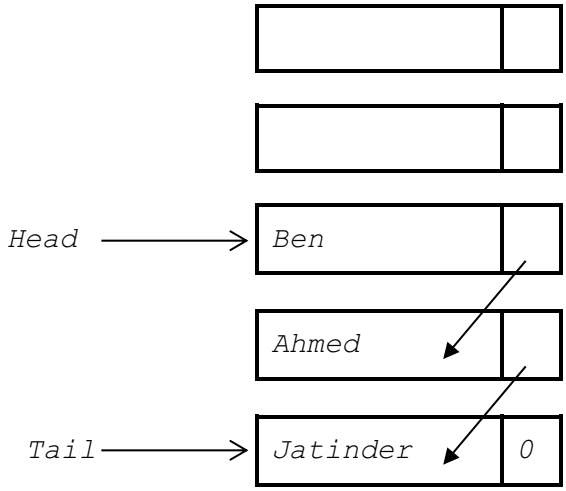
QUESTION 1.

6 (a)



1 mark for Head and Tail pointers
1 mark for 3 correct items – linked as shown
1 mark for correct order with null pointer in last nod **[3]**

(b) (i)

Queue

HeadPointer		Name	Po
0	[1]		
	[2]		
TailPointer	[3]		
0	[4]		5
	[5]		6
FreePointer	[6]		7
1	[7]		8
	[8]		9
	[9]		10
	[10]		0

Mark as follows:

HeadPointer =0 & TailPointer = 0
FreePointer assigned a value
Pointers[1] to [9] links the nodes together
Pointer[10] = 'Null' **[4]**

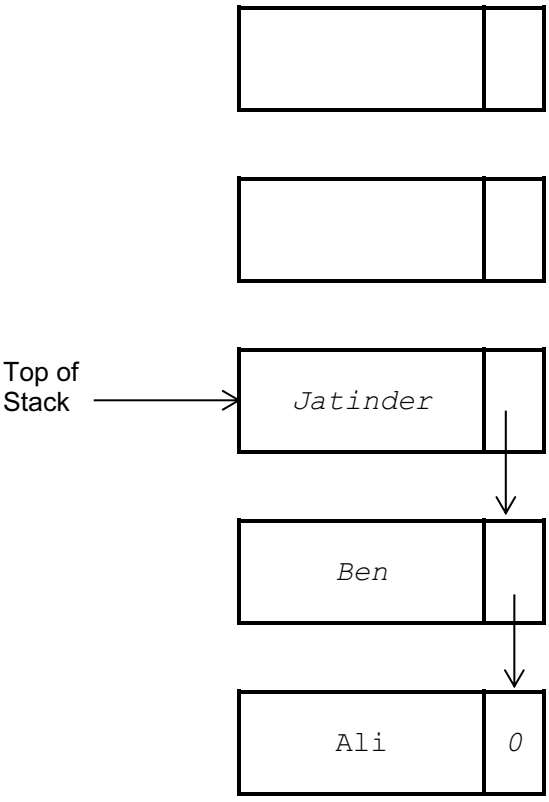
(ii)

```
PROCEDURE RemoveName()
    // Report error if Queue is empty
    IF HeadPointer = 0
        THEN
            Error
        ELSE
            OUTPUT Queue[HeadPointer].Name
            // current node is head of queue
            CurrentPointer ← HeadPointer
            // update head pointer
            HeadPointer ← Queue[CurrentPointer].Pointer
            //if only one element in queue,then update tail pointer
            IF HeadPointer = 0
                THEN
                    TailPointer ← 0
            ENDIF
            // link released node to free list
            Queue[CurrentPointer].Pointer ← FreePointer
            FreePointer ← CurrentPointer
    ENDIF
ENDPROCEDURE
```

[max 6]

QUESTION 2.

5 (a)



1 mark for Top of Stack pointer
1 mark for 3 correct items
1 mark for correct order with null pointer in last node **[3]**

(b) (i)

Stack

TopOfStackPointer

0

FreePointer

1

	Name	Pointer
[1]		2
[2]		3
[3]		4
[4]		5
[5]		6
[6]		7
[7]		8
[8]		9
[9]		10
[10]		0

Mark as follows:
TopOfStackPointer
FreePointer
Pointers[1] to [9]
Pointer[10]

