Algorithm for merging the elements of two singly linked lists into a single list

Input: A pointer to the first node of the first singly linked list, say HEAD1 and a pointer to the first node of the second list, say HEAD2.

Output: A pointer to the first node of the merged list, say HEAD or suitable unsuccessful message.

Data structure used: Two singly linked lists where each node of each list contains a data element, say DATA and the address of the immediate next node, say LINK with HEAD holding the address of the first node.

Steps:

```
Begin
If HAD1 = NULL And HEAD2 = NULL
Then
        Print "Both the lists are empty, merging not possible"
Else
        //Copying the elements of the first list into new list
        Set count = 0
        Set temp1 = HEAD1
        While (temp1!= NULL)
        Begin
                If count = 0
                Then
                        Set HEAD = Getnode()
                        If HEAD = NULL
                        Then
                                Print "Memory for new node is not available"
                        Else
                                Set HEAD →DATA = temp1 →DATA
                                Set HEAD →LINK = NULL
                                Set count = count + 1
                                Set temp = HEAD
                        End If
                Else
                        Set temp →LINK = Getnode()
                        If temp→LINK = NULL
                        Then
                                Print "Memory for new node is not available"
                        Else
                                Set temp = temp→LINK
```

```
                    Set temp →DATA = temp1 →DATA
                    Set temp →LINK = NULL
                    Set count = count + 1
                End If
            End If
        End While
        //Copying the nodes of the second list into the new list
        Set temp2 = HEAD2
        While (temp2!= NULL)
        Begin
            If count = 0
            Then
                Set HEAD = Getnode()
                If HEAD = NULL
                Then
                    Print "Memory for new node is not available"
                Else
                    Set HEAD →DATA = temp2 →DATA
                    Set HEAD →LINK = NULL
                    Set count = count + 1
                    Set temp = HEAD
                End If
            Else
                Set temp →LINK = Getnode()
                If temp→LINK = NULL
                Then
                    Print "Memory for new node is not available"
                Else
                    Set temp = temp→LINK
                    Set temp →DATA = temp2→DATA
                    Set temp →LINK = NULL
                    Set count = count + 1
                End If
            End If
        End While
End If
End
```

Algorithm for merging elements of two singly linked lists into a single list by taking elements alternatively

Input: A pointer to the first node of the singly linked list, say HEAD.

Output: All the nodes containing negative values are deleted from the singly linked list or suitable unsuccessful message.

Data structure used: A singly linked list where each node contains a data element, say DATA and the address of the immediate next node, say LINK with HEAD holding the address of the first node.

Steps:

Algorithm for deleting all the negative items from a singly linked list

Input: A pointer to the first node of the singly linked list, say HEAD.

Output: All the nodes containing negative values are deleted from the singly linked list or suitable unsuccessful message.

Data structure used: A singly linked list where each node contains a data element, say DATA and the address of the immediate next node, say LINK with HEAD holding the address of the first node.

Steps:

1. Begin
   //Counting the no of negative items in the list
2. Set count = 0
3. Set temp2 = HEAD
4. While(temp2!= NULL)
5. Begin
6. If temp2 →DATA < 0
7. Then
8. Set count = count + 1
9. End If
10. Set temp2 = temp2 →LINK
11. End While
12. If (count = 0)
13. Then
14. Print “No negative items present in the list, deletion not possible”
15. Else
16. Set i = 1
17. While (i<= count)
18. Begin
19. Set temp2 = HEAD

```
20.             While (temp2 != NULL)
21.             Begin
22.                     If (temp2 →DATA < 0)
23.                     Then
24.                             Break
25.                     Else
26.                             Set temp1 = temp2
27.                             Set temp2 = temp2 →LINK
28.                     End If
29.             End While
30.             Print "Deleted item is" temp2→DATA
31.             If (temp2 = HEAD)
32.             Then
33.                     Set Head = temp2 →LINK
34.             Else
35.                     Set temp1 →LINK = temp2 →LINK
36.             End If
37.             free(temp2)
38.             Set i = i + 1
39.         End While
40. End If
41. End
```

Algorithm for deleting all the items with a specific value from a singly linked list

Input: A pointer to the first node of the singly linked list, say HEAD and a value, say VAL that has to be searched in the nodes for deletion.

Output: All the nodes containing VAL are deleted from the singly linked list or suitable unsuccessful message.

Data structure used: A singly linked list where each node contains a data element, say DATA and the address of the immediate next node, say LINK with HEAD holding the address of the first node.

