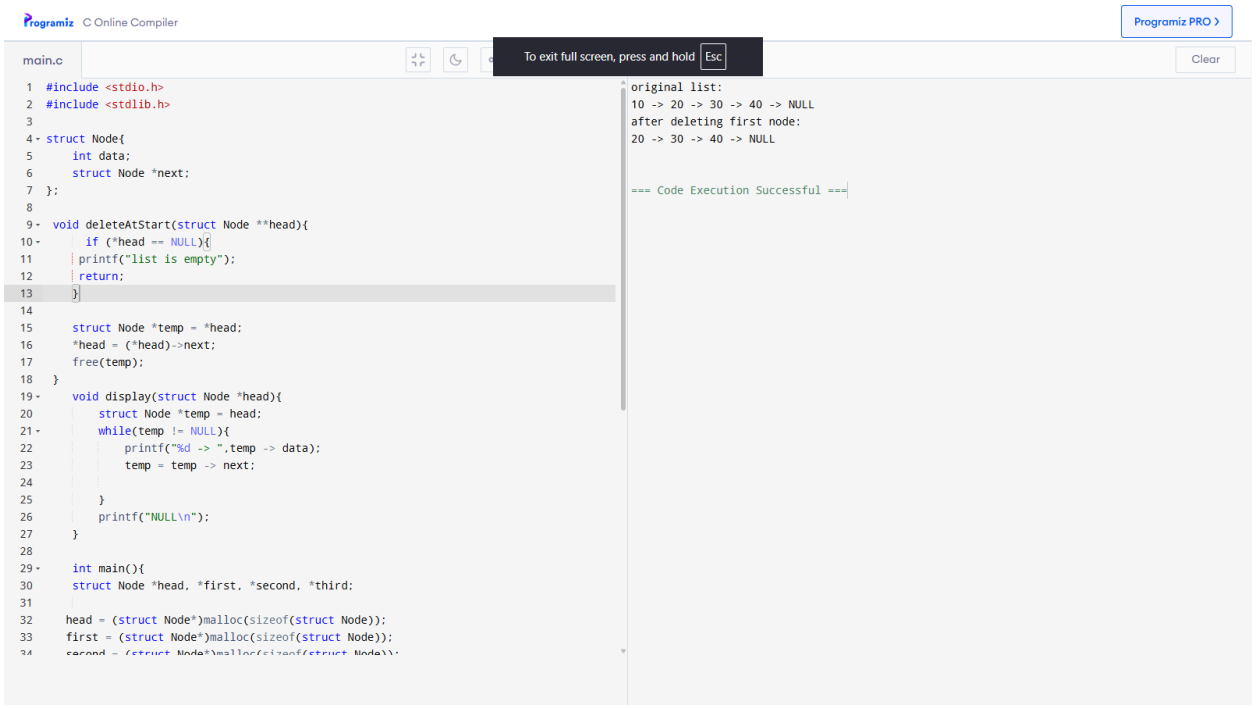
1.Delection at Begining



Programiz C Online Compiler

Programiz PRO ›

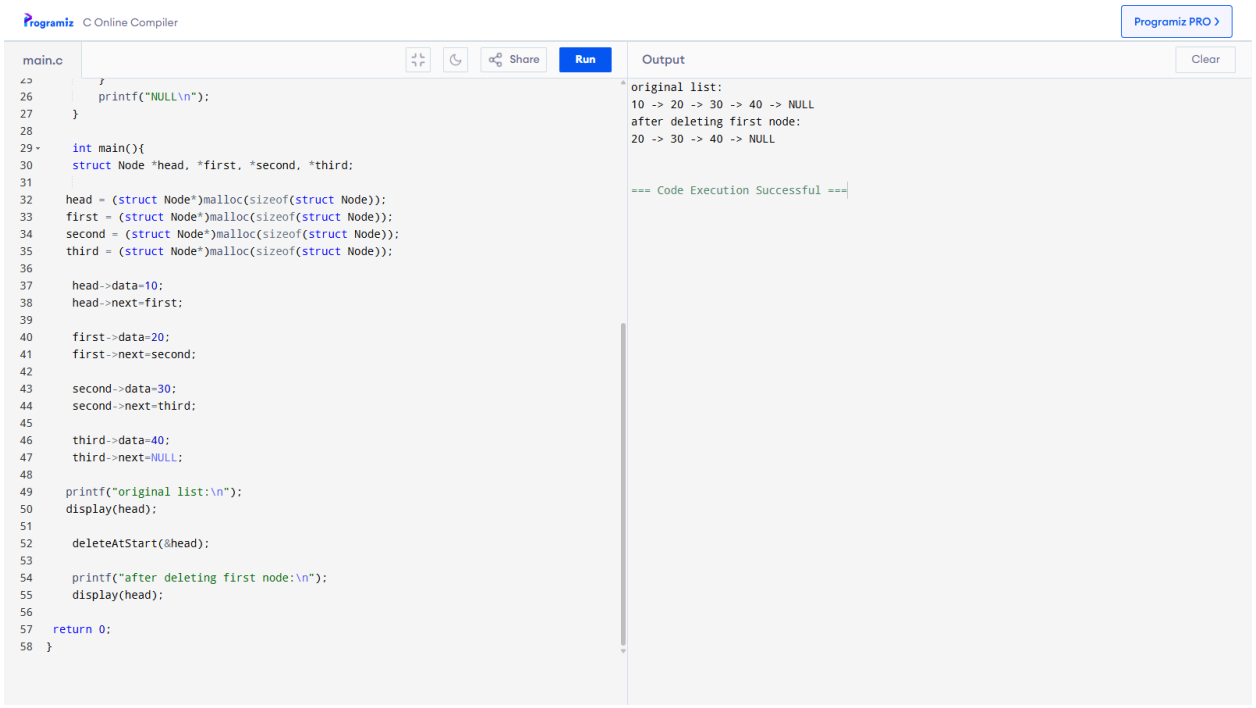
main.c

To exit full screen, press and hold Esc

Clear

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

struct Node{
    int data;
    struct Node *next;
};

 void deleteAtStart(struct Node **head){
      if (*head == NULL){
    printf("list is empty");
    return;
    }

    struct Node *temp = *head;
    *head = (*head)->next;
    free(temp);
 }
    void display(struct Node *head){
        struct Node *temp = head;
        while(temp != NULL){
            printf("%d -> ",temp -> data);
            temp = temp -> next;

        }
        printf("NULL\n");
    }

    int main(){
    struct Node *head, *first, *second, *third;

   head = (struct Node*)malloc(sizeof(struct Node));
   first = (struct Node*)malloc(sizeof(struct Node));
```

```
original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
20 -> 30 -> 40 -> NULL


=== Code Execution Successful ===
```



Programiz C Online Compiler

Programiz PRO ›

main.c

Share

Run

Output

Clear

```c
        }
        printf("NULL\n");
    }

    int main(){
    struct Node *head, *first, *second, *third;

   head = (struct Node*)malloc(sizeof(struct Node));
   first = (struct Node*)malloc(sizeof(struct Node));
   second = (struct Node*)malloc(sizeof(struct Node));
   third = (struct Node*)malloc(sizeof(struct Node));

    head->data=10;
    head->next=first;

    first->data=20;
    first->next=second;

    second->data=30;
    second->next=third;

    third->data=40;
    third->next=NULL;

   printf("original list:\n");
   display(head);

    deleteAtStart(&head);

    printf("after deleting first node:\n");