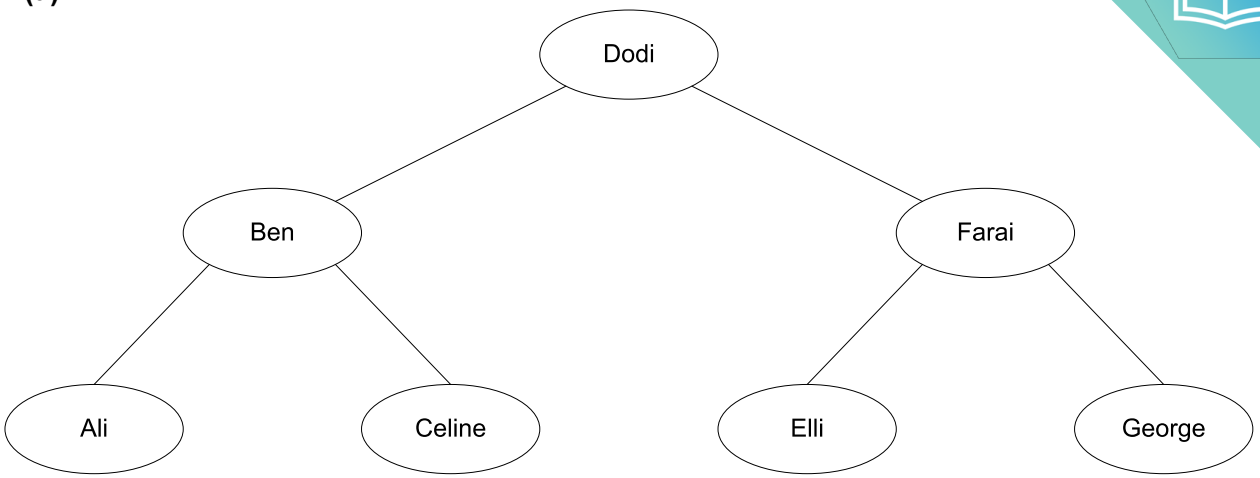
[4]

(ii)

```
PROCEDURE Pop()
    // Report error if Stack is empty
    IF TopOfStackPointer = 0
      THEN
        Error
      ELSE
        OUTPUT Stack[TopOfStackPointer].Name
        // take a copy of the current top of stack pointer
        TempPointer ← TopOfStackPointer
        // update the top of stack pointer
        TopOfStackPointer ← Stack[TempPointer].Pointer
        // link released node to free list
        Stack[TempPointer].Pointer ← FreePointer
        FreePointer ← TempPointer
    ENDIF
ENDPROCEDURE
```

QUESTION 3.

4 (a)



[4]

(b)

RootPointer: 1

FreePointer: 8

Tree

	Name	LeftPointer	RightPointer
[1]	Dodi	5	2
[2]	Farai	3	4
[3]	Elli	0	0
[4]	George	0	0
[5]	Ben	7	6
[6]	Celine	0	0
[7]	Ali	0	0
[8]		9	0
[9]		10	0
[10]		0	0

[7]

(c) (i)

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

[5]

(ii) A procedure that calls itself // is defined in terms of itself
Line number: 04/09

[2]

QUESTION 4.

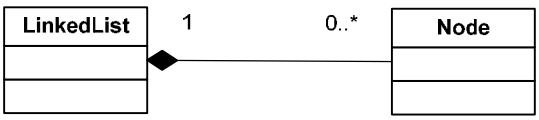
1 (a) (i)	`TYPE LinkedList` 1
	
	`(DECLARE) Surname : STRING` 1
	`(DECLARE) Ptr : INTEGER`
	
	`ENDTYPE` 1
	
	Accept:
	`LinkedList : RECORD` 1
	
	`Surname : STRING` 1
	`Ptr : INTEGER`
	
	`ENDRECORD` 1
	
	Accept:
	`TYPE LinkedList = RECORD` 1
	
	`Surname : STRING` 1
	`Ptr : INTEGER`
	
	`ENDTYPE / ENDRECORD` 1
	
	Accept:
	`STRUCTURE LinkedList` 1
	
	`(DECLARE) Surname : STRING` 1
	`(DECLARE) Ptr : INTEGER`
	
	`ENDSTRUCTURE` 1
	
	Accept `AS / OF` instead of `:`	

(ii)	`(DECLARE)` **`SurnameList[1:5000]`** `:` **`LinkedList`** Accept `AS / OF` instead of `:` Accept `()` instead of `[]` Accept without lower bound Index separator can be `, : ...`	**2**
(b) (i)	`Wu` Accept with quotes	**1**
(ii)	`6`	**1**
(c) (i)	`IsFound` + relevant description 1 `BOOLEAN` 1	**2**

Question	Answer

(ii)	Accept `()` instead of `[]`
	`01 Current ← StartPtr`
	`02 IF Current = 0`
	`03 THEN`
	`04 OUTPUT "Empty List"` (or similar message) (accept without quotes) Reject "Error"
	`05 ELSE`
	`06 IsFound ← FALSE`
	`07 INPUT ThisSurname`
	`08 REPEAT`
	`09 IF SurnameList[Current].Surname = ThisSurname`
	`10 THEN`
	`11 IsFound ← TRUE`
	`12 OUTPUT "Surname found at position ", Current`
	`13 ELSE`
	`14 // move to the next list item`
	`15 Current ← SurnameList[Current].Ptr`
	`16 ENDIF`
	`17 UNTIL IsFound = TRUE OR Current = 0`
	`18 IF IsFound = FALSE`
	`19 THEN`
	`20 OUTPUT "Not Found"`
	`21 ENDIF`
	`22 ENDIF`

QUESTION 5.

