

3(a)	1 mark each - LinkedList declared as 2D array with (min) 20×2 elements (Integer) with all data initialised to -1, all nodes linked correctly (Global) FirstNode (Int) initialised as -1 and (global) FirstEmpty (Int) initialised as 0 VB.NET <pre>Dim LinkedList(20, 2) As Integer Dim FirstNode As Integer Dim FirstEmpty As Integer Sub Main(args As String()) FirstNode = -1 FirstEmpty = 0 For x = 0 To 18 LinkedList(x, 0) = -1 LinkedList(x, 1) = x + 1 Next LinkedList(19, 0) = -1 LinkedList(19, 1) = -1 End Sub</pre>
3(a)	Python <pre>LinkedList = [] #global FirstNode = -1 FirstEmpty = 0 for x in range(0, 19): LinkedList.append([-1, x + 1]) LinkedList[19][0] = -1 LinkedList[19][1] = -1</pre> Java <pre>private static Integer[][] LinkedList = new Integer[20][2]; private static Integer FirstNode; private static Integer FirstEmpty; public static void main(String args[]){ FirstNode = -1; FirstEmpty = 0; for(Integer X = 0; X < 19; X++){ LinkedList[X][0] = -1; LinkedList[X][1] = X + 1; } LinkedList[19][0] = -1; LinkedList[19][1] = -1; }</pre>

3(b)	1 mark each to max 6 - Procedure header (and end) taking (min) 5 data items as input from the user - Checking if linked list is full (FirstEmpty = -1)ending procedure/loop/not doing anything further (otherwise) LinkedList[FirstEmpty, 0] = data input - LinkedList[FirstEmpty, 1] = FirstNode - FirstNode = FirstEmpty - FirstEmpty = LinkedList[FirstEmpty, 1] before any update to FirstEmpty's pointer e.g. Python <pre>def InsertData(): global LinkedList global FirstNode global FirstEmpty for _ in range(5): if FirstEmpty != -1: nextEmpty = LinkedList[FirstEmpty][1] LinkedList[FirstEmpty][0] = int(input("Value: ")) LinkedList[FirstEmpty][1] = FirstNode FirstNode = FirstEmpty FirstEmpty = nextEmpty</pre>
------	---

3(b)

VB.NET

```

Sub InsertData()
    Dim NewItem As Integer
    Dim NextEmpty As Integer

    For x = 0 To 4
        Console.WriteLine("Enter the next number")
        NewItem = Console.ReadLine()

        If FirstEmpty = -1 Then
            x = 5
        Else
            NextEmpty = LinkedList(FirstEmpty, 1)
            LinkedList(FirstEmpty, 0) = NewItem
            LinkedList(FirstEmpty, 1) = FirstNode
            FirstNode = FirstEmpty
            FirstEmpty = NextEmpty
        End If

        Next x
    End Sub

```

3(b)

Java

```

public static void InsertData(){
    Integer NewItem;
    Integer CurrentPointer = 0;
    Integer PreviousPointer = 0;
    Scanner scanner = new Scanner(System.in);
    Integer NextEmpty;

    for(Integer X = 0; X < 5; X++){
        System.out.println("Enter the next number");
        NewItem = Integer.parseInt(scanner.nextLine());
        if(FirstEmpty == -1){
            X = 5;
        }else{
            NextEmpty = LinkedList[FirstEmpty][1];
            LinkedList[FirstEmpty][0] = NewItem;
            LinkedList[FirstEmpty][1] = FirstNode;
            FirstNode = FirstEmpty;
            FirstEmpty = NextEmpty;
        }
    }
}

```

3(c)(i)	1 mark each - Procedure header (and end) starting with node at index FirstNode and outputting data LinkedList[FirstNode,0] - Following pointers until end reached and outputting data for each node Python <pre>def OutputLinkedList(): global LinkedList global FirstNode global FirstEmpty CurrentPointer = FirstNode Flag = True while Flag: print(LinkedList[CurrentPointer][0]) CurrentPointer = LinkedList[CurrentPointer][1] if CurrentPointer == -1: Flag = False</pre> VB.NET <pre>Sub OutputLinkedList() Dim CurrentPointer As Integer = FirstNode Dim Flag As Boolean = True While Flag Console.WriteLine(LinkedList(CurrentPointer, 0)) CurrentPointer = LinkedList(CurrentPointer, 1) If CurrentPointer = -1 Then Flag = False End If End While</pre>
---------	---

