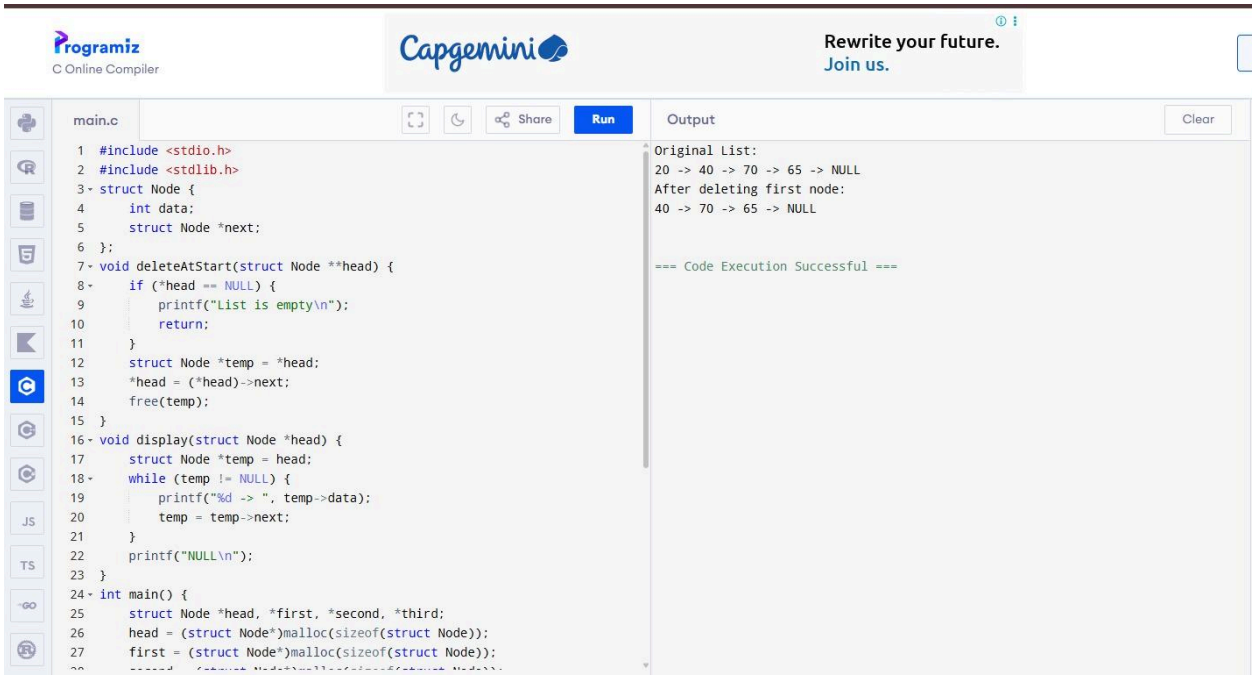
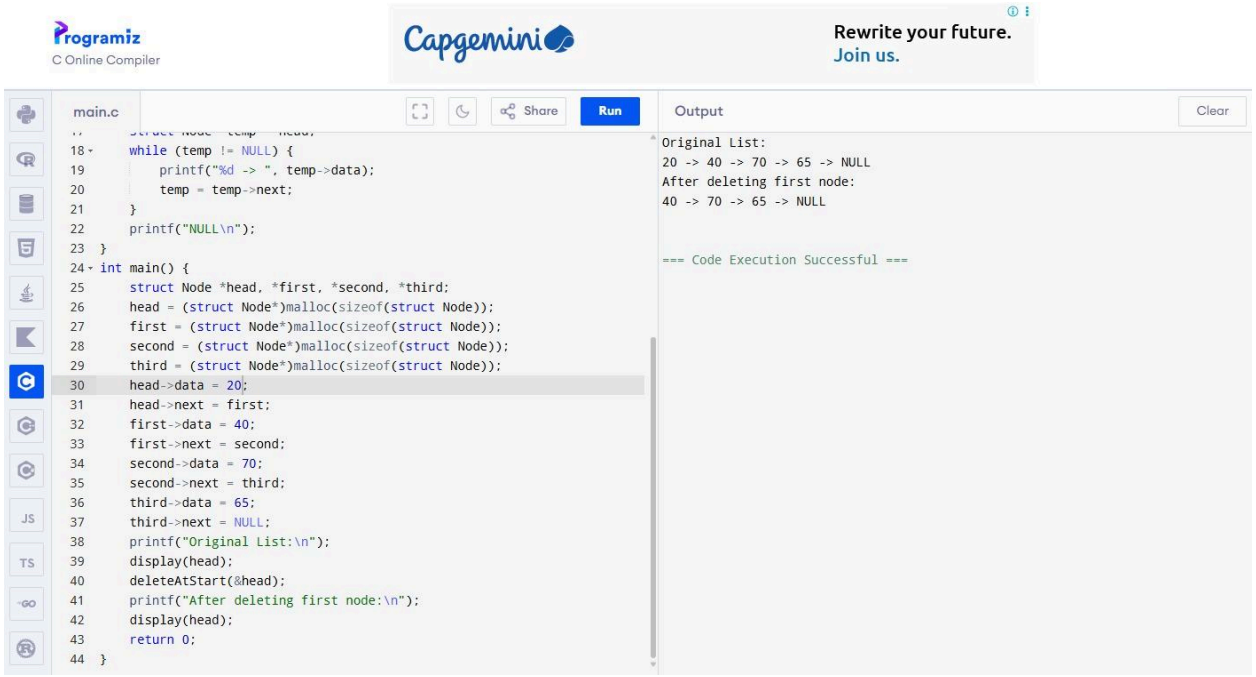
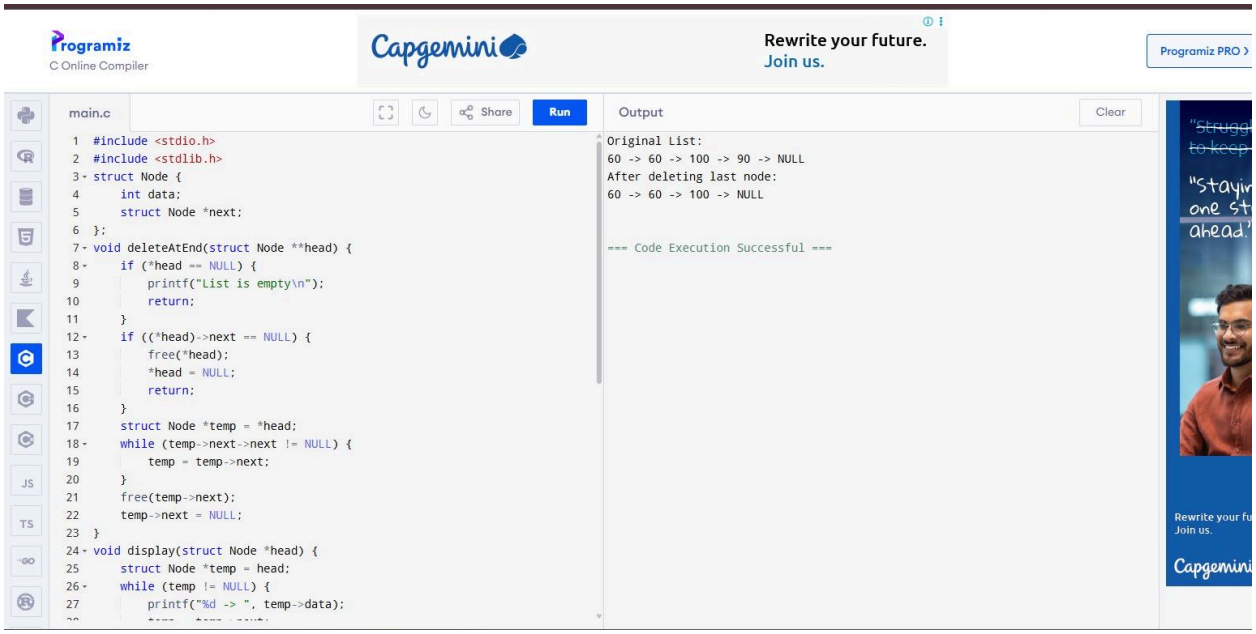
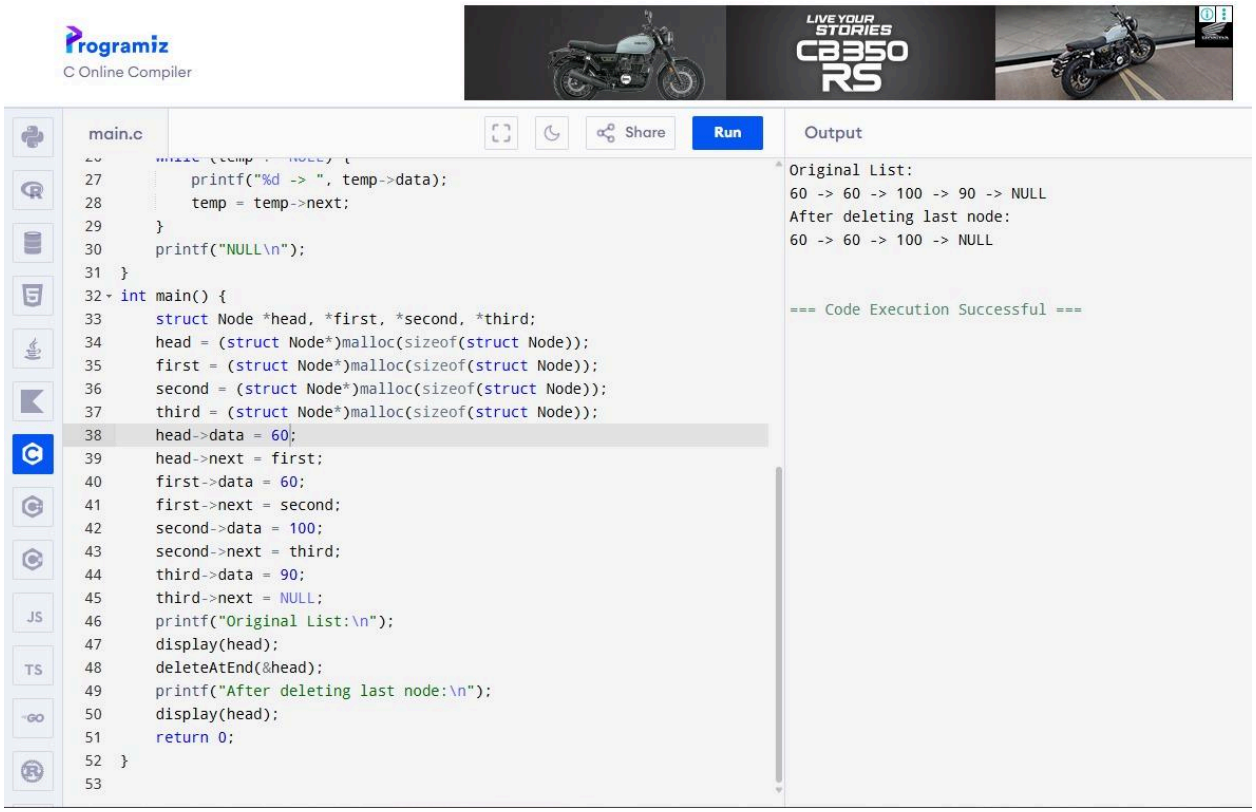
1. Deletion at beginning





2. Deletion at end





3. Deletion at position

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
void deleteAtPosition(struct Node **head, int position) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = *head;
    if (position == 1) {
        *head = temp->next;
        free(temp);
        return;
    }
    for (int i = 1; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
        printf("Invalid position\n");
        return;
    }
    struct Node *nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;
    free(nodeToDelete);
```

Output | Clear

```
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

Share | Run

```c
}
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 = 40;
    head->next = first;
    first->data = 20;
    first->next = second;
    second->data = 150;
    second->next = third;
    third->data = 180;
    third->next = NULL;
    printf("Original List:\n");
    display(head);
    int position = 3;
    deleteAtPosition(&head, position);
    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 ===
```

Programiz
C Online Compiler

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
void deleteAtPosition(struct Node **head, int position) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = *head;
    if (position == 1) {
        *head = temp->next;
        free(temp);
        return;
    }
    for (int i = 1; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
        printf("Invalid position\n");
        return;
    }
    struct Node *nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;
    free(nodeToDelete);
```

Output | Clear

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


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