—> linked list (insertion)

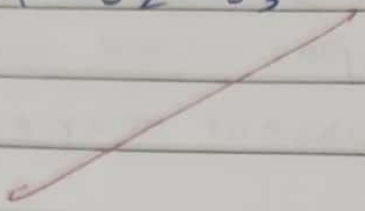
```
#include <stdio.h>
#include <stdlib.h>

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

struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(
                                    struct Node));
    newNode -> data = value;
    newNode -> next = NULL;
    return newNode;
}

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

struct Node* insertAtBeg(struct Node* head, int value){
    struct Node* newNode = createNode(value);
    newNode -> next = head;
    return newNode;
```

```
void insertatpos(struct Node* head, int value, int pos) {
    struct Node* newNode = createNode(value);
    struct Node* current = head;
    for(int i = 1; i<pos-1 && current != NULL; i++) {
        current = current->next;
    }
    if(current != NULL) {
        newNode->next = current->next;
        current->next = newNode;
    } else {
        printf("Invalid position\n");
    }
}

void insertatend(struct Node* head, int value) {
    struct Node* newNode = createNode(value);
    struct Node* current = head;
    while(current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;

int main() {
    struct Node* head = NULL;
    head = insertatBeg(head, 1);
    insertAtEnd(head, 3);
    insertatPos(head, 2, 2);
    displayList(head);
    return 0;
}
```

O/P Linked list : 1 ⟶ NULL

linked list : 1 → 2 → NULL

Linked list : 1 → 2 → 3 ⟶ NULL



—> Linked list (Delehon)

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

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

Stouct Node* createNode (int data) {
  stouct Node * newnode = (stouct Node*) malloc (sizeof(
                                    stouct Node));
  newNode->data = data;
  newNode->next = NULL;
  return new Node;
}

Stouct Node* deletefiost (stouct Node* head) {
    if (head == NULL) {
        printf ("List Empty");
        rehon NULL;
    }
    Stouct Node* temp = head;
    head = head->next;
    free (temp);
    return head;
}
```

```
Struct Node* deleteElement (struct Node*head, int value){
    ~~if(~~
    struct Node* current = head;
    struct Node* p = NULL;
    while (current != NULL && current -> data != value){
        p = current;
        current = current -> next;
    }

    if (current != NULL){
        if (p == NULL){
            head = head -> next;
    } else {
            p -> next = current -> next;
        }
        free (current);
    } else {
        printf("Element %d not found", value);
    }
    return head;
}

Struct Node* deletelast (struct Node* head) {
    ~~if (head ==~~
    if (head -> next == NULL){
        free(head);
        return NULL;
    }
Struct Node* current = head;
struct Node* p = NULL;
```

```
while (current->next != NULL) {
    p = current;
    current = current -> next;
}
    free (current);
    p -> next = NULL;
    return head;
}
}

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

int main () {
    struct Node* head = NULL;
    head = delete first (head);
    display_list (head);
    head = delete Element (head, 2);
    display_list (head);
    head = delete last (head);
    display list (head);
    return 0;
}
```

O/P

Linked list: 1 -> 2 -> 3 -> NULL

Linked list: 2 -> 3 -> NULL

Linked list: 3 -> NULL

Linked list: NULL

22/01/24