

1. Deletion at beginning

The screenshot shows the Programiz C Online Compiler interface. The code in the editor is:

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Node {
4     int data;
5     struct Node *next;
6 };
7 void deleteAtStart(struct Node **head) {
8     if (*head == NULL) {
9         printf("List is empty\n");
10        return;
11    }
12    struct Node *temp = *head;
13    *head = (*head)->next;
14    free(temp);
15 }
16 void display(struct Node *head) {
17     struct Node *temp = head;
18     while (temp != NULL) {
19         printf("%d -> ", temp->data);
20         temp = temp->next;
21     }
22     printf("NULL\n");
23 }
24 int main() {
25     struct Node *head, *first, *second, *third;
26     head = (struct Node*)malloc(sizeof(struct Node));
27     first = (struct Node*)malloc(sizeof(struct Node));
28     second = (struct Node*)malloc(sizeof(struct Node));
29     third = (struct Node*)malloc(sizeof(struct Node));
30     head->data = 20;
31     head->next = first;
32     first->data = 40;
33     first->next = second;
34     second->data = 70;
35     second->next = third;
36     third->data = 65;
37     third->next = NULL;
38     printf("Original List:\n");
39     display(head);
40     deleteAtStart(&head);
41     printf("After deleting first node:\n");
42     display(head);
43     return 0;
44 }
```

The output window shows the execution results:

```
Original List:
20 -> 40 -> 70 -> 65 -> NULL
After deleting first node:
40 -> 70 -> 65 -> NULL

== Code Execution Successful ==
```

The screenshot shows the Programiz C Online Compiler interface. The code in the editor is:

```
main.c
1 struct Node *temp = head;
2 while (temp != NULL) {
3     printf("%d -> ", temp->data);
4     temp = temp->next;
5 }
6 printf("NULL\n");
7
8 int main() {
9     struct Node *head, *first, *second, *third;
10    head = (struct Node*)malloc(sizeof(struct Node));
11    first = (struct Node*)malloc(sizeof(struct Node));
12    second = (struct Node*)malloc(sizeof(struct Node));
13    third = (struct Node*)malloc(sizeof(struct Node));
14    head->data = 20;
15    head->next = first;
16    first->data = 40;
17    first->next = second;
18    second->data = 70;
19    second->next = third;
20    third->data = 65;
21    third->next = NULL;
22    printf("Original List:\n");
23    display(head);
24    deleteAtStart(&head);
25    printf("After deleting first node:\n");
26    display(head);
27    return 0;
28 }
```

The output window shows the execution results:

```
* Original List:
20 -> 40 -> 70 -> 65 -> NULL
After deleting first node:
40 -> 70 -> 65 -> NULL

== Code Execution Successful ==
```

2. Deletion at end

Programiz
C Online Compiler

Capgemini

Rewrite your future.
Join us.

Original List:
60 -> 60 -> 100 -> 90 -> NULL
After deleting last node:
60 -> 60 -> 100 -> NULL

--- Code Execution Successful ---

"strugg
to keep
"Stayin
one st
ahead."

Rewrite your fu
Join us.

Capgemini

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Node {
4     int data;
5     struct Node *next;
6 };
7 void deleteAtEnd(struct Node **head) {
8     if (*head == NULL) {
9         printf("List is empty\n");
10    return;
11 }
12 if ((*head)->next == NULL) {
13     free(*head);
14     *head = NULL;
15     return;
16 }
17 struct Node *temp = *head;
18 while (temp->next != NULL) {
19     temp = temp->next;
20 }
21 free(temp->next);
22 temp->next = NULL;
23 }
24 void display(struct Node *head) {
25     struct Node *temp = head;
26     while (temp != NULL) {
27         printf("%d -> ", temp->data);
28         temp = temp->next;
29     }
30     printf("NULL\n");
31 }
32 int main() {
33     struct Node *head, *first, *second, *third;
34     head = (struct Node*)malloc(sizeof(struct Node));
35     first = (struct Node*)malloc(sizeof(struct Node));
36     second = (struct Node*)malloc(sizeof(struct Node));
37     third = (struct Node*)malloc(sizeof(struct Node));
38     head->data = 60;
39     head->next = first;
40     first->data = 60;
41     first->next = second;
42     second->data = 100;
43     second->next = third;
44     third->data = 90;
45     third->next = NULL;
46     printf("Original List:\n");
47     display(head);
48     deleteAtEnd(&head);
49     printf("After deleting last node:\n");
50     display(head);
51     return 0;
52 }
```

