C-Program for single linked list:

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

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

struct Node* head = NULL;

void insert(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
}

void delete() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    head = head->next;
    free(temp);
}

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

int main() {
    insert(3);
    insert(7);
    insert(9);
    display();
```

```
    delete();
    display();

    return 0;
}
```
Output : 9 -> 7 -> 3 -> NULL
7 -> 3 -> NULL

C program for Double linked list:

```
#include <stdio.h>
#include <stdlib.h>
    int data;
    struct Node* next;
    struct Node* prev;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
void insertAtBeginning(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
void insertAtEnd(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* temp = *head;
      while (temp->next != NULL) {
```

```
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}
void deleteNode(Node** head, int data) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }
    if ((*head)->data == data) {
        Node* temp = *head;
        *head = (*head)->next;
        if (*head != NULL) {
            (*head)->prev = NULL;
        }
        free(temp);
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL) {
        if (temp->next->data == data) {
            Node* nodeToDelete = temp->next;
            temp->next = temp->next->next;
            if (temp->next != NULL) {
                temp->next->prev = temp;
            }
            free(nodeToDelete);
            return;
        }
        temp = temp->next;
    }
    printf("Node not found\n");
}
void printList(Node* head) {
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}
int main() {
    Node* head = NULL;
```

```
    insertAtBeginning(&head, 10);
    insertAtEnd(&head, 20);
    insertAtBeginning(&head, 5);
    insertAtEnd(&head, 30);
        printf("List: ");
    printList(head);
     deleteNode(&head, 20);
     printf("List after deletion: ");
    printList(head);
    return 0;
}
```

Output : List: 5 10 20 30
List after deletion: 5 10 30

C -Program for Circular linked list:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void insertAtBeginning(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        (*head)->next = *head;
    } else {
        Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        newNode->next = *head;
        temp->next = newNode;
        *head = newNode;
```

```
    }
}
void insertAtEnd(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        (*head)->next = *head;
    } else {
        Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = *head;
    }
}
void deleteNode(Node** head, int data) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }
    if ((*head)->data == data) {
        Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = (*head)->next;
        *head = (*head)->next;
        free(temp);
        return;
    }
    Node* temp = *head;
    while (temp->next != *head) {
        if (temp->next->data == data) {
            Node* nodeToDelete = temp->next;
            temp->next = temp->next->next;
            free(nodeToDelete);
            return;
        }
        temp = temp->next;
    }
    printf("Node not found\n");
}
void printList(Node* head) {
```

```
    Node* temp = head;
    do {
        printf("%d ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("\n");
}
int main() {
    Node* head = NULL;
      insertAtBeginning(&head, 10);
    insertAtEnd(&head, 20);
    insertAtBeginning(&head, 5);
    insertAtEnd(&head, 30);
    printf("List: ");
    printList(head);
        deleteNode(&head, 20);
     printf("List after deletion: ");
    printList(head);
    return 0;
}
```

Output : List: 5 10 20 30
List after deletion: 5 10 30