Operations on a Singly Linked List

Algorithm for splitting a singly linked list from a specific value

Input: A pointer to the first node of the singly linked list, say HEAD and a value, say VAL from which singly linked list has to be split.

Output: The singly linked list is split into two lists with HEAD holding the address of the first list and HEAD_NEW holding the address of the first node of the second list with the first node with VAL 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
2. If HEAD = NULL
3. Then
4. Print “List is empty, splitting is not possible”
5. Else
6. Set temp2 = HEAD
7. While temp2! = NULL
8. Begin
9. If temp2 →DATA = VAL
10. Then
11. Break
12. End If
13. Set temp1 = temp2
14. Set temp2 = temp2 →LINK
15. End While
16. If temp2 = NULL
17. Then
18. Print “VAL not found in the list, splitting not possible”
19. Else
20. If temp2 = HEAD
21. Then
22. Print “VAL found in the first node, splitting not possible”
23. Else
24. Set temp1→LINK = NULL
25. Set HEAD_NEW = temp2
26. End If
27. End If
28. End If
29. End

Algorithm for splitting a singly linked list from a specific position

Input: A pointer to the first node of the singly linked list, say HEAD and a specific position, say POS from which singly linked list has to be split.

Output: The singly linked list is split into two lists with HEAD holding the address of the first list and HEAD_NEW holding the address of the first node of the second list with the first node with VAL 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
2. If HEAD = NULL
3. Then
4. Print “List is empty, splitting is not possible”
5. Else
6. Set temp2 = HEAD
7. Set count = 0
8. While temp2! = NULL
9. Begin
10. Set count = count + 1
11. Set temp2 = temp2 →LINK
12. End While
13. If POS ≤ 1 Or POS > count
14. Then
15. Print “List can’t be split from POS”
16. Else
17. Set temp2 = HEAD
18. Set i = 1
19. While i < POS
20. Begin
21. Set temp1 = temp2
22. Set temp2 = temp2 →LINK
23. Set i = i + 1
24. End While
25. Set temp1→LINK = NULL
26. Set HEAD_NEW = temp2
27. End If
28. End If
29. End

Algorithm for splitting a singly linked list of integers into two lists containing the even and odd integers

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

Output: The singly linked list is split into two lists with EVEN holding the address of the first list containing the even integers and ODD holding the address of the first node of the second list containing the odd integers 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
2.  If HEAD = NULL
3.  Then
4.      Print "List is empty, splitting is not possible"
5.  Else
6.      Set c_even = 0
7.      Set c_odd = 0
8.      Set temp = HEAD
9.      While temp ! = NULL
10.     Begin
11.         If temp→DATA % 2 = 0
12.         Then
13.             Set c_even = c_even + 1
14.             If c_even = 1
15.             Then
16.                 Set EVEN = Getnode()
17.                 If EVEN = NULL
18.                 Then
19.                     Print "Memory allocation is not possible"
20.                 Else
21.                     Set EVEN→DATA = temp→DATA
22.                     Set EVEN→LINK = NULL
23.                     Set temp1 = EVEN
24.                 End If
25.             Else
26.                 Set temp1→LINK = Getnode()
27.                 If temp1→LINK = NULL
28.                 Then
29.                     Print "Memory allocation is not possible"
30.                 Else
31.                     Set temp1 = temp1→LINK
32.                     Set temp1→DATA = temp →DATA
33.                     Set temp1→LINK = NULL
```

```
                    End If
                End If
            Else
                Set c_odd = c_odd + 1
                If c_odd = 1
                Then
                    Set ODD = Getnode()
                    If ODD = NULL
                    Then
                        Print “Memory allocation is not possible”
                    Else
                        Set ODD→DATA = temp→DATA
                        Set ODD→LINK = NULL
                        Set temp2 = ODD
                    End If
                Else
                    Set temp2→LINK = Getnode()
                    If temp2→LINK = NULL
                    Then
                        Print “Memory allocation is not possible”
                    Else
                        Set temp2 = temp2→LINK
                        Set temp2→DATA = temp →DATA
                        Set temp2→LINK = NULL
                    End If
                End If
            End If
            Set temp = temp→LINK
        End While
End If
End
```

Algorithm for splitting a singly linked list of integers into two lists containing positive and negative integers

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

Output: The singly linked list is split into two lists with POSITIVE holding the address of the first list containing the positive integers and NEGATIVE holding the address of the first node of the second list containing the negative integers 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
2. If HEAD = NULL