3(c)(i)	End Sub Java <pre>public static void OutputLinkedList(){ Integer CurrentPointer = FirstNode; Boolean Flag = true; while (Flag){ System.out.println(LinkedList[CurrentPointer][0]); CurrentPointer = LinkedList[CurrentPointer][1]; if (CurrentPointer == -1){Flag = false;} } }</pre>
3(c)(ii)	1 mark for calling InsertData() then OutputLinkedList() Python <pre>InsertData() OutputLinkedList()</pre> VB.NET <pre>InsertData() OutputLinkedList()</pre> Java <pre>InsertData(); OutputLinkedList();</pre>
3(c)(iii)	1 mark for inputs of 5 1 2 3 8 and output of 8 3 2 1 5

- 3(d)(i) 1 mark each to max 5
- Procedure header (and end) with parameter
 - Checking data in FirstNode against parameter ...
 - ... (if found) updating FirstNode to LinkedList[FirstNode, 1]
 - (Otherwise) following pointers in loop/recursive call ...
 - ... comparing to data to remove each time
 - ... storing previous pointer through each loop...
 - ... when found, updating previous pointer to found node's pointer
 - Adding deleted node to end of/start of empty list (and updating FirstEmpty if needed)

Python

```
def RemoveData(ItemToRemove):
    global LinkedList
    global FirstNode
    global FirstEmpty

    if LinkedList[FirstNode][0] == ItemToRemove:
        NewFirst = LinkedList[FirstNode][1]
        LinkedList[FirstNode][1] = FirstEmpty
        FirstEmpty = FirstNode
        FirstNode = NewFirst

    else:
        if FirstNode != -1:
            CurrentPointer = FirstNode
            PreviousNode = -1
            while (ItemToRemove != LinkedList[CurrentPointer][0] and CurrentPointer != -1):
                PreviousNode = CurrentPointer
                CurrentPointer = LinkedList[CurrentPointer][1]

            if ItemToRemove == LinkedList[CurrentPointer][0]:
                LinkedList[PreviousNode][1] = LinkedList[CurrentPointer][1]
                LinkedList[CurrentPointer][0] = -1
                LinkedList[CurrentPointer][1] = FirstEmpty
                FirstEmpty = CurrentPointer
```

3(d)(i) VB.NET

```
Sub RemoveData(ItemToRemove)
    If LinkedList(FirstNode, 0) = ItemToRemove Then
        Dim NewFirst As Integer = LinkedList(FirstNode, 1)
        LinkedList(FirstNode, 1) = FirstEmpty
        FirstEmpty = FirstNode
        FirstNode = NewFirst
    Else
        If FirstNode <> -1 Then
            Dim CurrentPointer As Integer = FirstNode
            Dim PreviousNode As Integer = -1
            Dim Flag As Boolean = True
            Dim Found As Boolean = False
            While Flag And Not (Found)
                If (CurrentPointer <> -1) Then
                    If (ItemToRemove <> LinkedList(CurrentPointer, 0)) Then
                        PreviousNode = CurrentPointer
                        CurrentPointer = LinkedList(CurrentPointer, 1)
                    Else
                        Found = True
                    End If
                Else
                    Flag = False
                End If
            End While

            If Found Then
                LinkedList(PreviousNode, 1) = LinkedList(CurrentPointer, 1)
                LinkedList(CurrentPointer, 0) = -1
                LinkedList(CurrentPointer, 1) = FirstEmpty
                FirstEmpty = CurrentPointer
            End If
        End If
    End If
End Sub
```

3(d)(i)	Java <pre> public static void RemoveData(Integer ItemToRemove){ Integer CurrentPointer = 0; Integer PreviousNode = 0; Integer NewFirst = 0; if(LinkedList[FirstNode][0] == ItemToRemove){ NewFirst = LinkedList[FirstNode][1]; LinkedList[FirstNode][1] = FirstEmpty; FirstEmpty = FirstNode; FirstNode = NewFirst; }else{ if (FirstNode != -1){ CurrentPointer = FirstNode; PreviousNode = -1; while(ItemToRemove != LinkedList[CurrentPointer][0] && CurrentPointer != -1){ PreviousNode = CurrentPointer; CurrentPointer = LinkedList[CurrentPointer][1]; } if(ItemToRemove == LinkedList[CurrentPointer][0]){ LinkedList[PreviousNode][1] = LinkedList[CurrentPointer][1]; LinkedList[CurrentPointer][0] = -1; LinkedList[CurrentPointer][1] = FirstEmpty; FirstEmpty = CurrentPointer; } } } } </pre>
---------	---

