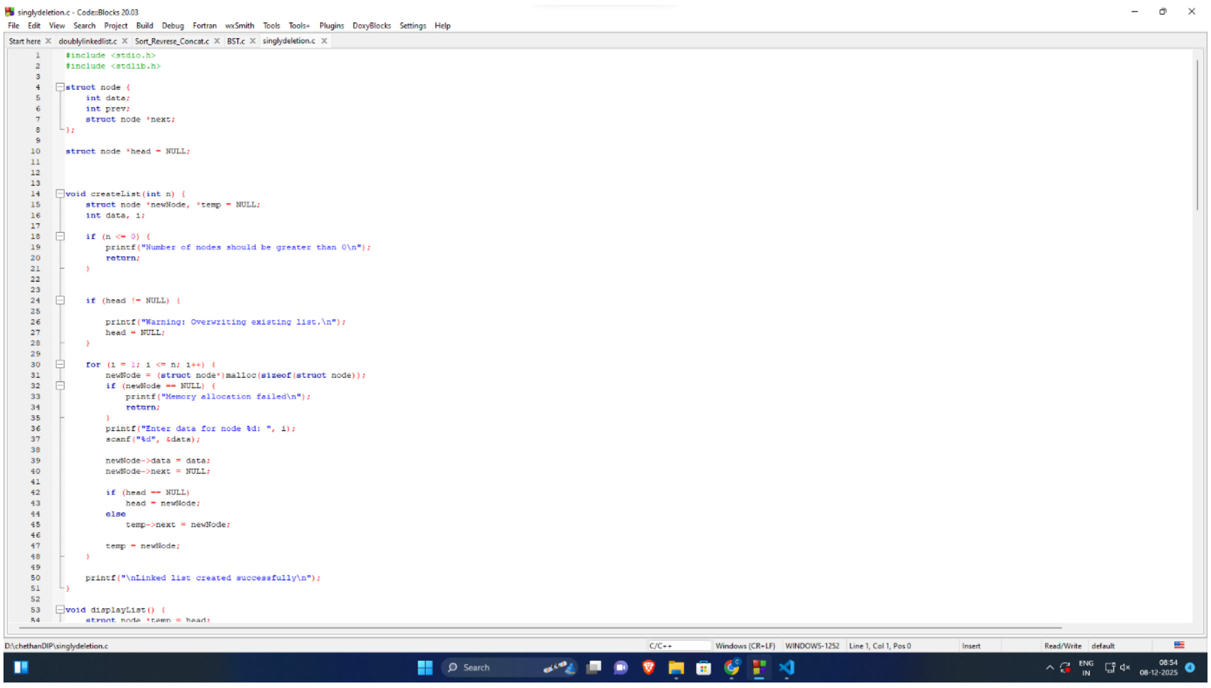
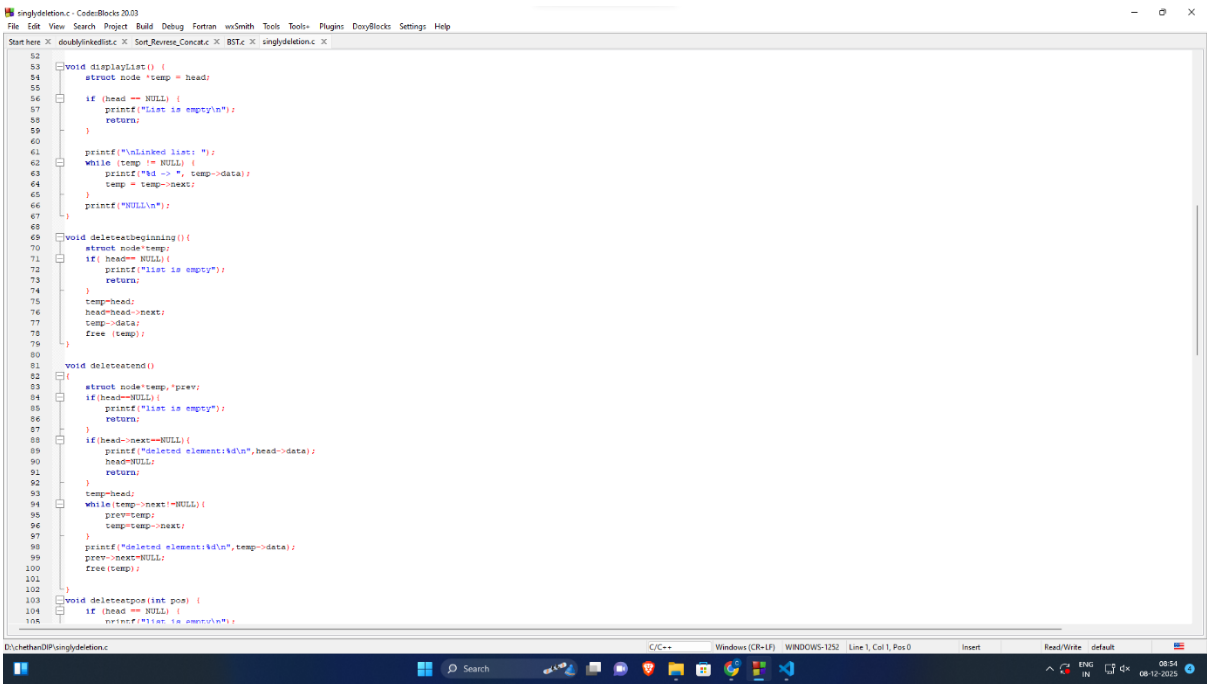