    display(head);

 return 0;
}
```

```
original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
20 -> 30 -> 40 -> NULL


=== Code Execution Successful ===
```

2.Delection at End

Programiz
C Online Compiler

Programiz PRO >

main.c

Share | Run

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

struct Node{
    int data;
    struct Node *next;
};

void deleteAtEnd(struct Node **head){
    if (*head == NULL){
     printf("list is empty");
     return;
    }
    if((*head) -> next == NULL){
        free(head);
        head = NULL;

    }
    struct Node *temp = *head;
    while(temp->next->next != NULL){
        temp = temp->next;
    }
    free(temp->next);
    temp->next = NULL;
}
    void display(struct Node *head){
        struct Node *temp = head;
        while(temp != NULL){
            printf("%d -> ",temp -> data);
            temp = temp -> next;

        }
        printf("NULL\n");
    }
```

Output | Clear

```
original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
10 -> 20 -> 30 -> NULL


=== Code Execution Successful ===
```


Programiz
C Online Compiler

Programiz PRO >

main.c

Share | Run

```c
            temp = temp -> next;

        }
        printf("NULL\n");
    }

    int main(){
    struct Node *head, *first, *second, *third;

    head = (struct Node*)malloc(sizeof(struct Node));
    first = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

     head->data=10;
     head->next=first;

     first->data=20;
     first->next=second;

     second->data=30;
     second->next=third;

     third->data=40;
     third->next=NULL;

    printf("original list:\n");
    display(head);

     deleteAtEnd(&head);

     printf("after deleting first node:\n");
     display(head);
```