```
3.  Then
4.      Print “List is empty, splitting is not possible”
5.  Else
6.      Set c_positive = 0
7.      Set c_negative = 0
8.      Set temp = HEAD
9.      While temp ! = NULL
10.     Begin
11.         If temp→DATA > 0
12.         Then
13.             Set c_positive = c_positive + 1
14.             If c_positive = 1
15.             Then
16.                 Set POSITIVE = Getnode()
17.                 If POSITIVE = NULL
18.                 Then
19.                     Print “Memory allocation is not possible”
20.                 Else
21.                     Set POSITIVE→DATA = temp→DATA
22.                     Set POSITIVE→LINK = NULL
23.                     Set temp1 = POSITIVE
24.                 End If
25.             Else
26.                 Set temp1→LINK = Getnode()
27.                 If temp1→LINK = NULL
28.                 Then
29.                     Print “Memory allocation is not possible”
30.                 Else
31.                     Set temp1 = temp1→LINK
32.                     Set temp1→DATA = temp →DATA
33.                     Set temp1→LINK = NULL
34.                 End If
35.             End If
36.         Else
37.             Set c_negative = c_negative + 1
38.             If c_negative = 1
39.             Then
40.                 Set NEGATIVE = Getnode()
41.                 If NEGATIVE = NULL
42.                 Then
43.                     Print “Memory allocation is not possible”
```

```
                    Else
                        Set NEGATIVE →DATA = temp→DATA
                        Set NEGATIVE →LINK = NULL
                        Set temp2 = NEGATIVE
                    End If
                Else
                    Set temp2→LINK = Getnode()
                    If temp2→LINK = NULL
                    Then
                        Print "Memory allocation is not possible"
                    Else
                        Set temp2 = temp2→LINK
                        Set temp2→DATA = temp →DATA
                        Set temp2→LINK = NULL
                    End If
                End If
            End If
            Set temp = temp→LINK
        End While
End If
End
```

Algorithm for reversing the nodes of a singly linked list

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

Output: The singly linked list is reversed that is the last node becomes the first, the second last node becomes the second and so on with HEAD holding the address of the first node 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
2. If HEAD = NULL
3. Then
4.      Print "The list is empty"
5. Else If HEAD →LINK = NULL
6. Then
7.      Print "The list contains only one node, reversal node possible"
8. Else
9.      Set curr = HEAD
10.     Set prev = curr →LINK
11.     Set curr →LINK = NULL
12.     While prev→LINK != NULL
13.     Begin
```

14. Set temp = prev →LINK
15. Set prev →LINK = curr
16. Set curr = prev
17. Set prev = temp
18. End While
19. Set prev →LINK = curr
20. Set HEAD = prev
21. End If
22. End

Algorithm for sorting the nodes of a singly linked list

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

Output: The elements of the singly linked list are sorted in order otherwise 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
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
18. End

Algorithm to remove the duplicate items of a singly linked list

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

Output: The duplicate elements of the singly linked list are removed 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
2. If HEAD = NULL

```
3.  Then
4.      Print “List is empty, removal of duplicate is possible”
5.  Else If HEAD →LINK = NULL
6.  Then
7.      Print “List contains only one node, removal is possible”
8.  Else
9.      // Sorting the list
10.     Set temp1 = HEAD
11.     While temp1 →LINK != NULL
12.     Begin
13.         Set temp2 = temp1→LINK
14.         While(temp2!=NULL)
15.         Begin
16.             If temp1→DATA > temp2→DATA
17.             Then
18.                 Set temp = temp1→DATA
19.                 Set temp1→DATA = temp2→DATA
20.                 Set temp2→DATA = temp
21.             End If
22.         Set temp2 = temp2→LINK
23.         End While
24.     Set temp1= temp1→LINK
25.     End While
26.     // Removing duplicate items
27.     Set curr = HEAD
28.     While curr →LINK != NULL
29.     Begin
30.         If (curr→DATA)  = ((curr→LINK)→DATA)
31.         Then
32.             Set next = (curr →LINK)→LINK
33.             free(curr→LINK)        //free(), a procedure that dynamically de-allocates a node
34.             Set curr→LINK = next
35.         End If
36.     End While
37. End If
38. End
```