Question	Answer	Marks
4(a)(i)	containment/aggregation	**1**
4(a)(ii)	 1 mark for the two classes (in boxes) and connection with correct end point 1 mark for 0 ..* 0	**Max 2**

Question	Answer	

4(b)	**mark as follows:**
	• Class heading and ending
	• Constructor heading and ending
	• Parameters in constructor heading
	• Declaration of (private) attributes : Pointer, Data
	• Assignment of parameters to Pointer and Data
	
	Python Example
	`class Node:` 1
	`def __init__(self, D, P):` 1 + 1
	`self.__Data = D` 1
	`self.__Pointer = P` 1
	`return`
	
	Example Pascal
	`type` } `Node = class` 1
	`private` 1
	`Data : String;` } `Pointer : Integer;`
	`public`
	`constructor Create(D : string; P : integer);`
	`procedure SetPointer(P : Integer);`
	`procedure SetData(D : String);`
	`function GetData() : String;` } ignore
	`function GetPointer() : Integer;`
	`end;`
	`constructor Node.Create(D : string; P : integer);` 1+1
	`begin`
	`Data := D;` } `Pointer := P;` 1
	`end;`
	
	Example VB.NET
	`Class Node` 1
	`Private Data As String` } `Private Pointer As Integer` 1
	`Public Sub New(ByVal D As String, ByVal P As Integer)` 1+1
	`Data = D` } `Pointer = P` 1
	`End Sub`
	`End Class`	

4(c)(i)	A pointer that doesn't point to any data/node/address	**1**

Question	Answer	
4(c)(ii)	–1 (accept NULL) The array only goes from 0 to 7 // the value is not an array index	
4(c)(iii)	**mark as follows:** • Class and constructor heading and ending • Declare private attributes (HeadPointer, FreeListPointer, NodeArray) • Initialise HeadPointer to null • Initialise FreeListPointer to 0 • Looping 8 times … • Creating empty node in NodeArray • Use .SetPointer method to point each new node to next node • Set last node pointer to null pointer	**Max 7**

Python Example

```
class LinkedList:                                          1
   def __init__(self):                                    1
      self.__HeadPointer = - 1                             1
      self.__FreeListPointer = 0                           1
      self.__NodeArray = []
      for i in range(8):                                   1
         ThisNode = Node("", (i + 1))                      1
         self.__NodeArray.append(ThisNode)
      self.__NodeArray[7].SetPointer(- 1)                  1
```

Example Pascal

```
type
LinkedList = class                                         1
      private
         HeadPointer : Integer;
         FreeList : Integer;
         NodeArray : Array[0..7] of Node;
      public
         constructor Create();
         procedure FindInsertionPoint(NewData : string; var
                     PreviousPointer, NextPointer : integer);
         procedure AddToList(NewData : string);
         procedure OutputListToConsole();
   end;
constructor LinkedList.Create();                           1
   var i : integer;
   begin
      HeadPointer := -1;                                   1
      FreeList := 0;                                       1
      for i := 0 To 7 do                                   1
         NodeArray[i] := Node.Create('', (i + 1));         1
      NodeArray[7].SetPointer(-1);                         1
   end;
```

Question	Answer	

	Example VB.NET
	`Class LinkedList` 1
	`  Private HeadPointer As Integer`
	`  Private FreeList As Integer`
	`  Private NodeArray(7) As Node`
	
	`  Public Sub New()` 1
	`  HeadPointer = -1` 1
	`  FreeList = 0` 1
	`  For i = 0 To 7` 1
	`  NodeArray(i) = New Node("", (i + 1))` 1
	`  Next`
	`  NodeArray(7).SetPointer(-1)` 1
	`  End Sub`
	`End Class`	

4(c)(iv)	• Creating instance of LinkedList assigned to contacts
	
	Python Example
	`contacts = LinkedList()`
	
	Pascal Example
	`var contacts : LinkedList;`
	`  contacts := LinkedList.Create;`
	
	VB.NET Example
	`Dim contacts As New LinkedList`
		1

Question	Answer	

4(c)(v)	**mark as follows:**
	• Start with HeadPointer
	• Output node data
	• Loop until null pointer
	• Following pointer to next node
	• Use of getter (ie GetData/GetPointer)
	
	Python Example
	`def OutputListToConsole(self) :`
	`  Pointer = self.__HeadPointer` 1
	`  while Pointer != -1 :` 1
	`  print(self.__NodeArray[Pointer].GetData())` 1+1
	`  Pointer = self.__NodeArray[Pointer].GetPointer()` 1
	`  print()`
	`  return`
	
	Pascal Example
	`procedure LinkedList.OutputListToConsole();`
	`  var Pointer : integer;`
	`  begin`
	`  Pointer := HeadPointer;` 1
	`  while Pointer <> -1 do` 1
	`  begin`
	`  WriteLn(NodeArray[Pointer].GetData);` 1+1
	`  Pointer := NodeArray[Pointer].GetPointer;` 1
	`  end;`
	`  end;`
	