Steps:

1. Begin
   //Counting the no of nodes with VAL in the list
2. Set count = 0
3. Set temp2 = HEAD
4. While(temp2!= NULL)
5. Begin
6. If temp2 →DATA = VAL

```
7.          Then
8.                  Set count = count + 1
9.          End If
10.         Set temp2 = temp2 →LINK
11. End While
12. If (count = 0)
13. Then
14.         Print “No nodes with VAL is present in the list, deletion not possible”
15. Else
16.         Set i = 1
17.         While (i<= count)
18.         Begin
19.                 Set temp2 = HEAD
20.                 While (temp2 != NULL)
21.                 Begin
22.                         If (temp2 →DATA = VAL)
23.                         Then
24.                                 Break
25.                         Else
26.                                 Set temp1 = temp2
27.                                 Set temp2 = temp2 →LINK
28.                         End If
29.                 End While
30.                  If (temp2 = HEAD)
31.                 Then
32.                         Set Head = temp2 →LINK
33.                 Else
34.                         Set temp1 →LINK = temp2 →LINK
35.                 End If
36.                 free(temp2)
37.                 Set i = i + 1
38.         End While
39. End If
40. End
```

Algorithm for deleting all the nodes with value greater than a specific value from a singly linked list

Input: A pointer to the first node of the singly linked list, say HEAD a value, say VAL that has to be searched in the nodes for deletion.

Output: All the nodes with values greater than VAL are deleted from the singly linked list or suitable unsuccessful message.

Data structure used: A singly linked list where each node contains a data element, say DATA and the address of the immediate next node, say LINK with HEAD holding the address of the first node.

Steps:

1. Begin
   //Counting the no of items in the list
2. Set count = 0
3. Set temp2 = HEAD
4. While(temp2!= NULL)
5. Begin
6. Set count = count + 1
7. Set temp2 = temp2 →LINK
8. End While
9. If (count = 0)
10. Then
11. Print “List is empty, deletion not possible”
12. Else
13. Set i = 1
14. While (i<= count)
15. Begin
16. Set temp2 = HEAD
17. While (temp2 != NULL)
18. Begin
19. If (temp2 →DATA > VAL)
20. Then
21. Break
22. Else
23. Set temp1 = temp2
24. Set temp2 = temp2 →LINK
25. End If
26. End While
27. If temp2 = NULL

28. Then
29. Print “No node exists in the list with data greater than VAL”
30. Exit
31. End If
32. Print “Deleted item is” temp2→DATA
33. If (temp2 = HEAD)
34. Then
35. Set Head = temp2 →LINK
36. Else
37. Set temp1 →LINK = temp2 →LINK
38. End If
39. free(temp2)
40. Set i = i + 1
41. End While
42. End If
43. End

Algorithm for finding the median of the elements of a singly linked list of integers

Input: A pointer to the first node of the singly linked list, say HEAD.

Output: The median of the elements of the list, say MEDIAN or suitable unsuccessful message.

Data structure used: A singly linked list where each node contains a data element, say DATA and the address of the immediate next node, say LINK with HEAD holding the address of the first node.

Steps:

1. Begin
   //Sorting the nodes of the list
2. Set temp1 = HEAD
3. While temp1 →LINK != NULL
4. Begin
5. Set temp2 = temp1→LINK
6. While(temp2!=NULL)
7. Begin
8. If temp1→DATA > temp2→DATA
9. Then
10. Set temp = temp1→DATA
11. Set temp1→DATA = temp2→DATA
12. Set temp2→DATA = temp
13. End If
14. Set temp2 = temp2→LINK

```
15.         End While
16. Set temp1= temp1→LINK
17. End While
    //Counting the number of nodes in the list
18. Set count = 0
19. Set temp1 = HEAD
20. While(temp1!= NULL)
21. Begin
22.         Set count = count + 1
23.         Set temp = temp →LINK
24. End While
25. //Finding the median of the elements
26. If (count % 2 == 0)
27. Then
28.         Set temp1 = HEAD
29.         Set i = 1
30.         While (i < count /2 )
31.         Begin
32.                 Set temp1 = temp1 →LINK
33.                 Set i = i + 1
34.         End While
35.         Set MEDIAN = (temp→DATA + (temp→LINK) →DATA) /2
36. Else
37.         Set temp1 = HEAD
38.         Set i = 1
39.         While (i < (count + 1) / 2 )
40.         Begin
41.                 Set temp1 = temp1 →LINK
42.                 Set i = i + 1
43.         End While
44.         Set MEDIAN = temp →DATA
45. End If
46. End
```