

```
singlydel.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran ws5nsh Tools Tools- Plugins DeyBlocks Settings Help
Start here X doublylinkedlist X Sort_Revers_Concate X BSTc X singlydel.c X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node {
5     int data;
6     int prev;
7     struct node *next;
8 };
9
10 struct node *head = NULL;
11
12
13
14 void createList(int n) {
15     struct node *newNode, *temp = NULL;
16     int data, i;
17
18     if (n <= 0) {
19         printf("Number of nodes should be greater than 0\n");
20         return;
21     }
22
23
24     if (head != NULL) {
25         printf("Warning: Overwriting existing list.\n");
26         head = NULL;
27     }
28
29
30     for (i = 1; i <= n; i++) {
31         newNode = (struct node *) malloc(sizeof(struct node));
32         if (newNode == NULL) {
33             printf("Memory allocation failed\n");
34             return;
35         }
36         printf("Enter data for node %d: ", i);
37         scanf("%d", &data);
38
39         newNode->data = data;
40         newNode->next = NULL;
41
42         if (head == NULL)
43             head = newNode;
44         else
45             temp->next = newNode;
46
47         temp = newNode;
48     }
49
50     printf("Unlinked list created successfully\n");
51 }
52
53 void displayList() {
54     struct node *temp = head;
```

```
singlydel.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran ws5nsh Tools Tools- Plugins DeyBlocks Settings Help
Start here X doublylinkedlist X Sort_Revers_Concate X BSTc X singlydel.c X
52
53 void displayList() {
54     struct node *temp = head;
55
56     if (head == NULL) {
57         printf("List is empty\n");
58         return;
59     }
60
61     printf("Unlinked list: ");
62     while (temp != NULL) {
63         printf("%d -> ", temp->data);
64         temp = temp->next;
65     }
66     printf("NULL\n");
67
68
69 void deleteFromBeginning() {
70     struct node *temp;
71     if (head == NULL) {
72         printf("List is empty");
73         return;
74     }
75     temp = head;
76     head = head->next;
77     temp->data;
78     free(temp);
79 }
80
81 void deleteFromEnd() {
82     struct node *temp, *prev;
83     if (head == NULL) {
84         printf("List is empty");
85         return;
86     }
87     if (head->next == NULL) {
88         printf("Deleted element id\n", head->data);
89         head = NULL;
90         return;
91     }
92     temp = head;
93     while (temp->next != NULL) {
94         prev = temp;
95         temp = temp->next;
96     }
97     printf("Deleted element id\n", temp->data);
98     prev->next = NULL;
99     free(temp);
100
101
102 void deleteFromPos(int pos) {
103     if (head == NULL) {
104         printf("List is empty\n");
105     }
```

```

singlydeletion.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocxBlocks Settings Help
Start here X doublylinkedlist.c X Sort_Revers_Concate X BSTc X singlydeletion.c X
103 void deleteatpos(int pos) {
104     if (head == NULL) {
105         printf("List is empty\n");
106         return;
107     }
108     if (pos <= 0) {
109         printf("Invalid position\n");
110         return;
111     }
112     struct node *temp = head, *prev = NULL;
113     int count = 1;
114     if (pos == 1) {
115         head = head->next;
116         printf("Deleted element: %d\n", temp->data);
117         free(temp);
118         return;
119     }
120     while (temp != NULL && count < pos) {
121         prev = temp;
122         temp = temp->next;
123         count++;
124     }
125     if (temp == NULL) {
126         printf("Position out of range\n");
127         return;
128     }
129     prev->next = temp->next;
130     printf("Deleted element: %d\n", temp->data);
131     free(temp);
132 }
133
134 int main() {
135     int choice, n, data, pos;
136     while (1) {
137         printf("\n--- Singly Linked List Operations ---\n");
138         printf("1. Create linked list\n");
139         printf("2. delete at beginning\n");
140         printf("3. delete at end\n");
141         printf("4. delete at pos\n");
142         printf("5. Display list\n");
143         printf("7. Exit\n");
144         printf("Enter your choice: ");
145         if (scanf("%d", &choice) != 1) {
146             continue;
147         }
148         while (getchar() != '\n') {
149             printf("Invalid input. Please enter a number.\n");
150             continue;
151         }
152         switch (choice) {
153             case 1:
154                 printf("Enter number of nodes: ");
155                 scanf("%d", &n);
156                 createlist(n);
157                 break;
158             case 2:
159                 deleteatbegin();
160                 break;
161             case 3:
162                 deleteatend();
163                 break;
164             case 4:
165                 printf("Enter position to delete: ");
166                 scanf("%d", &pos);
167                 deleteatpos(pos);
168                 break;
169             case 5:
170                 displaylist();
171                 break;
172             case 7:
173                 printf("Exiting...\n");
174                 exit(0);
175             default:
176                 printf("Invalid choice. Try again.\n");
177         }
178     }
179     return 0;
180 }