	VB.NET Example
	
	`Public Sub OutputListToConsole()`
	`  Dim Pointer As Integer`
	`  Pointer = HeadPointer` 1
	`  Do While Pointer <> -1` 1
	`  Console.WriteLine(NodeArray(Pointer).GetData)` 1+1
	`  Pointer = NodeArray(Pointer).GetPointer` 1
	`  Loop`
	`End Sub`	

Question	Answer	

4(c)(vi)	**mark as follows:**
	• Store free list pointer as NewNodePointer
	• Store new data item in free node
	• Adjust free pointer
	• F list is currently empty
	• Make the node the first node
	• Set pointer of this node to Null Pointer
	• Find insertion point
	• If previous pointer is Null pointer
	• Link this node to front of list
	• Link new node between Previous node and next node
	
	Python Example
	`def AddToList(self, NewData):`
	
	`  NewNodePointer = self.__FreeListPointer`
	
	`  self.__NodeArray[NewNodePointer].SetData(NewData)`
	
	`  self.__FreeListPointer = self.__NodeArray[self.__FreeListPointer].GetPointer()`
	
	`  if self.__HeadPointer == -1:`
	
	`  self.__HeadPointer = NewNodePointer`
	`  self.__NodeArray[NewNodePointer ].SetPointer(-1)`
	`  else:`
	`  PreviousPointer, NextPointer = self.FindInsertionPoint(NewData)`
	`  if PreviousPointer == -1 :`
	
	`  self.__NodeArray[NewNodePointer ].SetPointer (self.__HeadPointer)`
	`  self.__HeadPointer = NewNodePointer`
	`  else:`
	
	`  self.__NodeArray[NewNodePointer ].SetPointer(NextPointer)`
	`  self.__NodeArray[PreviousPointer].SetPointer(NewNodePointer)`	

Question	Answer

	Pascal Example
	`procedure LinkedList.AddToList(NewData : string);`
	`var NewNodePointer , PreviousPointer,`
	`NextPointer : integer;`
	`begin`
	`// make a copy of free list pointer`
	`NewNodePointer := FreeListPointer;`
	`// store new data item in free node`
	`NodeArray[NewNodePointer].SetData(NewData);`
	`// adjust free pointer`
	`FreeListPointer :=`
	`NodeArray[FreeListPointer].GetPointer;`
	`// if list is currently empty`
	`if HeadPointer = -1`
	`then`
	`// make the node the first node`
	`begin`
	`HeadPointer := NewNodePointer;`
	`// set pointer to Null pointer`
	`NodeArray[NewNodePointer].SetPointer(-1);`
	`end`
	`else`
	`// find insertion point`
	`begin`
	`FindInsertionPoint(NewData, PreviousPointer,`
	`NextPointer);`
	`// if previous pointer is Null pointer`
	`if PreviousPointer = -1`
	`then`
	`// link node to front of list`
	`begin`
	`NodeArray[NewNodePointer]`
	`.SetPointer(HeadPointer);`
	`HeadPointer := NewNodePointer ;`
	`end`
	`else`
	`// link new node between`
	`Previous node and next node`
	`begin`
	`NodeArray[NewNodePointer ]`
	`.SetPointer(NextPointer);`
	`NodeArray[PreviousPointer]`
	`.SetPointer(NewNodePointer);`
	`end;`
	`end;`
	`end;`

Question	Answer

	VB.NET Example
	`Public Sub AddToList(ByVal NewData As String)`
	`  Dim NewNodePointer, PreviousPointer, NextPointer As Integer`
	`  ' make copy of free list pointer`
	`  NewNodePointer= FreeListPointer`
	`  ' store new data item in free node`
	`  NodeArray(NewNodePointer).SetData(NewData)`
	`  ' adjust free pointer`
	`  FreeListPointer = NodeArray(FreeListPointer).GetPointer`
	`  ' if list iscurrently empty`
	`  If HeadPointer = -1 Then`
	`  ' make the node the first node`
	`  HeadPointer = NewNodePointer`
	`  ' set pointer to Null pointer`
	`  NodeArray(NewNodePointer).SetPointer(-1)`
	`  Else`
	`  ' find insertion point`
	`  FindInsertionPoint(NewData, PreviousPointer,`
	`NextPointer)`
	`  ' if previous pointer is Null pointer`
	`  If PreviousPointer = -1 Then`
	`  ' link to front of list`
	`  NodeArray(NewNodePointer).SetPointer(HeadPointer)`
	`  HeadPointer = NewNodePointer`
	`  Else`
	`  ' link new node between Previous node and next node`
	`  NodeArray(NewNodePointer).SetPointer(NextPointer)`
	`  NodeArray(PreviousPointer).SetPointer(NewNodePointer)`
	`  End If`
	`  End If`
	`End Sub`