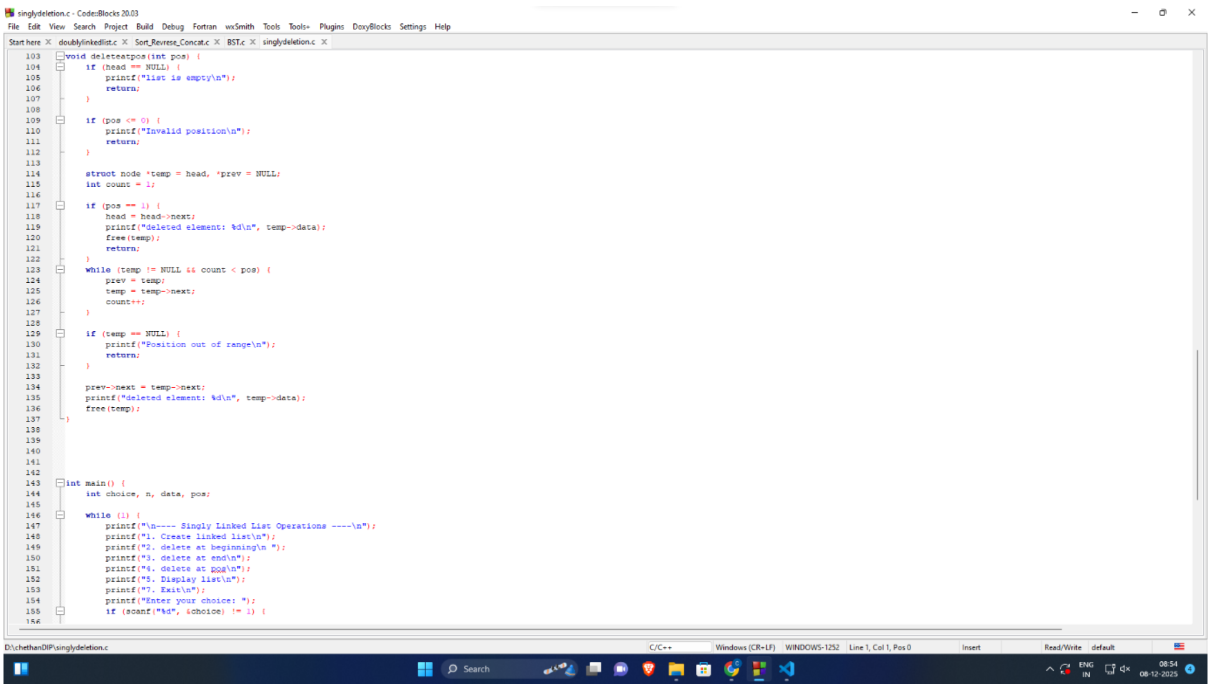

