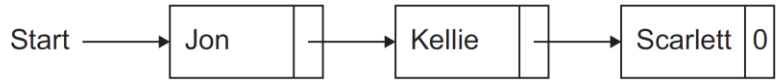
Question No. 1

(a)



Mark as follows:
Three correct items [1]
Indication of correct order with start and termination [1]

(b)

```
Type ListNode
   Pointer as Integer
   Name As String
EndType
```

Mark as follows:
Record structure definition [1]
Pointer field definition [1]
Node data definition [1]

(c)

```
Dim NameList[1..50] As ListNode
```

Mark as follows:
Appropriate size of array [1]
Use of user defined record type [1]

(d) (i)

HeadPointer

0

FreePointer

1

NameList

	Name	Pointer
[1]		2
[2]		3
[3]		4
[4]		5
:		
[49]		50
[50]		0

Mark as follows:
HeadPointer [1]
FreePointer [1]
Pointers[1] – [49] [1]
Pointer[50] [1]

(ii)

```
FOR Index ← 1 TO 49
   NameList[Index].Pointer ← Index + 1
ENDFOR
NameList[50].Pointer ← 0
HeadPointer ← 0
FreePointer ← 1
```

Mark as follows:
Correct FOR loop [1]
Correct setting of Pointer[50], HeadPointer and FreePointer [1]

(e) (i)

```
01 PROCEDURE AddItem(NewItem)
02 //
03   NameList[FreePointer].Name ← NewItem
04   CurrentPointer ← HeadPointer                          [1]
05 //
06   REPEAT
07     IF NameList[CurrentPointer].Name < NewItem          [1]
08       THEN
09         PreviousPointer ← CurrentPointer
10         CurrentPointer ← NameList[CurrentPointer].Pointer  [1]
11     ENDIF
12   UNTIL NameList[CurrentPointer].Name > NewItem
13 //
14   IF CurrentPointer = HeadPointer
15     THEN
16       NameList[FreePointer].Pointer ← HeadPointer
17       HeadPointer ← FreePointer
18     ELSE
19       NameList[FreePointer].Pointer
20                         ← NameList[PreviousPointer].Pointer
21       NameList[PreviousPointer] ← FreePointer
22   ENDIF
23   FreePointer ← NameList[FreePointer].Pointer
24 ENDPROCEDURE
```

(ii) New item placed in node at head of Free List [1]

(iii) Loop that repeats until position of new item located [1]
Records current pointer and then updates current pointer [1]

(iv) Check to see whether new item is first in linked list [1]
If first item then place item at head of list [1]
If not first item then adjust pointers to place it in correct position in list [1]

[Total: 22]

Question No. 2

1 (a) (i)	TYPE LinkedList	1	3
	(DECLARE) Surname : STRING (DECLARE) Ptr : INTEGER	1	
	ENDTYPE	1	
	Accept: LinkedList : RECORD	1	
	Surname : STRING Ptr : INTEGER	1	
	ENDRECORD	1	
	Accept: TYPE LinkedList = RECORD	1	
	Surname : STRING Ptr : INTEGER	1	
	ENDTYPE / ENDRECORD	1	
	Accept: STRUCTURE LinkedList	1	
	(DECLARE) Surname : STRING (DECLARE) Ptr : INTEGER	1	
	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 BOOLEAN	1 1	2

(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	6
	Accept = for assignment	

Question No. 3

6(a)(i)	1 mark per bullet:
	• TYPE ListNode declaration and ENDTYPE
	• DECLARE Player : String
	• DECLARE Pointer : INTEGER
	
	`TYPE ListNode`
	`  DECLARE Player : STRING`
	`  DECLARE Pointer : INTEGER`
	`ENDTYPE`
		3

6(a)(ii)	1 mark per bullet: • DECLARE Scorers : ARRAY[0:9] • OF ListNode `DECLARE Scorers : ARRAY[0:9] OF ListNode`	2

6(b)	1 mark for each completed statement
	
	`FUNCTION SearchList(Find, Position) RETURNS INTEGER`
	`  IF Scorer[Position].Player = Find`
	`  THEN`
	`  RETURN Position`
	`  ELSE`
	`  IF Scorer[Position].Player <> -1`
	`  THEN`
	`  Position ← SearchList(Find, Scorer[Position].Pointer)`
	`  RETURN Position`
	`  ELSE`
	`  RETURN 99`
	`  ENDIF`
	`  ENDIF`
	`ENDPROCEDURE`
		5

Question No. 4

6(a)	• A–B–E • E–C–D with D null pointer	2
6(b)	It indicates the end of the list // it doesn't point anywhere/to any data/to another node	1

6(c)(i)	`FUNCTION FindValue(Value : INTEGER) RETURNS INTEGER`
	`  DECLARE CurrentPointer : INTEGER`
	`  CurrentPointer ← StartPointer`
	`  WHILE CurrentPointer <> NULL AND LinkedList[CurrentPointer].Data <> Value`
	`  CurrentPointer ← LinkedList[CurrentPointer].Pointer`
	`  ENDWHILE`
	`  IF LinkedList[CurrentPointer].Data = Value`
	`  THEN`
	`  RETURN CurrentPointer`
	`  ELSE`
	`  RETURN -1`
	`  ENDIF`
	`ENDFUNCTION`
		6

6(c)(ii)	One mark per bullet point to max 7 • **Function** header, taking parameter (and returning Boolean) • Assign a new pointer to StartPointer • Iterate/recursive calls through nodes correctly updating current pointer • Checking for empty list **and** returning FALSE • Checking if end of list … • … check data in last node • Checking if data found… • … set pointer of found node to NULL (return to free chain) • … if found update previous node pointer to NULL • … return TRUE • If end of list **and** not found then return FALSE	7

```
FUNCTION DeleteNode(NodeData : STRING) RETURNS BOOLEAN
   IF StartPointer = NULL
      THEN
          RETURN FALSE
      ELSE
         CurrentPointer ← StartPointer
         IF LinkedList[CurrentPointer].Data = NodeData
            THEN
               StartPointer ← LinkedList[CurrentPointer].Pointer
               RETURN TRUE
            ELSE
               PreviousPointer ← CurrentPointer
               WHILE CurrentPointer <> NULL AND
                                       LinkedList[CurrentPointer].Data <> NodeData
                   PreviousPointer ← CurrentPointer
                   CurrentPointer ← LinkedList[CurrentPointer].Pointer
               ENDWHILE
```

6(c)(ii)

```
               IF CurrentPointer = NULL
                  THEN
                     IF LinkedList[CurrentPointer].Data = NodeData
                         THEN
                            LinkedList[PreviousPointer].Pointer ← NULL
                            RETURN TRUE
                         ELSE
                            RETURN FALSE
                     ENDIF
                  ELSE
                     IF LinkedList[CurrentPointer].Data = NodeData
                        THEN
                           LinkedList[PreviousPointer].Pointer ←
                                                   LinkedList[CurrentPointer].Pointer
                           LinkedList[CurrentPointer].Pointer ← NULL
                           RETURN TRUE
                     ENDIF
               ENDIF
         ENDIF
   ENDIF
ENDFUNCTION
```