Question	Answer

Pseudocode for reference:

```
PROCEDURE AddToList(NewData)
   // remember value of free list pointer
   NewNodePointer ← FreeListPointer
   // add new data item to free node pointed to by free list
   NodeArray[NewNodePointer].Data ← NewData
   // adjust free pointer to point to next free node
   FreeListPointer  ← NodeArray[FreeList].Pointer
   // is list currently empty?
   IF HeadPointer = NullPointer
      THEN
         // make the node the first node
         HeadPointer ← NewnodePointer
         // set pointer of new node to Null pointer
         NodeArray[NewNodePointer].Pointer ← NullPointer
      ELSE
         // find insertion point
         CALL FindInsertionPoint(NewData, PreviousPPointer, NextPointer)
         // if previous pointer is Null pointer
         IF PreviousPointer = NullPointer
            THEN
               // link new node to front of list
               NodeArray[NewNodePointer].Pointer ← HeadPointer
               HeadPointer ← NewNodePointer
         ELSE
            // link new node between previous node and next node
            NodeArray[NewNodePointer].Pointer ← NextPOinter
            NodeArray[PreviousPointer].Pointer ← NewNodePointer
         END IF
   ENDIF
END PROCEDURE
```

QUESTION 6.

Question	Answer	Marks
2(a)	• A pointer that doesn't point to another node/other data/address // indicates the end of the branch	1
2(b)	one mark per bullet • node with 'Athens' linked to left pointer of Berlin (ignore null pointer) • null pointers in left and right pointers of Athens	2
2(c)(i)	RootPointer: 0 FreePointer: 7 **1 mark** [0] 2 | Dublin | 1 [1] -1/∅ | London | 3 [2] 6 | Berlin | 5 [3] 4 | Paris | -1/∅ [4] -1/∅ | Madrid | -1/∅ [5] -1/∅ | Copenhagen | -1/∅ [6] -1/∅ | Athens | -1/∅ [7] 8 | | -1/∅ [8] 9 | | -1/∅ [9] -1/∅ | | -1/∅	5
2(c)(ii)	• –1 • It is not the number for any node.	2

Question	Answer	Marks
2(d)(i)	TYPE Node **LeftPointer : INTEGER** **RightPointer : INTEGER** 1 **Data : STRING** ENDTYPE DECLARE Tree : ARRAY[0 : 9] **OF Node** 1 DECLARE FreePointer : INTEGER DECLARE RootPointer : INTEGER PROCEDURE CreateTree() DECLARE Index : INTEGER **RootPointer ← -1** 1 **FreePointer ← 0** 1 FOR Index ← 0 TO 9 // link nodes **Tree[Index].LeftPointer ← Index + 1** 1 **Tree[Index].RightPointer ← -1** 1 ENDFOR **Tree[9].LeftPointer ← -1** 1 ENDPROCEDURE	**7**

Question	Answer	Marks

2(d)(ii)	PROCEDURE AddToTree(ByVal NewDataItem : STRING)
	// if no free node report an error
	IF FreePointer **= -1** 1
	THEN
	ERROR("No free space left")
	ELSE // add new data item to first node in the free list
	NewNodePointer ← FreePointer
	Tree[NewNodePointer].Data ← NewDataItem 1
	// adjust free pointer
	FreePointer ← **Tree[FreePointer].LeftPointer** 1
	// clear left pointer
	Tree[NewNodePointer].LeftPointer ← **-1** 1
	// is tree currently empty ?
	IF **RootPointer = -1** 1
	THEN // make new node the root node
	RootPointer ← NewNodePointer 1
	ELSE // find position where new node is to be added
	Index ← RootPointer
	CALL FindInsertionPoint(NewDataItem, Index, Direction)
		8

Question	Answer	Marks
	IF Direction = "Left" THEN // add new node on left **Tree[Index].LeftPointer ← NewNodePointer** 1 ELSE // add new node on right **Tree[Index].RightPointer ← NewNodePointer** **1** ENDIF ENDIF ENDIF ENDPROCEDURE	
2(e)	**1 mark per bullet** • test for base case (null/-1) • recursive call for left pointer • output data • recursive call for right pointer • order, visit left, output, visit right IF Pointer <> NULL 1 THEN TraverseTree(Tree[Pointer].LeftPointer) 1 OUTPUT Tree[Pointer].Data 1 + 1 TraverseTree(Tree[Pointer].RightPointer) 1 ENDIF ENDPROCEDURE	**5**

QUESTION 7.

Question	Answer	Marks
3(a)	1 mark per clause • `person(mimi).` • `food(lettuce).` • `likes(mimi, chocolate).` • `dislikes(mimi, sushi).` • `dislikes(mimi, lettuce).`	**5**
3(b)	1 mark per answer `chocolate, pizza`	**2**

3(c)	1 mark per bullet
	• `might_like(`**`B,A`**`)`
	• `Person(B)`
	• `Food(A)`
	• `AND`
	• `AND NOT`
	• `Dislikes` predicate
	
	For example:
	
	`might_like(B, A).`
	
	`IF person(B) AND food(A)`
	
	`AND NOT(dislikes(B, A)).`
		6

QUESTION 8.

