

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* next;
7 };
8
9 struct Node* head = NULL;
10
11
12 struct Node* createNode(int data) {
13     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
14     newNode->data = data;
15     newNode->next = NULL;
16     return newNode;
17 }
18
19
20 void insertAtBeginning(int data) {
21     struct Node* newNode = createNode(data);
22     newNode->next = head;
23     head = newNode;
24 }
25
26
27 void insertAtEnd(int data) {
28     struct Node* newNode = createNode(data);
29     if (head == NULL) {
30         head = newNode;
31         return;
32     }
33     struct Node* temp = head;
```

RunShare

CopyRunShare

Output

Linked List: 10 -> 15 -> 20 -> 30 -> NULL

Linked List: 15 -> 20 -> 30 -> NULL

Linked List: 15 -> 20 -> NULL

Linked List: 20 -> NULL

==== Code Execution Successful ===

main.c	   Share	Run	Output
			<pre>Linked List: 10 -> 15 -> 20 -> 30 -> NULL Linked List: 15 -> 20 -> 30 -> NULL Linked List: 15 -> 20 -> NULL Linked List: 20 -> NULL ==== Code Execution Successful ===</pre>

```
34+     while (temp->next != NULL) {
35         temp = temp->next;
36     }
37     temp->next = newNode;
38 }

39

40

41+ void insertAtPosition(int data, int position) {
42+     if (position < 1) {
43         printf("Invalid position!\n");
44         return;
45     }
46+     if (position == 1) {
47         insertAtBeginning(data);
48         return;
49     }
50     struct Node* newNode = createNode(data);
51     struct Node* temp = head;
52+     for (int i = 1; temp != NULL && i < position - 1; i++) {
53         temp = temp->next;
54     }
55+     if (temp == NULL) {
56         printf("Position out of range!\n");
57         return;
58     }
59     newNode->next = temp->next;
60     temp->next = newNode;
61 }

62

63

64+ void deleteAtBeginning() {
65+     if (head == NULL) {
66         printf("List is empty!\n");
67         return;
```

main.c	  	Run	Output
<pre>75 void deleteAtEnd() { 76 if (head == NULL) { 77 printf("List is empty!\n"); 78 return; 79 } 80 if (head->next == NULL) { 81 free(head); 82 head = NULL; 83 return; 84 } 85 struct Node* temp = head; 86 while (temp->next->next != NULL) { 87 temp = temp->next; 88 } 89 free(temp->next); 90 temp->next = NULL; 91 } 92 93 94 void deleteByData(int data) { 95 if (head == NULL) { 96 printf("List is empty!\n"); 97 return; 98 } 99 if (head->data == data) { 100 struct Node* temp = head; 101 head = head->next; 102 free(temp); 103 return; 104 } 105 struct Node* temp = head; 106 while (temp->next != NULL && temp->next->data != data) { 107 temp = temp->next; </pre>	  	Linked List: 10 -> 15 -> 20 -> 30 -> NULL Linked List: 15 -> 20 -> 30 -> NULL Linked List: 15 -> 20 -> NULL Linked List: 20 -> NULL ==== Code Execution Successful ===	

main.c



Run

Output

```
118
119 void display() {
120     struct Node* temp = head;
121     if (temp == NULL) {
122         printf("List is empty!\n");
123         return;
124     }
125     printf("Linked List: ");
126     while (temp != NULL) {
127         printf("%d -> ", temp->data);
128         temp = temp->next;
129     }
130     printf("NULL\n");
131 }
132
133 int main() {
134     insertAtBeginning(10);
135     insertAtEnd(20);
136     insertAtEnd(30);
137     insertAtPosition(15, 2);
138     display();
139
140     deleteAtBeginning();
141     display();
142
143     deleteAtEnd();
144     display();
145
146     deleteByData(15);
147     display();
148
149     return 0;
150 }
```

```
Linked List: 10 -> 15 -> 20 -> 30 -> NULL
Linked List: 15 -> 20 -> 30 -> NULL
Linked List: 15 -> 20 -> NULL
Linked List: 20 -> NULL
```

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