Programiz
C Online Compiler

Capgemini

LIVE YOUR STORIES
CB350 RS

Original List:
60 -> 60 -> 100 -> 90 -> NULL
After deleting last node:
60 -> 60 -> 100 -> NULL

--- Code Execution Successful ---

```
main.c
26     while (temp != NULL) {
27         printf("%d -> ", temp->data);
28         temp = temp->next;
29     }
30     printf("NULL\n");
31 }
32 int main() {
33     struct Node *head, *first, *second, *third;
34     head = (struct Node*)malloc(sizeof(struct Node));
35     first = (struct Node*)malloc(sizeof(struct Node));
36     second = (struct Node*)malloc(sizeof(struct Node));
37     third = (struct Node*)malloc(sizeof(struct Node));
38     head->data = 60;
39     head->next = first;
40     first->data = 60;
41     first->next = second;
42     second->data = 100;
43     second->next = third;
44     third->data = 90;
45     third->next = NULL;
46     printf("Original List:\n");
47     display(head);
48     deleteAtEnd(&head);
49     printf("After deleting last node:\n");
50     display(head);
51     return 0;
52 }
```

3. Deletion at position

Programiz
C Online Compiler

Capgemini

Rewrite your future.
Join us.

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Node {
4     int data;
5     struct Node *next;
6 };
7 void deleteAtPosition(struct Node **head, int position) {
8     if (*head == NULL) {
9         printf("List is empty\n");
10        return;
11    }
12    struct Node *temp = *head;
13    if (position == 1) {
14        *head = temp->next;
15        free(temp);
16        return;
17    }
18    for (int i = 1; temp != NULL && i < position - 1; i++) {
19        temp = temp->next;
20    }
21    if (temp == NULL || temp->next == NULL) {
22        printf("Invalid position\n");
23        return;
24    }
25    struct Node *nodeToDelete = temp->next;
26    temp->next = nodeToDelete->next;
27    free(nodeToDelete);
28 }
```

Output

```
Original List:
40 -> 20 -> 150 -> 180 -> NULL
After deleting node at position 3:
40 -> 20 -> 180 -> NULL

== Code Execution Successful ==
```

Programiz
C Online Compiler

Capgemini

Rewrite your future.
Join us.

```
main.c
28 }
29 void display(struct Node *head) {
30     struct Node *temp = head;
31     while (temp != NULL) {
32         printf("%d -> ", temp->data);
33         temp = temp->next;
34     }
35     printf("NULL\n");
36 }
37 int main() {
38     struct Node *head, *first, *second, *third;
39     head = (struct Node*)malloc(sizeof(struct Node));
40     first = (struct Node*)malloc(sizeof(struct Node));
41     second = (struct Node*)malloc(sizeof(struct Node));
42     third = (struct Node*)malloc(sizeof(struct Node));
43     head->data = 40;
44     head->next = first;
45     first->data = 20;
46     first->next = second;
47     second->data = 150;
48     second->next = third;
49     third->data = 180;
50     third->next = NULL;
51     printf("Original List:\n");
52     display(head);
53     int position = 3;
54     deleteAtPosition(&head, position);
55     printf("After deleting node at position %d:\n", position);
```

Output

```
Original List:
40 -> 20 -> 150 -> 180 -> NULL
After deleting node at position 3:
40 -> 20 -> 180 -> NULL

== Code Execution Successful ==
```

main.c

Run Output Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Node {
4     int data;
5     struct Node *next;
6 };
7 void deleteAtPosition(struct Node **head, int position) {
8     if (*head == NULL) {
9         printf("List is empty\n");
10        return;
11    }
12    struct Node *temp = *head;
13    if (position == 1) {
14        *head = temp->next;
15        free(temp);
16        return;
17    }
18    for (int i = 1; temp != NULL && i < position - 1; i++) {
19        temp = temp->next;
20    }
21    if (temp == NULL || temp->next == NULL) {
22        printf("Invalid position\n");
23        return;
24    }
25    struct Node *nodeToDelete = temp->next;
26    temp->next = nodeToDelete->next;
27    free(nodeToDelete);
28 }
```

Original List:
40 -> 20 -> 150 -> 180 -> NULL
After deleting node at position 3:
40 -> 20 -> 180 -> NULL

== Code Execution Successful ==