Question	Answer	Marks
3(a)	1 mark per clause • `room(corridor).` • `furniture(table).` • `furniture(lamp).` • `located(table, corridor).` • `located(lamp, corridor).`	**5**
3(b)	• `master_bedroom` • `spare_bedroom`	**2**
3(c)(i)	1 mark per bullet to max 2 • The first clause only says the nursery is next to the master bedroom • … but not that the master bedroom is next to the nursery • The second clause only says the master bedroom is next to the nursery • … but not that the nursery is next to the master bedroom • Goal to find rooms adjacent to master bedroom would not return nursery • … Example. `FindNextTo(X, master_bedroom)` • It is a two-way relationship	**2**
3(c)(ii)	1 mark per bullet • `room(main_bathroom).` • `nextTo(corridor, main_bathroom).` • `nextTo(main_bathroom, corridor).`	**3**

Question	Answer	Marks

3(d)	1 mark per bullet
	• `canBeMovedTo(`**`B,A`**`)`
	• `Furniture(B)`
	• `Room(A)`
	• `AND / ,`
	• `AND NOT / , NOT`
	• `Located(B,A)`
	
	Example:
	
	`canBeMovedTo(B,A)`
	
	`IF furniture(B) AND room(A)`
	
	`AND NOT(located(B,A)).`
		6

QUESTION 9.

Question	Answer	Marks

2(a)	1 mark per bullet point to max 4:
	
	• declaration of type Book
	• Title, Author and ISBN as String
	• Fiction as Boolean
	• LastRead as Date
	
	For example:
	
	`TYPE Book`
	`  DECLARE Title : String`
	`  DECLARE Author : String`
	`  DECLARE ISBN : String`
	`  DECLARE Fiction : Boolean`
	`  DECLARE LastRead : Date`
	`ENDTYPE`
		4

Question	Answer	Marks

2(b)	1 mark per bullet point to max 4:
	
	• **Function** header
	• … taking ISBN as parameter
	• Converting ISBN to integer
	• Calculating Hash (ISBN mod 2000 + 1)
	• Returning the calculated Hash
	
	Examples:
	
	Python:
	
	`def Hash(ISBN):`
	`  ISBNint = int(ISBN)`
	`  Hash = (ISBNint % 2000) + 1`
	
	VB.NET:
	
	`Function Hash (ISBN As String) As Integer`
	`  ISBNint = convert.toInt32(ISBN)`
	`  Hash = (ISBNint MOD 2000) + 1`
	`End Function`
	
	Pascal:
	
	`function Hash(ISBN : String) : Integer`
	`  begin`
	`  ISBNint = StrToInt(ISBN)`
	`  Hash = (ISBNint MOD 2000) + 1`
	`  end;`
		4

Question	Answer	Marks
2(c)	1 mark per bullet point to max 8: • Procedure FindBook declaration **and** prompt **and** input ISBN • Validate data input has 13 characters • … and are all numeric • ..loop until valid • Call Hash() with input data **and** store return data • Open MyBooks.dat for reading as random file **and** close • Finding the record using return value Hash() • Get the data for the record • …store in variable of type Book • …output all the data for the record	**8**

Question	Answer	Marks

2(c)	Example:
	
	`PROCEDURE FindBook()`
	`  DECLARE BookInfo : Book`
	
	`  REPEAT`
	`  ISBN ← input("Enter the ISBN number")`
	`  Valid ← TRUE`
	`  Size ← LENGTH(ISBN)`
	`  IF size <> 13`
	`  THEN`
	`  Valid ← FALSE`
	`  ELSE`
	`  FOR i ← 1 to 13`
	`  IF NOT( MID(ISBN,i,1) >= '0' AND MID(ISBN,i,1)<= '9' )`
	`  THEN`
	`  Valid ← FALSE`
	`  ENDIF`
	`  ENDFOR`
	`  ENDIF`
	`  UNTIL Valid`
	
	`  Filename ← "myBooks.dat"`
	`  OPENFILE Filename FOR RANDOM`
	`  RecordLocation ← Hash(ISBN)`
	`  SEEK FileName, RecordLocation`
	`  GETRECORD Filename, BookInfo`
	`  CLOSEFILE Filename`
	`  OUTPUT (BookInfo.Title & " " & BookInfo.Author & " " &`
	`  BookInfo.ISBN & " " & BookInfo.Fiction & " " &`
	`  BookInfo.LastRead)`
	`ENDPROCEDURE`	

QUESTION 10.

