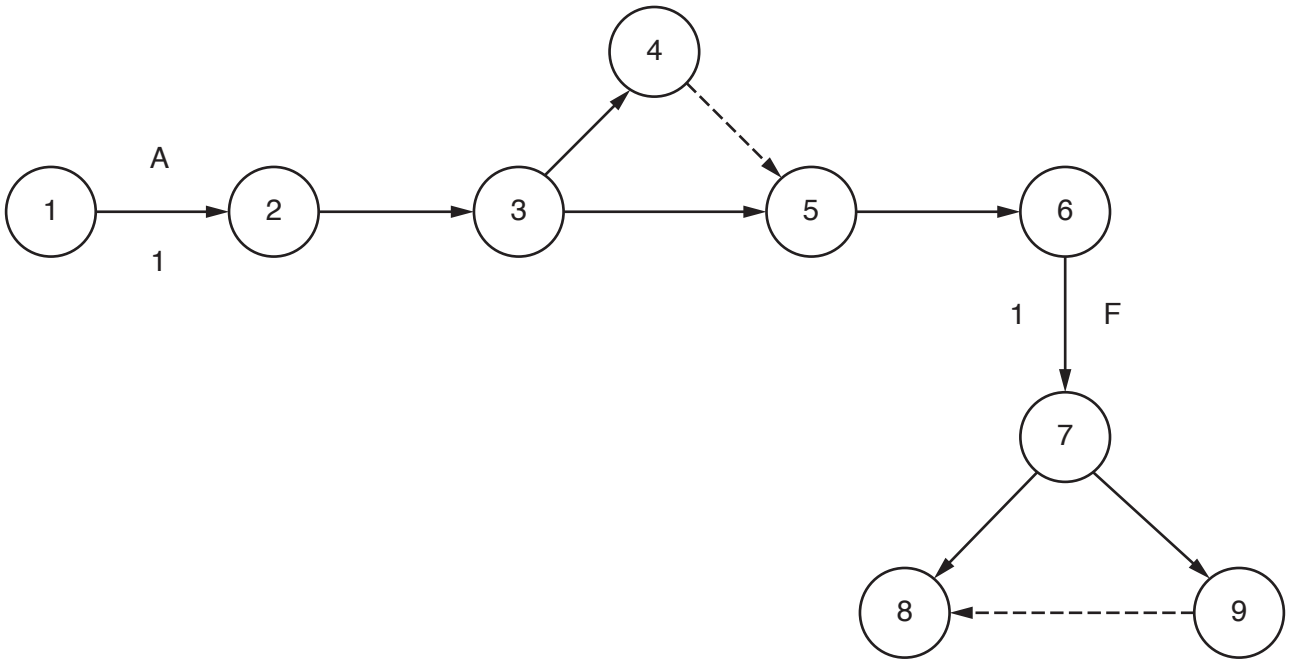
The university is developing software to manage each student's printing account and the process.

(a) Developing the software will include the following activities.

Activity	Description	Time in weeks	Predecessor
A	Identify requirements	1	-
B	Produce design	3	A
C	Write code	10	B
D	Test modules	7	B
E	Final system black-box testing	3	C, D
F	Install software	1	E
G	Acceptance testing	2	F
H	Create user documentation	2	F

(i) Add the correct activities and times to the following Program Evaluation Review Technique (PERT) chart for the software development.

Two activities and times have been done for you.



[6]

(ii) State what is meant by the **critical path** in a PERT chart.

.....

.....

(iii) Identify **and** describe a project planning technique, other than a PERT chart.

.....

.....

.....

..... [2]

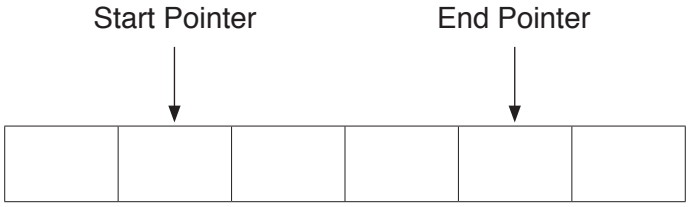
(b) When a student prints a document, a print job is created. The print job is sent to a print server.

The print server uses a queue to hold each print job waiting to be printed.

(i) The queue is circular and has six spaces to hold jobs.

The queue currently holds four jobs waiting to be printed. The jobs have arrived in the order A, B, D, C.

Complete the diagram to show the current contents of the queue.



[1]

(ii) Print jobs A and B are now complete. Four more print jobs have arrived in the order E, F, G, H.

Complete the diagram to show the current contents and pointers for the queue.

[3]

(iii) State what would happen if another print job is added to the queue in the status in **part (b)(ii)**.

.....

..... [1]

(iv) The queue is stored as an array, `Queue`, with six elements. The fol
removes a print job from the queue and returns it.