3(d)(ii)	1 mark for calling RemoveData(5), outputting "After", calling OutputLinkedList() Python <pre> LinkedList = [] FirstNode = -1 FirstEmpty = 0 for x in range(0, 19): LinkedList.append([-1, x + 1]) InsertData() OutputLinkedList() RemoveData(5) print("After") OutputLinkedList() </pre> VB.NET <pre> Sub Main(args As String()) FirstNode = -1 FirstEmpty = 0 For x = 0 To 19 LinkedList(x, 0) = -1 LinkedList(x, 1) = x + 1 Next InsertData() OutputLinkedList() RemoveData(5) Console.WriteLine("After") OutputLinkedList() End Sub </pre>
----------	---

3(d)(ii)	Java <pre> public static void main(String args[]){ FirstNode = -1; FirstEmpty = 0; for(Integer X = 0; X < 20; X++){ LinkedList[X][0] = -1; LinkedList[X][1] = X + 1; } InsertData(); OutputLinkedList(); RemoveData(5); System.out.println("After"); OutputLinkedList(); } </pre>
3(d)(iii)	1 mark for input and output. Test data 1: Input 5 6 8 9 5 'After' Output: 9 8 6 5 Test data 2: Input 10 7 8 5 6 "After" Output: 6 8 7 10

8(a)	One mark for reason, one for benefit Reason: (Program is) easier to design / implement / test / debug / modify Benefit: Easier to check that each stage works as expected
------	---

8(b)	Example algorithm based on finding position of first non-space character and then using substring function: <pre> FUNCTION DeleteSpaces(Line : STRING) RETURNS STRING DECLARE NewLine : STRING DECLARE EndOfLeading : BOOLEAN DECLARE Count, NumSpaces : INTEGER DECLARE NextChar : CHAR CONSTANT Space = " " NumSpaces ← 0 EndOfLeading ← FALSE FOR Count ← 1 TO LENGTH(Line) NextChar ← MID(Line, Count, 1) IF NextChar <> Space AND EndOfLeading = FALSE THEN NumSpaces ← Count - 1 // the number to trim EndOfLeading = TRUE ENDIF NEXT Count NewLine ← RIGHT(Line, LENGTH(Line) - NumSpaces) RETURN NewLine ENDFUNCTION </pre> Mark as follows: 1 Loop to length of parameter // Loop until first non-space character in Line 2 Extract a character in a loop 3 Identify <u>first</u> non-space character in a loop 4 Attempt at removing leading spaces in Line 5 Leading spaces removed from Line // Create new string without leading space 6 Return a string following a reasonable attempt at removing leading spaces in Line
------	--

8(c)

Example:

```

PROCEDURE Stage_2(F1, F2 : STRING)

    DECLARE Line : STRING
    DECLARE Count : INTEGER

    Count ← 0

    OPEN F1 FOR READ
    OPEN F2 FOR APPEND

    WHILE NOT EOF(F1)
        READFILE F1, Line
        Line ← DeleteSpaces(Line)
        Line ← DeleteComment(Line)
        IF Line <> "" THEN
            WRITEFILE F2, Line // skip blank lines
        ELSE
            Count ← Count + 1
        ENDIF
    ENDWHILE

    CLOSEFILE F1
    CLOSEFILE F2

    OUTPUT Count, " blank lines were removed"

ENDPROCEDURE

```

Mark as follows:

- 1 Procedure heading, parameters, ending
- 2 Open both files in correct modes **and** subsequently close
- 3 Loop to EOF(F1)
- 4 Read a line from F1 **in a loop**
- 5 Assign return values from DeleteComment() and DeleteSpaces()
in a loop
- 6 Check return value following both MP5 function calls is not an empty
string and if so write to F2 **in a loop**
- 7 Count the blank lines **in a loop**
- 8 Output number of blank lines removed following a reasonable attempt
after the loop