singlydeletion.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here doublylinkedlist.c Sort_Revrese_Concat.c BST.c singlydeletion.c
D:\chethanDIP\singlydeletion.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default
Search
ENG IN 08:54 08-12-2025

```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    int prev;
    struct node *next;
};

struct node *head = NULL;



void createList(int n) {
    struct node *newNode, *temp = NULL;
    int data, i;

    if (n <= 0) {
        printf("Number of nodes should be greater than 0\n");
        return;
    }


    if (head != NULL) {

        printf("Warning: Overwriting existing list.\n");
        head = NULL;
    }

    for (i = 1; i <= n; i++) {
        newNode = (struct node*)malloc(sizeof(struct node));
        if (newNode == NULL) {
            printf("Memory allocation failed\n");
            return;
        }
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;

        if (head == NULL)
            head = newNode;
        else
            temp->next = newNode;

        temp = newNode;
    }

    printf("\nLinked list created successfully\n");
}

void displayList() {
    struct node *temp = head;
```

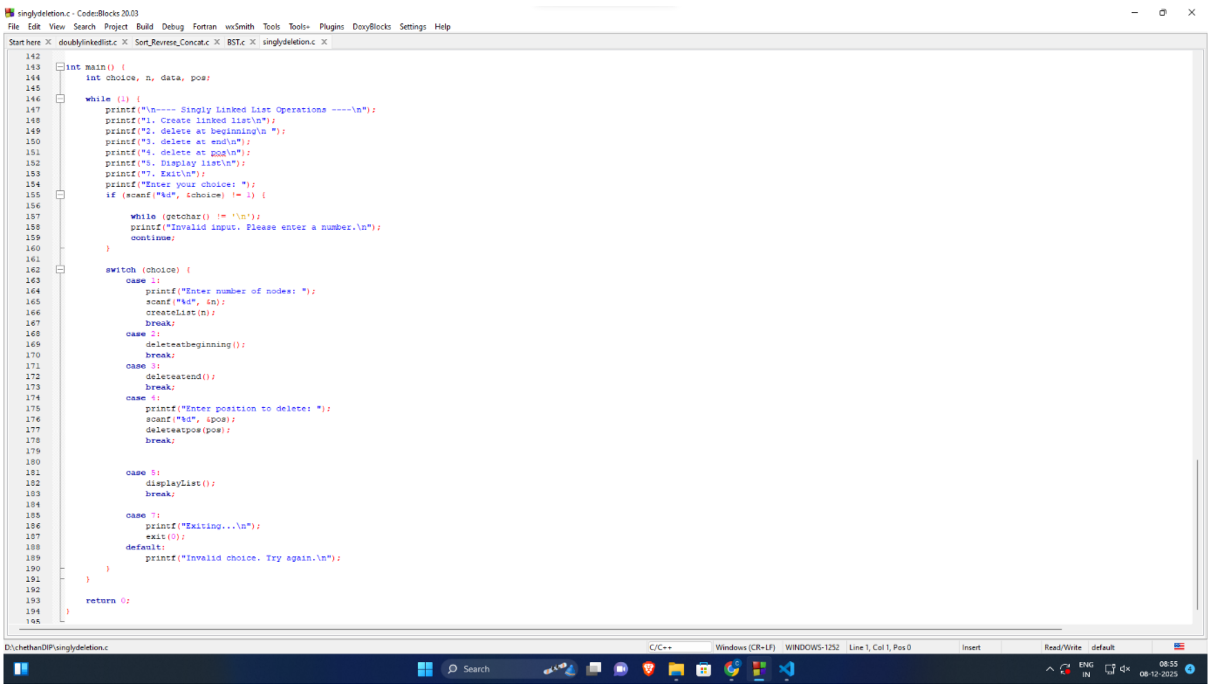

singlydeletion.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here doublylinkedlist.c Sort_Revrese_Concat.c BST.c singlydeletion.c
D:\chethanDIP\singlydeletion.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default
Search
ENG IN 08:54 08-12-2025

```c

void displayList() {
    struct node *temp = head;

    if (head == NULL) {
        printf("List is empty\n");
        return;
    }

    printf("\nLinked list: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

void deleteatbeginning(){
    struct node*temp;
    if( head== NULL){