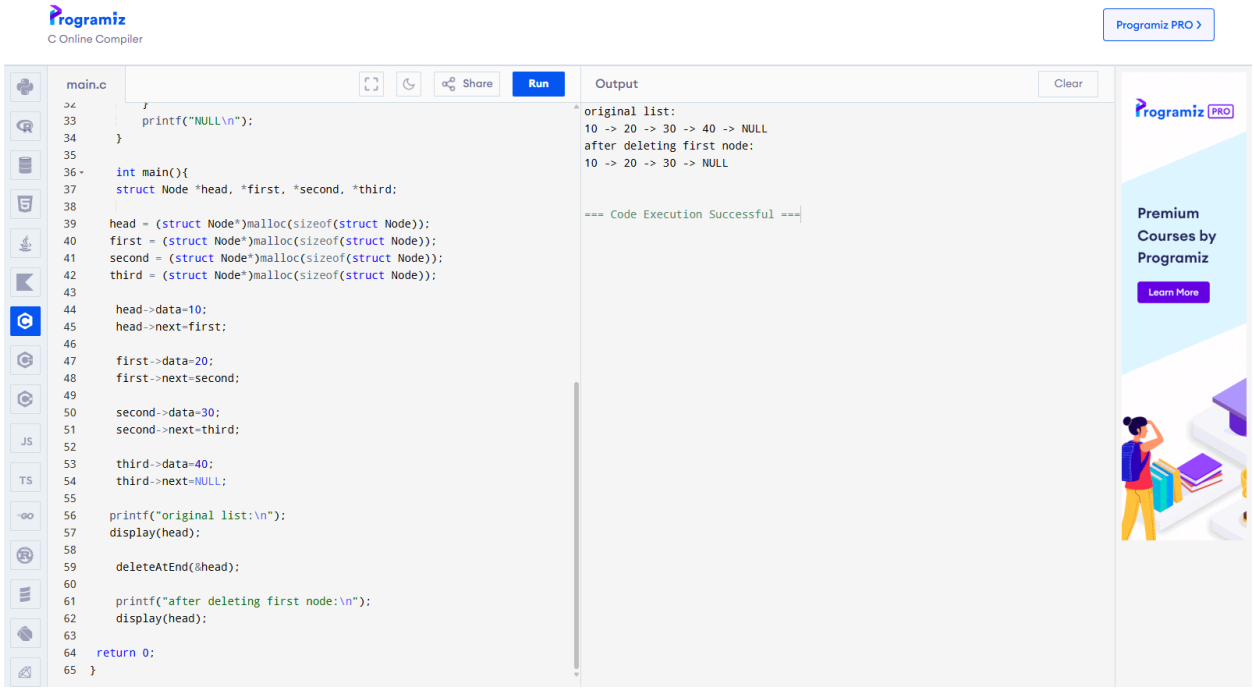
Output | Clear

```
original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
10 -> 20 -> 30 -> NULL


=== Code Execution Successful ===
```

Programiz PRO

Premium Courses by Programiz

Learn More



3.Delection at position

Programiz C Online Compiler

Programiz PRO >

main.c

To exit full screen, press and hold Esc

Clear

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

struct Node{
    int data;
    struct Node *next;
};

void deleteAtPosition(struct Node **head, int position){
    if (*head == NULL || position < 1){
        printf("Invalid operation\n");
        return;
    }

    struct Node *temp = *head;


    if(position == 1){
        *head = temp->next;
        free(temp);
        return;
    }

    for(int i = 1; i < position - 1 && temp->next != NULL; i++){
        temp = temp->next;
    }

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

    struct Node *delNode = temp->next;
```

```
Original list:
10 -> 20 -> 30 -> 40 -> NULL
After deleting node at position 3:
10 -> 20 -> 40 -> NULL


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

Output

Clear

```c
    }

    struct Node *delNode = temp->next;
    temp->next = delNode->next;
    free(delNode);
}

void display(struct Node *head){
    struct Node *temp = head;
    while(temp != NULL){
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main(){
    struct Node *head, *first, *second, *third;

    head = (struct Node*)malloc(sizeof(struct Node));
    first = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    head->data = 10;
    head->next = first;

    first->data = 20;
    first->next = second;

    second->data = 30;
    second->next = third;
```

Original list:
10 -> 20 -> 30 -> 40 -> NULL
After deleting node at position 3:
10 -> 20 -> 40 -> NULL

=== Code Execution Successful ===

Programiz C Online Compiler

Programiz PRO ›

main.c

Share Run

```c
    printf(" NULL\n ");
}

int main(){
    struct Node *head, *first, *second, *third;

    head = (struct Node*)malloc(sizeof(struct Node));
    first = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    head->data = 10;
    head->next = first;

    first->data = 20;
    first->next = second;

    second->data = 30;
    second->next = third;

    third->data = 40;
    third->next = NULL;

    printf("Original list:\n");
    display(head);

    deleteAtPosition(&head, 3);

    printf("After deleting node at position 3:\n");
    display(head);

    return 0;
}

```

Output

Clear

```
Original list:
10 -> 20 -> 30 -> 40 -> NULL
After deleting node at position 3:
10 -> 20 -> 40 -> NULL


=== Code Execution Successful ===
```