Complete the following **pseudocode** for the function `Remove`.

```
FUNCTION Remove RETURNS STRING
   DECLARE PrintJob : STRING
   IF ..... = EndPointer
      THEN
         RETURN "Empty"
      ELSE
         PrintJob ← Queue[.....]
         IF StartPointer = .....
            THEN
               StartPointer ← .....
            ELSE
               StartPointer ← StartPointer + 1
         ENDIF
         RETURN PrintJob
   ENDIF
ENDFUNCTION
```

[4]

(v) Explain why the circular queue could not be implemented as a stack.

.....

.....

.....

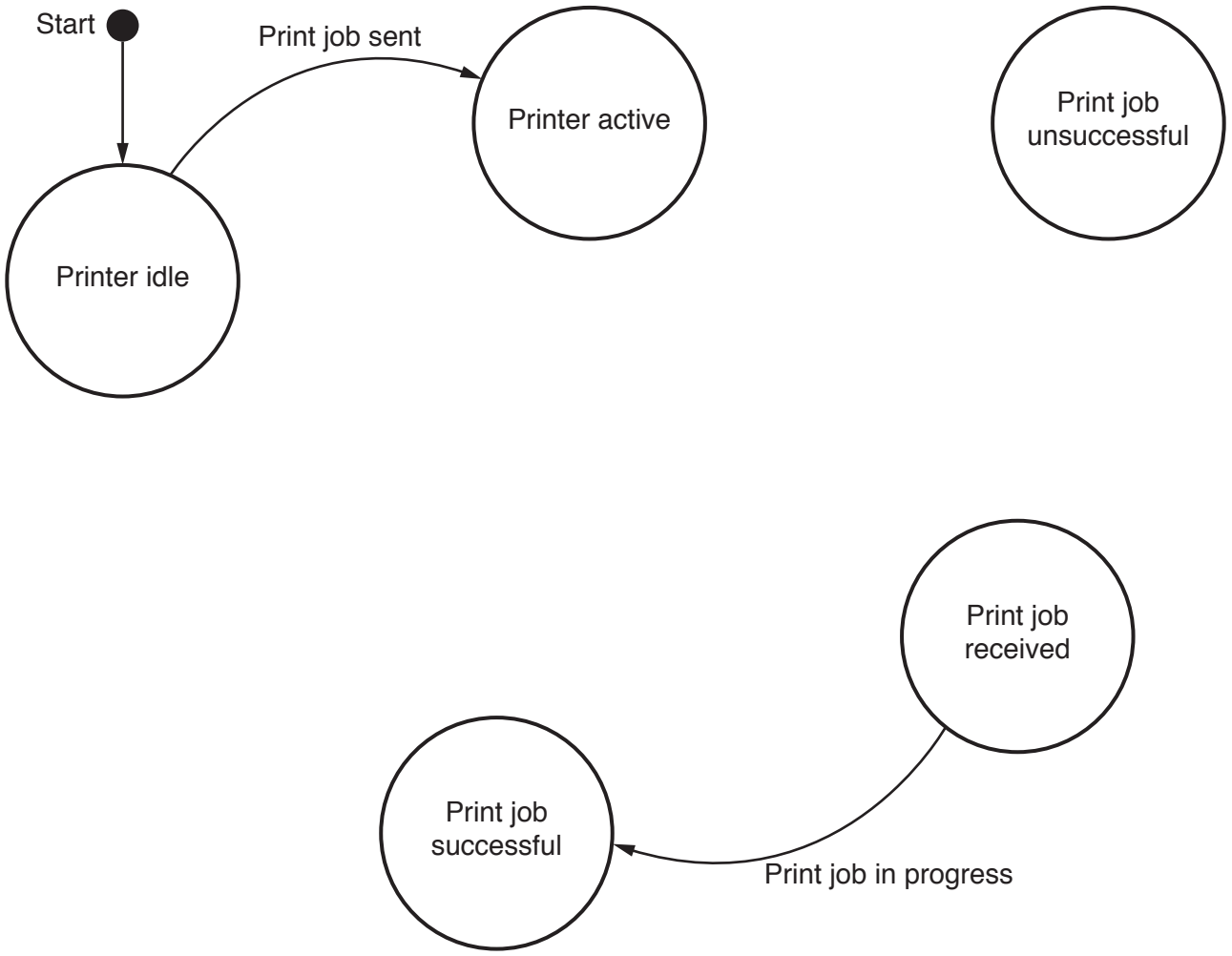
..... [2]

(c) The university wants to analyse how a printer and a print server deal with the

The following table shows the transitions from one state to another for the process

Current state	Event	Next state
Printer idle	Print job sent	Printer active
Printer active	Print job added to queue	Print job received
Print job received	Print job in progress	Print job successful
Print job received	Print job in progress	Print job unsuccessful
Print job successful	Check print queue	Printer active
Print job unsuccessful	Error message displayed	Printer active
Printer active	Timeout	Printer idle

Complete the state-transition diagram for the table.