        printf("list is empty");
        return;
    }
    temp=head;
    head=head->next;
    temp->data;
    free (temp);
}

void deleteatend()
{
    struct node*temp,*prev;
    if(head==NULL){
        printf("list is empty");
        return;
    }
    if(head->next==NULL){
        printf("deleted element:%d\n",head->data);
        head=NULL;
        return;
    }
    temp=head;
    while(temp->next!=NULL){
        prev=temp;
        temp=temp->next;
    }
    printf("deleted element:%d\n",temp->data);
    prev->next=NULL;
    free(temp);

}
void deleteatpos(int pos) {
    if (head == NULL) {
        printf("list is empty\n");
```



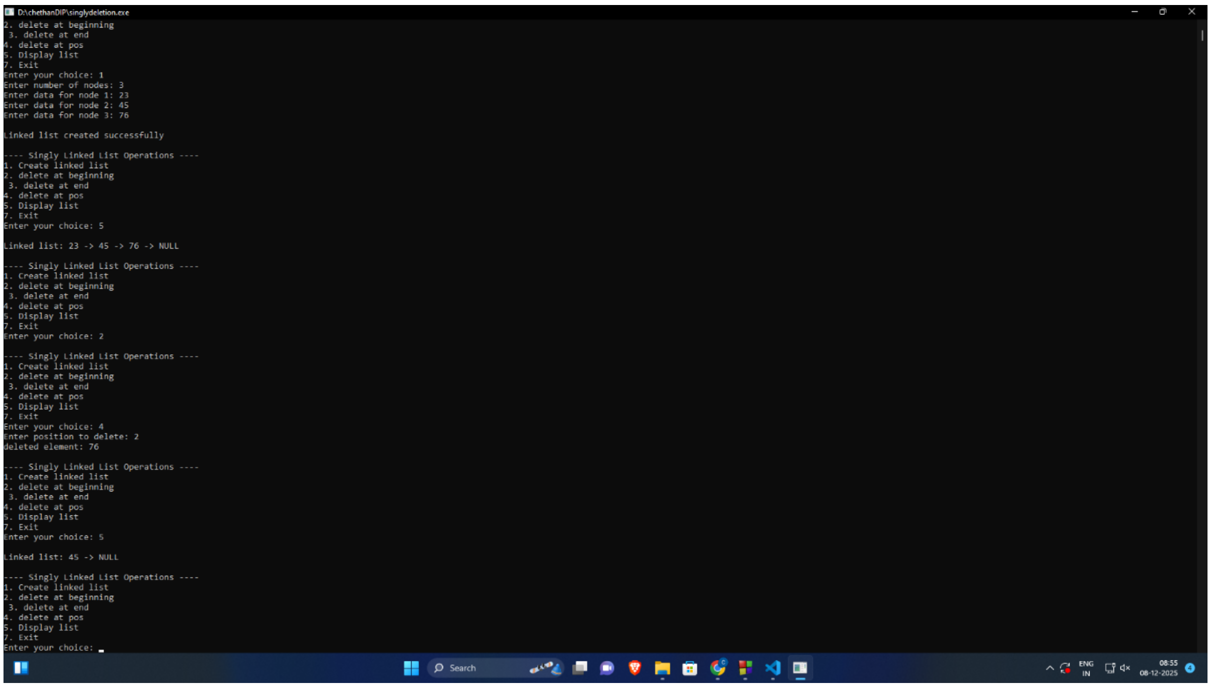
```c
void deleteatpos(int pos) {
    if (head == NULL) {
        printf("list is empty\n");
        return;
    }

    if (pos <= 0) {
        printf("Invalid position\n");
        return;
    }

    struct node *temp = head, *prev = NULL;
    int count = 1;

    if (pos == 1) {
        head = head->next;
        printf("deleted element: %d\n", temp->data);
        free(temp);
        return;
    }
    while (temp != NULL && count < pos) {
        prev = temp;
        temp = temp->next;
        count++;
    }

    if (temp == NULL) {
        printf("Position out of range\n");
        return;
    }

    prev->next = temp->next;
    printf("deleted element: %d\n", temp->data);
    free(temp);
}




int main() {
    int choice, n, data, pos;

    while (1) {
        printf("\n---- Singly Linked List Operations ----\n");
        printf("1. Create linked list\n");
        printf("2. delete at beginning\n ");
        printf("3. delete at end\n");
        printf("4. delete at pos\n");
        printf("5. Display list\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        if (scanf("%d", &choice) != 1) {
```

