- ➢ Single linked link:

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
struct Node *createNode(int data) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void insertAtBeginning(struct Node **head, int data) {
    struct Node *newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
}
void insertAtEnd(struct Node **head, int data) {
    struct Node *newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node *last = *head;
    while (last->next != NULL) {
        last = last->next;
    }
    last->next = newNode;
}
void deleteNode(struct Node **head, int key) {
    struct Node *temp = *head, *prev = NULL;
    if (temp != NULL && temp->data == key) {
        *head = temp->next;
        free(temp);
        return;
    }
    while (temp != NULL && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) return;
```

```c
        prev->next = temp->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 = NULL;
        insertAtBeginning(&head, 1);
        insertAtEnd(&head, 2);
        insertAtEnd(&head, 3);
        printf("Linked List:\n");
        display(head);
        deleteNode(&head, 2);
        printf("After deleting 2 from the linked list:\n");
        display(head);
        return 0;
    }
```

Output:

1 -> 2 -> 3 -> NULL
After deleting 2 from the linked list:
1 -> 3 -> NULL

➢ Double linked list:

```c
    #include <stdio.h>
    #include <stdlib.h>
    struct Node {
        int data;
        struct Node *prev;
        struct Node *next;
    };
    struct Node *createNode(int data) {
        struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
        if (newNode == NULL) {
            printf("Memory allocation failed\n");
            exit(1);
        }
        newNode->data = data;
        newNode->prev = NULL;
        newNode->next = NULL;
        return newNode;
    }
    void insertAtBeginning(struct Node **head, int data) {
        struct Node *newNode = createNode(data);
```

```c
      newNode->next = *head;
      if (*head != NULL) {
         (*head)->prev = newNode;
      }
      *head = newNode;
}
void insertAtEnd(struct Node **head, int data) {
      struct Node *newNode = createNode(data);
      if (*head == NULL) {
         *head = newNode;
         return;
      }
      struct Node *last = *head;
      while (last->next != NULL) {
         last = last->next;
      }
      last->next = newNode;
      newNode->prev = last;
}
void deleteNode(struct Node **head, int key) {
      if (*head == NULL) return;

      struct Node *temp = *head;
      if (temp != NULL && temp->data == key) {
         *head = temp->next;
         if (*head != NULL)
            (*head)->prev = NULL;
         free(temp);
         return;
      }

      while (temp != NULL && temp->data != key) {
         temp = temp->next;
      }

      if (temp == NULL) return;

      if (temp->prev != NULL)
         temp->prev->next = temp->next;
      if (temp->next != NULL)
         temp->next->prev = temp->prev;

      free(temp);
}
void display(struct Node *head) {
      struct Node *temp = head;
      printf("NULL <-> ");
      while (temp != NULL) {
```

```c
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    struct Node *head = NULL;
    insertAtBeginning(&head, 1);
    insertAtEnd(&head, 2);
    insertAtEnd(&head, 3);
    printf("Double Linked List:\n");
    display(head);
    deleteNode(&head, 2);
    printf("After deleting 2 from the double linked list:\n");
    display(head);
    return 0;
}
```

NULL <-> 1 <-> 2 <-> 3 <-> NULL
After deleting 2 from the double linked list:
NULL <-> 1 <-> 3 <-> NULL

➤ Circular linked list:

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
struct Node *createNode(int data) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void insertAtBeginning(struct Node **head, int data) {
    struct Node *newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        newNode->next = *head;
    } else {
        struct Node *last = *head;
        while (last->next != *head) {
            last = last->next;
        }
```

```c
            newNode->next = *head;
            last->next = newNode;
            *head = newNode;
        }
    }
    void insertAtEnd(struct Node **head, int data) {
        struct Node *newNode = createNode(data);
        if (*head == NULL) {
            *head = newNode;
            newNode->next = *head;
        } else {
            struct Node *last = *head;
            while (last->next != *head) {
                last = last->next;
            }
            last->next = newNode;
            newNode->next = *head;
        }
    }
    void deleteNode(struct Node **head, int key) {
        if (*head == NULL) return;
        struct Node *temp = *head, *prev = NULL;
        while (temp->data != key) {
            if (temp->next == *head) {
                printf("Key not found in the list\n");
                return;
            }
            prev = temp;
            temp = temp->next;
        }
        if (temp->next == *head && prev == NULL) {
            *head = NULL;
            free(temp);
            return;
        }
        if (temp == *head) {
            prev = *head;
            while (prev->next != *head) {
                prev = prev->next;
            }
            *head = (*head)->next;
            prev->next = *head;
            free(temp);
        } else if (temp->next == *head) {
            prev->next = *head;
            free(temp);
        } else {
            prev->next = temp->next;
```

```c
        free(temp);
    }
}
void display(struct Node *head) {
    struct Node *temp = head;
    printf("HEAD -> ");
    if (head != NULL) {
        do {
            printf("%d -> ", temp->data);
            temp = temp->next;
        } while (temp != head);
        printf("HEAD\n");
    } else {
        printf("List is empty.\n");
    }
}
int main() {
    struct Node *head = NULL;
    insertAtBeginning(&head, 1);
    insertAtEnd(&head, 2);
    insertAtEnd(&head, 3);
    printf("Circular Linked List:\n");
    display(head);
    deleteNode(&head, 2);
    printf("After deleting 2 from the circular linked list:\n");
    display(head);
    return 0;
}
```
Output:

Circular Linked List:

HEAD -> 1 -> 2 -> 3 -> HEAD

After deleting 2 from the circular linked list:

HEAD -> 1 -> 3 -> HEAD