Question	Answer	Marks
5(c)(ii)	1 mark per bullet point to max 2: Example: • If there is any delay to a task which is part of the critical path • … then the overall project will be delayed • Gives the earliest possible completion time • … this allows you to organise/allocate resources (efficiently) • Frequently recalculate the critical path … • … to see if there are any delays/new critical path has arisen • Can identify where there is slack in activities • … so they can start later without affecting the critical path	**2**

Question	Answer	Marks

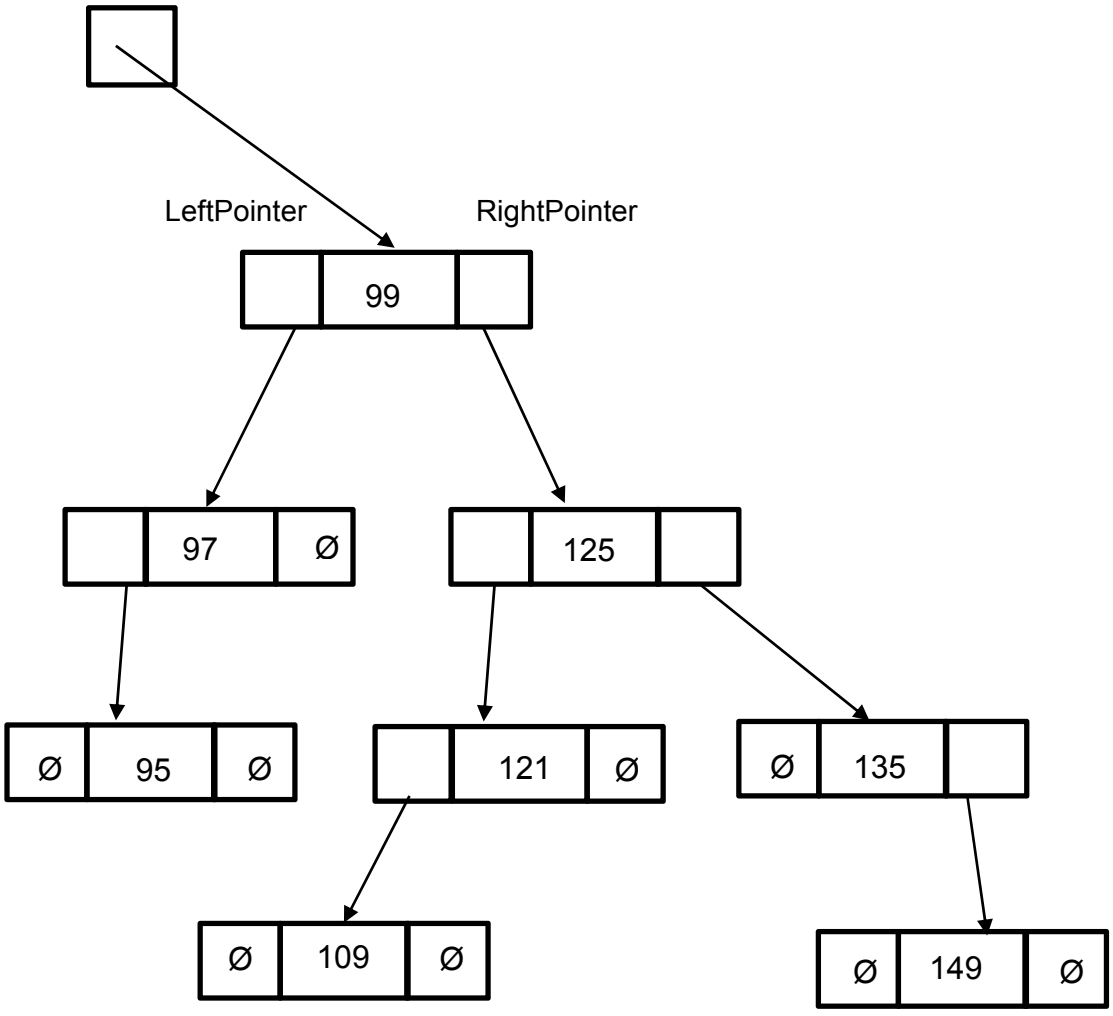
6(a)(i)	1 mark per bullet point:
	
	• Declaring a record type // class etc.
	• Declaring Country as a string
	• Declaring Pointer as an integer
	
	Example:
	
	`TYPE ListElement`
	`  DECLARE Country : STRING`
	`  DECLARE Pointer : INTEGER`
	`ENDTYPE`
		3

Question	Answer	Marks
6(a)(ii)	1 mark per bullet point: • Declaring `CountryList` as an array with 15 elements • Of type `ListElement` Example: `DECLARE CountryList : ARRAY[1 : 15] OF ListElement`	**2**