D:\chethanDIP\singlydeletion.c

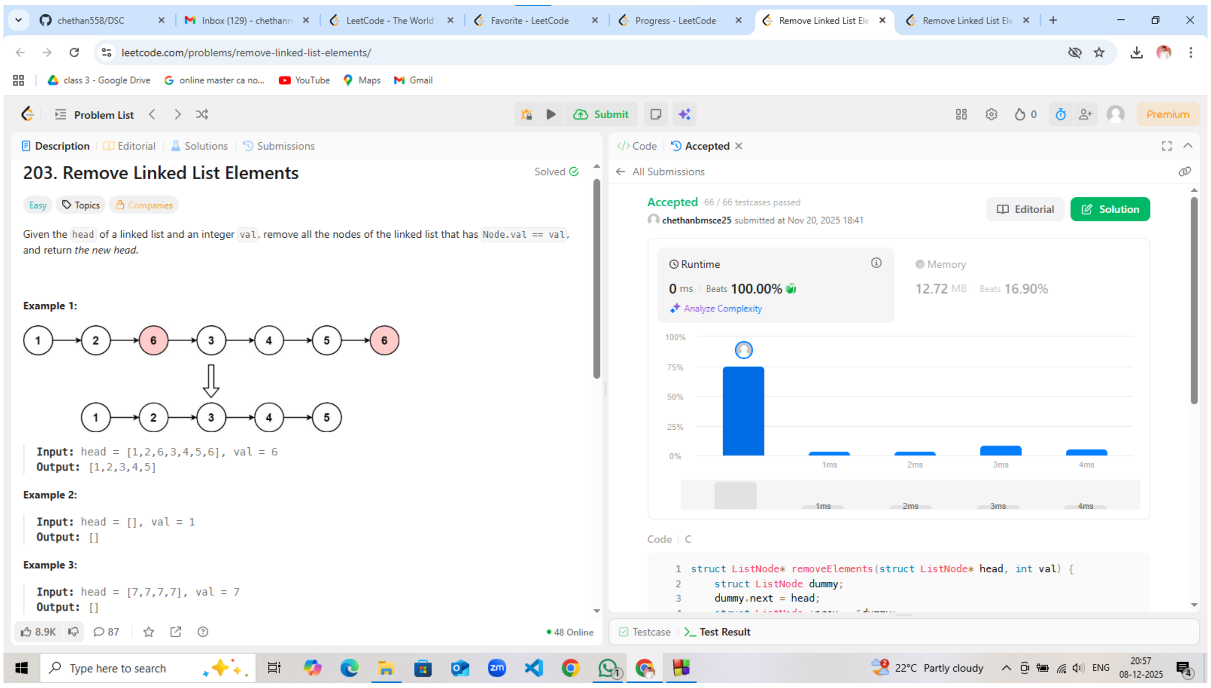


```c

int main() {
    int choice, n, data, pos;

    while (1) {
        printf("\n---- Singly Linked List Operations ----\n");
        printf("1. Create linked list\n");
        printf("2. delete at beginning\n ");
        printf("3. delete at end\n");
        printf("4. delete at pos\n");
        printf("5. Display list\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        if (scanf("%d", &choice) != 1) {

             while (getchar() != '\n');
             printf("Invalid input. Please enter a number.\n");
             continue;
        }

        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                deleteatbeginning();
                break;
            case 3:
                deleteatend();
                break;
            case 4:
                printf("Enter position to delete: ");
                scanf("%d", &pos);
                deleteatpos(pos);
                break;


            case 5:
                displayList();
                break;

            case 7:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```

D:\chethanDIP\singlydeletion.c



