**Data Structures**

Deleting duplicate node from a sorted linked list

**Deleting duplicate value node from a sorted linked list:**



 1 add 2nd 1 add 3rd 3 NULL  

* Data in the linked list must be in sorted order.
* Ensure that no two nodes shouldn’t have same data.
* Traverse through the linked list until the last but one node.
* While traversing through the linked list compare data between two

Consecutive nodes and if data is repeated, point the next pointer of the

Current node to the next pointer of the next node so that the repeated

Node gets deleted from the linked list.

* Release the memory of the deleted node using free().
* Normal approach.
* Recursive approach.

**Function for Deleting duplicate value node from a sorted linked list(while loop):**

//deleting duplicate nodes  
lin\_list \* DeleteDuplicateNode(lin\_list \*head)  
{  
 **if**(head==NULL){  
 **return** head;  
 }  
 lin\_list \*temp=head;  
 **while**(temp->next!=NULL){  
 **if**(temp->data==temp->next->data){  
 temp->next=temp->next->next;  
 }  
 **else**{  
 temp=temp->next;  
 }  
 }  
 **return** head;  
}

**Function Deleting duplicate value node from a sorted linked list(using recursion):**

//deleting duplicate nodes  
lin\_list \* DeleteDuplicateNode(lin\_list \*head)

{  
 **if**(head->next==NULL || head==NULL){  
 **return** head;  
 }  
 **if**(head->data==head->next->data){  
 head->next=head->next->next;  
 head=DeleteDuplicateNode(head);  
 **return** head;  
 }  
 head->next=DeleteDuplicateNode(head->next);  
 **return** head;  
}

**Whole program:**

#include<stdio.h>  
#include<stdlib.h>  
//creating a node.  
**typedef struct** lin\_list{  
 **int** data;  
 **struct** lin\_list \*next;  
}lin\_list;  
//inserting nodes  
lin\_list \*insertnode(lin\_list \*head,**int** data) {  
 lin\_list \*newnode=(lin\_list\*)malloc(**sizeof**(lin\_list));  
 newnode->data=data;  
 newnode->next=head;  
 head=newnode;  
 **return** head;  
}  
//printing the linked list.  
**void** PrintElements(lin\_list \*head){  
 //base condition  
 **if**(head==NULL){  
 **return**;  
 }  
 printf("%d ",head->data);  
 PrintElements(head->next);  
}  
//deleting duplicate nodes  
lin\_list \* DeleteDuplicateNode(lin\_list \*head) {  
 **if**(head==NULL){  
 **return** head;  
 }  
 lin\_list \*temp=head;  
 **while**(head->next!=NULL){  
 **if**(head->data==head->next->data){  
 head->next=head->next->next;  
 }  
 **else**{  
 head=head->next;  
 }  
 }  
 **return** temp;  
}  
//main  
**int** main(){  
 lin\_list \*headA=NULL;  
  
 //inserting elements into linked list  
 headA=insertnode(headA,3);  
 headA=insertnode(headA,2);  
 headA=insertnode(headA,2);  
 headA=insertnode(headA,2);  
 headA=insertnode(headA,1);  
 PrintElements(headA);printf("\n");  
 //deleting Duplicate nodes  
 headA=DeleteDuplicateNode(headA);  
 PrintElements(headA);

headA=insertnode(headA,1);printf("\n");

PrintElements(headA);printf("\n");

headA=DeleteDuplicateNode(headA);

PrintElements(headA);  
 **return** 0;  
}

**Output:**

1 2 2 2 3

1 2 3

1 1 2 3

1 2 3