6(b)	1 mark for each completed statement
	
	`PROCEDURE DeleteNode(NodeValue: STRING, ThisPointer: INTEGER, PreviousPointer: INTEGER)`
	` IF CountryList[ThisPointer].Value = NodeValue`
	`  THEN`
	`  CountryList[ThisPointer].Value ← ""`
	`  IF ListHead = ThisPointer`
	`  THEN`
	`  ListHead ← CountryList[ThisPointer].Pointer`
	`  ELSE`
	`  CountryList[PreviousPointer].Pointer ← CountryList[ThisPointer].Pointer`
	`  ENDIF`
	`  CountryList[LastNode].Pointer ← ThisPointer`
	`  LastNode ← ThisPointer`
	`  CountryList[ThisPointer].Pointer ← -1`
	`  ELSE`
	`  IF CountryList[ThisPointer].Pointer <> -1`
	`  THEN`
	`  CALL DeleteNode(NodeValue,CountryList[ThisPointer].Pointer, ThisPointer)`
	`  ELSE`
	`  OUTPUT "DOES NOT EXIST"`
	`  ENDIF`
	` ENDIF`
	`ENDPROCEDURE`
		5

QUESTION 11.

1(b)	1 mark per bullet point to max 3 • (Linear) data structure • First in First out // FIFO // An item is added to the end of the queue **and** an item is removed from the front • All items are kept in the order they are entered • It has a head pointer and a tail pointer • Can be static or dynamic • A queue can be circular … • …when the (tail) pointer reaches the last position it returns to the first	**3**

Question	Answer
2(a)(i)	1 mark per bullet point • 95 to left of 97 • 109 to left of 121 • 135 to right of 125 • 149 to right of 135 • Null points in all places and no inappropriate pointers RootPointer 

Question	Answer
2(a)(ii)	1 mark per bullet point • FreePointer as 8 • 99 • 125 • 121 and 97 • 109 and 95 • 135 and 149

RootPointer
0

FreePointer
8

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

QUESTION 12.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>2(a)</td><td>1 mark for correct tick

<table><tr><th>Statement</th><th>Tick (✓)</th></tr><tr><td>Last in first out</td><td>✓</td></tr><tr><td>First in first out</td><td></td></tr><tr><td>Last in last out</td><td></td></tr></table></td><td>1</td></tr>
<tr><td>2(b)(i)</td><td>1 mark for correct stack

<table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td>10</td></tr><tr><td>35</td></tr><tr><td>20</td></tr></table></td><td>1</td></tr>
<tr><td>2(b)(ii)</td><td>1 mark for correct stack

<table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td>65</td></tr><tr><td>50</td></tr><tr><td>35</td></tr><tr><td>20</td></tr></table></td><td>1</td></tr>
</table>

Question	Answer	Marks

2(b)(iii)	1 mark for each bullet
	
	• Function `Push` …
	• …taking parameter (returning Boolean)
	• Checking if `Top` = 7 …
	• …returning `FALSE` if full
	• …returning `TRUE` otherwise
	• if not full, increment `Top`
	• ... add parameter to `Top` of `ArrayStack`
	
	`FUNCTION Push (BYVALUE DataItem : Integer) (RETURNS Boolean)`
	`  IF Top = 7`
	`  THEN`
	`  RETURN FALSE`
	`  ELSE`
	`  Top ← Top + 1`
	`  ArrayStack[Top] ← DataItem`
	`  RETURN TRUE`
	`  ENDIF`
	`ENDFUNCTION`
		7

QUESTION 13.

Question	Answer	Marks

2(a)	1 mark per completed statement
	
	`FUNCTION AddToQueue(Number : INTEGER) RETURNS BOOLEAN`
	`  CONSTANT FirstIndex = 0`
	`  CONSTANT LastIndex = 7`
	`  TempPointer ← EndPointer + 1`
	`  IF TempPointer > LastIndex`
	`   THEN`
	`    TempPointer ← FirstIndex`
	`  ENDIF`
	`  IF TempPointer = StartPointer`
	`   THEN`
	`     RETURN FALSE`
	`   ELSE`
	`    EndPointer ← TempPointer`
	`    NumberQueue[EndPointer] ← Number`
	`    RETURN TRUE`
	`  ENDIF`
	`ENDFUNCTION`
		5

2(b)	1 mark per bullet point 1 mark for: • … if the start pointer reaches the end of the queue, it becomes the index of the first element in the queue **Max 3** from: • Checks if the circular queue is empty // Checks if the queue has any data in it • … if it is empty it **reports** that it is empty • If not empty, return the value at the position of the **start pointer** … • … then increments the start pointer	**4**

Question	Answer	Marks
2(c)	1 mark per bullet point to **max 3** e.g. • Stack • Linked list • Dictionary • (Binary) tree	**3**

Question	Answer	Marks
3(a)	1 mark per test data type • Normal / valid • Abnormal / erroneous / invalid • Boundary / extreme	**3**
3(b)(i)	1 mark for each name **Description** / **Name of debugging feature** A point where the program can be halted to see if the program works to this point / **Breakpoint** One statement is executed and then the program waits for input from the programmer to move to the next statement. / **Stepping // step through/over/into**	**2**
3(b)(ii)	1 mark for name, 1 for description e.g. • variable watch window • observe how variables change during execution // view current status of variables • Error list • describes the error // gives line number of error	**2**