```
D:\chethanDIP\singlydeletion.exe
2. delete at beginning
 3. delete at end
4. delete at pos
5. Display list
7. Exit
Enter your choice: 1
Enter number of nodes: 3
Enter data for node 1: 23
Enter data for node 2: 45
Enter data for node 3: 76

Linked list created successfully

---- Singly Linked List Operations ----
1. Create linked list
2. delete at beginning
 3. delete at end
4. delete at pos
5. Display list
7. Exit
Enter your choice: 5

Linked list: 23 -> 45 -> 76 -> NULL

---- Singly Linked List Operations ----
1. Create linked list
2. delete at beginning
 3. delete at end
4. delete at pos
5. Display list
7. Exit
Enter your choice: 2

---- Singly Linked List Operations ----
1. Create linked list
2. delete at beginning
 3. delete at end
4. delete at pos
5. Display list
7. Exit
Enter your choice: 4
Enter position to delete: 2
deleted element: 76

---- Singly Linked List Operations ----
1. Create linked list
2. delete at beginning
 3. delete at end
4. delete at pos
5. Display list
7. Exit
Enter your choice: 5

Linked list: 45 -> NULL

---- Singly Linked List Operations ----
1. Create linked list
2. delete at beginning
 3. delete at end
4. delete at pos
5. Display list
7. Exit
Enter your choice: _
```



leetcode.com/problems/remove-linked-list-elements/

203. Remove Linked List Elements

Solved

Easy · Topics · Companies

Given the `head` of a linked list and an integer `val`, remove all the nodes of the linked list that has `Node.val == val`, and return *the new head*.

Example 1:

```
Input: head = [1,2,6,3,4,5,6], val = 6
Output: [1,2,3,4,5]
```

Example 2:

```
Input: head = [], val = 1
Output: []
```

Example 3:

```
Input: head = [7,7,7,7], val = 7
Output: []
```

Accepted 66 / 66 testcases passed

chethanbmsce25 submitted at Nov 20, 2025 18:41

Runtime: 0 ms | Beats 100.00%

Memory: 12.72 MB | Beats 16.90%

Code | C

```c
struct ListNode* removeElements(struct ListNode* head, int val) {
    struct ListNode dummy;
    dummy.next = head;
```

DATE: PAGE:

Impliment singly linked list following operations a) create linked list b) Deletion of first element, specified element and last element c) Display.

<u>pseudocode</u>

```
Node {
    data
    next
}

head
```

<u>Delete from Beginning</u>

```
deleteBeginning ():
    if head = null;
        print ("list is empty")
    endif
    temp = head
    head = head.next
    free temp
```

<u>Delete at end</u>