```

```

singlydeletion.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocxBlocks Settings Help
Start here X doublylinkedlist.c X Sort_Revers_Concate X BSTc X singlydeletion.c X
142 int main() {
143     int choice, n, data, pos;
144     while (1) {
145         printf("\n--- Singly Linked List Operations ---\n");
146         printf("1. Create linked list\n");
147         printf("2. delete at beginning\n");
148         printf("3. delete at end\n");
149         printf("4. delete at pos\n");
150         printf("5. Display list\n");
151         printf("7. Exit\n");
152         printf("Enter your choice: ");
153         if (scanf("%d", &choice) != 1) {
154             continue;
155         }
156         while (getchar() != '\n') {
157             printf("Invalid input. Please enter a number.\n");
158             continue;
159         }
160         switch (choice) {
161             case 1:
162                 printf("Enter number of nodes: ");
163                 scanf("%d", &n);
164                 createlist(n);
165                 break;
166             case 2:
167                 deleteatbegin();
168                 break;
169             case 3:
170                 deleteatend();
171                 break;
172             case 4:
173                 printf("Enter position to delete: ");
174                 scanf("%d", &pos);
175                 deleteatpos(pos);
176                 break;
177             case 5:
178                 displaylist();
179                 break;
180             case 7:
181                 printf("Exiting...\n");
182                 exit(0);
183             default:
184                 printf("Invalid choice. Try again.\n");
185         }
186     }
187     return 0;
188 }

```

```
D:\chethan\DP\ singly linked list\main.c
1. delete at beginning
2. delete at end
3. delete at pos
4. delete at pos
5. Display list
6. 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
6. 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
6. Exit
Enter your choice: 2
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
6. Exit
Enter your choice: 4
Enter position to delete: 2
Deleted element: 76
Linked list: 23 -> 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
6. 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
6. Exit
Enter your choice: 6
```

203. Remove Linked List Elements

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

chethanbmcc25 submitted at Nov 20, 2025 18:41

Runtime: 0 ms | Beats 100.00%

Memory: 12.72 MB | Beats 16.90%

```
1 struct ListNode* removeElements(struct ListNode* head, int val) {
2     struct ListNode dummy;
3     dummy.next = head;
4     struct ListNode* curr = dummy.next;
5     while (curr != NULL) {
6         if (curr->val == val) {
7             struct ListNode* next = curr->next;
8             free(curr);
9             curr = next;
10        } else {
11            curr = curr->next;
12        }
13    }
14    return dummy.next;
15 }
```

Implement singly linked list following operations: a) create linked list
b) Deletion of first element, specified element and last element
c) display.

pseudocode

```
node {
    data
    next
}
```

head

Delete from Beginning

Delete Beginning ():

if head == null;

print ("List is empty")

endif

temp = head

head = head.next

free temp

Delete at end

Delete at end ():

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



```

Free current.next
current.next = null
end function

```

```

delete value at position (value):

```

```

if head == null:
    print "List is empty"
    return

```

```

end if

```

```

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.next

```

```

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 <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;
}

```

```

void displaylist() {
    struct node *temp = head;
    if (head == NULL) {
        printf("list is empty\n");
        return;
    }
    printf("In 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\n");
        return;
    }

```

```

    temp = head;
    head = head->next;
    temp->data;
    free(temp);
}

```

```
void deleteend() {  
    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 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 ... \n");
        printf("1. create at 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. 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:

```

print("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 linked list

1. create a linked list
2. delete at beginning
3. delete at end
4. delete at pos
5. Display list
6. Exit

Enter your choice: 1

Enter number of nodes: 4

Enter data for node: 2

Enter data for node: 4

Enter data for node: 6

Enter data for node: 8

Enter your choice: 5
 2 → 4 → 6 → 8 → null
 Enter your choice: 2
 Deleted element: 4
 Enter your choice: 5
 2 → 4 → 6 → 8 → null
 Enter your choice: 4
 Enter your position to delete: 2
 Deleted element: 6
 Enter your choice: 5
 2 → 8 → null
 Enter your choice: 3
 Deleted element: 8

20/11/25
 Sen

Remove linked list elements

code:

```

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;
}
  
```