[5]

(d) The university wants to assess troubleshooting issues with a printer. It wants ... table to do this.

The troubleshooting actions are:

- check the connection from computer to printer, if the error light is flashing **and** ... document has not been printed
- check the ink status, if the quality is poor
- check whether there is a paper jam, if the error light is flashing **and** the document has not been printed
- check the paper size selected, if the paper size is incorrect.

(i) Describe the purpose of a decision table.

.....

.....

.....

..... [2]

(ii) Complete the rules for the actions in the following decision table.

		Rules							
Conditions	Document printed but the quality is poor	Y	Y	Y	Y	N	N	N	N
	Error light is flashing on printer	Y	Y	N	N	Y	Y	N	N
	Document printed but paper size is incorrect	Y	N	Y	N	Y	N	Y	N
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[4]

(iii) Simplify your solution by removing redundancies.

		Rules							
Conditions	Document printed but the quality is poor								
	Error light is flashing on printer								
	Document printed but paper size is incorrect								
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[5]

(e) There are 1000 students at the university. They will each require a printing a

Students need to buy printing credits that will be added to their account. Each p uses one printing credit.

The university needs software to keep track of the number of printing credits each student in their account. The university has decided to implement the software using object-oriente programming (OOP).

The following diagram shows the design for the class `PrintAccount`. This includes the attributes and methods.

PrintAccount
FirstName : STRING // parameter sent to Constructor()
LastName : STRING // parameter sent to Constructor()
PrintID : STRING // parameter sent to Constructor()
Credits : INTEGER // initialised to 50

Constructor() // instantiates an object of the PrintAccount class,
// and assigns initial values to the attributes

GetName() // returns FirstName and LastName concatenated
// with a space between them

GetPrintID() // returns PrintID
SetFirstName() // sets the FirstName for a student
SetLastName() // sets the LastName for a student
SetPrintID() // sets the PrintID for a student
AddCredits() // increases the number of credits for a student
RemoveCredits() // removes credits from a student account

(i) Write **program code** for the `Constructor()` method.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(ii) Write **program code** for the `SetFirstName()` method.

Programming language .....

Program code

.....

.....

.....

.....

..... [2]

(iii) Write **program code** for the `GetName()` method.

Programming language .....

Program code

.....

.....

.....

.....

.....

..... [2]

(iv) The method `AddCredits()` calculates the number of printing credits and adds the printing credits to the student's account.

- Credits cost $1 for 25 credits.
- If a student buys $20 or more of credits in a single payment, they receive an extra 50 credits.
- If a student buys between $10 and $19 (inclusive) of credits in a single payment, they receive an extra 25 credits.

Payment from a student is stored in the variable `MoneyInput`. This is passed as a parameter.

Write **program code** for `AddCredits()`. Use constants for the values that do not change.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(v) A global array, `StudentAccounts`, stores 1000 instances of `PrintAc`

Write **pseudocode** to declare the array `StudentAccounts`.

.....

..... [2]

(vi) The main program has a procedure, `CreateID()`, that:

- takes the first name and last name as parameters
- creates `PrintID` that is a concatenation of:
  - the first three letters of the first name in lower case
  - the first three letters of the last name in lower case
  - the character '1'
    for example, the name Bill Smith would produce `"bilsmi1"`
- checks if the `PrintID` created already exists in the global array `StudentAccounts`:
  - If `PrintID` does not exist, it creates an instance of `PrintAccount` in the next free index in `StudentAccounts`.
  - If `PrintID` does exist, the number is incremented until a unique ID is created, for example, `"bilsmi2"`. It then creates an instance of `PrintAccount` in the next free index in `StudentAccounts`.

The global variable `NumberStudents` stores the number of print accounts that have currently been created.

Write **program code** for the procedure `CreateID()`. Do not write the procedure header.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

QUESTION 16.

2 The number of cars that cross a bridge is recorded each hour. This number is placed in a circular queue before being processed.

(a) The queue is stored as an array, `NumberQueue`, with eight elements. The function `AddToQueue` adds a number to the queue. `EndPointer` and `StartPointer` are global variables.

Complete the following **pseudocode** algorithm for the function `AddToQueue`.

```
FUNCTION AddToQueue(Number : INTEGER) RETURNS BOOLEAN
   DECLARE TempPointer : INTEGER
   CONSTANT FirstIndex = 0
   CONSTANT LastIndex = .....
   TempPointer ← EndPointer + 1
   IF ..... > LastIndex
       THEN
          TempPointer ← .....
   ENDIF
   IF TempPointer = StartPointer
       THEN
          RETURN .....
       ELSE
          EndPointer ← TempPointer
          NumberQueue[EndPointer] ← .....
          RETURN TRUE
   ENDIF
ENDFUNCTION
```

[5]

(b) Describe how a number is removed from the circular queue to be processed.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) A queue is one example of an Abstract Data Type (ADT).

Identify **three other** Abstract Data Types.

1 .....

2 .....

3 .....

[3]