```
Deleteatend ():
    if head == Null:
        print "list is empty"
        return
    if head.next == Null:
            free head
            head = null
            return
    current = head
    while current.next.next != Null:
            current = current.next
    end
```

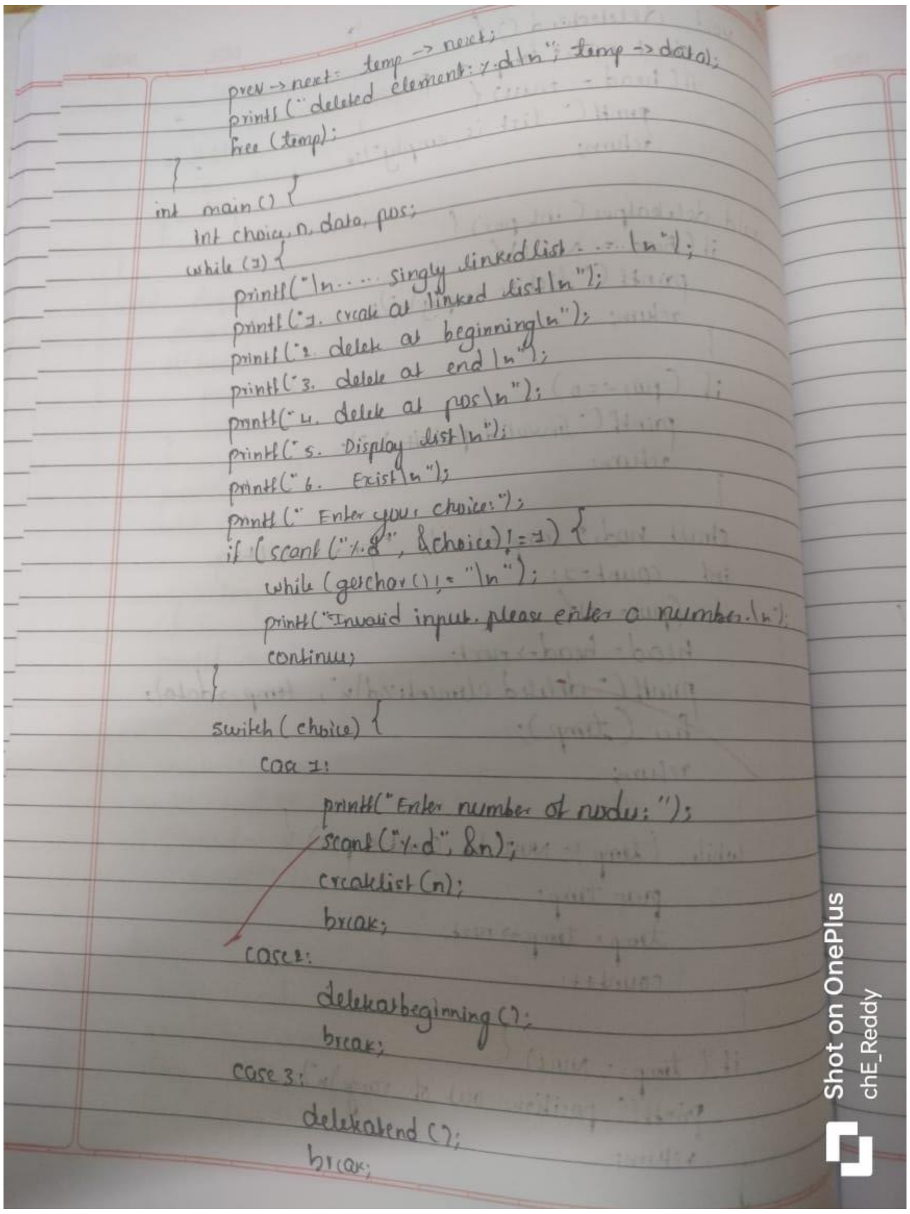
DATE:

```
            Free current.next ;
            current.next = null ;
        end function

deletevalueatposition(value):
    if head = = null:
        print "list is empty"
        return
    endif
    if head.data = = value:
        temp = head
        head = head.next
        Free temp
        return
    end
    current = head
    while current.next != null and current.next data != value
            current = current.value
    end
    If current.next = null:
        print value not found
    end
    temp = current.next
    current.next = current.next.next
    Free temp
end
```

```
#include <stdio.h>
#include <stdio.h>

struct node {
      int data;
      int prev;
      struct node * next;
 };
struct node* head = NULL;
void createlist (int n) {
     struct node * newnode, * temp = NULL;
     int data, i;
     if (n<=0) {
        printf(" number of node should be greater than 0\n");
        return;
     }
     if (head != NULL) {
        printf(" warning: overwriting existing list.\n");
        head = NULL;
     }
     for (i=1; i<=n; i++) {
        newnode = (struct node *) malloc (sizeof (struct node)
        if (newnode == NULL) {
           printf(" memory allocation failed\n");
           return;
        }
        printf(" Enter data for node %d: ", i);
        scanf("%d", &data);
        newnode -> data = data;
        newNode -> next = NULL;
```

```
            if(head == NULL)
                head = newNode;
            else
                temp->next = newNode;
            temp = newnode;
        }
}
void  displaylist () {
      struct node* temp = head;
      if(head == NULL) {
          printf(" list is empty\n");
          return;
      }
      printf("\n linked list: ");
      while ( temp != NULL) {
           printf(" %d -> ", temp -> data);
           temp = temp -> next;
      }
      printf(" null\n");
}

void  deleteatbeginning () {
      struct node* temp;
      if (head == NULL) {
           printf(" list is empty");
           return;
      }
      temp = head;
      head = head -> next;
      temp -> data;
      free (temp);
}
```

```
void deleteatend () {
    struct node* temp, *  prev;
    if( head == NULL) {
        printf(" list is empty");
        return;

void deleteatpos (int pos) {
    if (head == NULL){
        printf (" list is empty\n");
        return;
    }
    if (pos<=0)
        printf("Invalid posistion\n");
        return;
    }
    struct node* temp = head,  *prev = NULL;
    int   count=1;
    if  (pos==1){
        head= head->next;
        printf(" deleted element:%d\n", temp->data);
        free (temp);
        return;
    }
    while  (temp != NULL && count < pos) {
            prev= temp;
            temp= temp -> next;
            count++;
    }
    if ( temp== NULL) {
        printf(" position out of range\n");
        return;
    }
```



```
        prev -> next = temp -> next;
        printf ("deleted element: %d\n", temp -> data);
        free (temp);
}
int main () {
    int choice, n, data, pos;
    while (1) {
        printf ("\n.... singly linkedlist ...\n");
        printf ("1. create a linked list\n");
        printf ("2. delete at beginning\n");
        printf ("3. delete at end\n");
        printf ("4. delete at pos\n");
        printf ("5. Display list\n");
        printf ("6. Exist\n");
        printf (" Enter your choice: ");
        if (scanf ("%d", &choice) != 1) {
            while (getchar () != "\n");
            printf ("Invalid input. please enter a number.\n");
            continue;
        }
        switch (choice) {
            case 1:
                printf (" Enter number of nodes: ");
                scanf ("%d", &n);
                createlist (n);
                break;
            case 2:
                deleteatbeginning ();
                break;
            case 3:
                deleteatend ();
                break;
            case 4:
                printf("Enter position to delete");
                scanf("%d", &pos);
                deleteatpos (pos);
                break;
            case 5:
                displaylist();
                break;
            case 6:
                printf("Exiting... \n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

Output:

```
singly linkedlist
1. create a linked list
2 delete at beginning
3. delete at end
4. delete at pos
5 Display list
6. Exist
Enter your choice: 1
Enter number of nodes: 4
Enter data for node: 2
Enter data for node2: 4
Enter data for node3: 6
Enter data for node4: 8
Enter your choice:5
2 -> 4 -> 6 -> 8 -> NULL
Enter your choice: 2
Deleted element:10
Enter your choice:5
4 -> 6 -> 8 -> NULL
Enter your choice:4
Enter your position to delete:2
Deleted element:6
Enter your choice:5
4 -> 8 -> NULL
Enter your choice:3
Deleted element: 8
```

[illegible] 20/1/25

Remove linked list elements

Leetcode:

```
struct ListNode* RemoveElements(struct ListNode* head, int val){
    struct ListNode dummy;
    dummy.next = head;
    struct ListNode *prev = &dummy;
    while(prev->next != NULL){
        if(prev->next->val == val) {
            struct ListNode * toDelete = prev->next;
            prev->next = toDelete->next;
            free(toDelete);
        } else {
                prev = prev->next;
        }
    }

    return dummy.next.
}
```