Program

#include<stdio.h>

#include<stdlib.h>

struct node

{

    struct node \*prev;

    int data;

    struct node \*next;

};

struct node \*temp1=NULL,\*temp2=NULL, \*head=NULL, \*newnode=NULL;

void insertion\_big();

void insertion\_data();

void insertion\_end();

void delete\_big();

void delete\_data();

void delete\_end();

void traversal();

void main(){

int a,b,c;

while(1){

menu:  //back to previous menu

printf("\n----------------LINKED LIST MENU----------------------\n");

printf("1. Traversal/Display\n");

printf("2. Insertion\n");

printf("3. Deletion\n");

printf("4. searching\n");

printf("0. Exit\n");

printf("enter the operation do you to perform\n");

scanf("%d",&a);

switch(a){

    case 1: traversal();

            break;

    case 2: while(1){

            printf(" \n--------------------INSERTION MENU-------------------\n");// insertion

            printf("1. Traversal \n");

            printf("2. Insertion at the beginning\n");

            printf("3. Insertion at the end\n");

            printf("4. Insertion after a particular element\n");

            printf("9. previous \n");

            printf("0. exit\n");

            printf("enter the operation do you to perform\n");

            scanf("%d",&b);

            switch(b){

                case 1: traversal();

                        break;

                case 2: insertion\_big();

                        break;

                case 3: insertion\_end();

                        break;

                case 4: insertion\_data();

                        break;

                case 9: goto menu;

                case 0: exit(1);

                default: printf("invalid option\n");

                        break;

            }

    }

            break;

    case 3: while(1){

            printf(" \n--------------------DELETION MENU-------------------\n"); //deletion

            printf("1. Traversal \n");

            printf("2. Deletion at the beginning\n");

            printf("3. Deletion at the end\n");

            printf("4. Deletion a particular element\n");

            printf("9. previous \n");

            printf("0. exit\n");

            printf("enter the operation do you to perform\n");

            scanf("%d" &c);

            switch(c){

                case 1: traversal();

                        break;

                case 2: delete\_big();

                        break;

                case 3: delete\_end();

                        break;

                case 4: delete\_data();

                        break;

                case 9: goto menu;

                case 0: exit(1);

                default: printf("invalid option\n");

                        break;

            }

    }

            break;

    case 4: while(1){

                printf("\n --------------------SEARCHING MENU-------------------\n");

            printf("1. Traversal \n");

            printf("2. searching with postion\n");

            printf("3. searching with data\n");

            printf("9. previous \n");

            printf("0. exit \n");

            printf("enter the operation do you to perform\n");

            scanf("%d",&c);

            switch(c){

                case 1: traversal();

                        break;

                case 2: search\_pos();

                        break;

                case 3: search\_data();

                        break;

                case 9: goto menu;

                case 0: exit(1);

                default: printf("invalid option\n");

                        break;

            }

    }

            break;

    case 0: exit(1);

    default:printf("Invalid option\n");

            break;

}

}

}

void traversal(){

    struct node \*temp1;

    temp1=head;

    if(head==NULL){

        printf("\nNO Element Found\n");

    }

    else{

        while(temp1!=NULL){

            printf("%d\t",temp1->data);

            temp1=temp1->next;

        }

    }

}

void search\_pos(){

    int pos,i=1,flag=0;

    struct node \*temp=NULL;

    temp=head;

    printf("enter the position\n");

    scanf("%d",&pos);

    while(temp!=NULL){

        if(i==pos){

            flag=1;

            printf("%d\n",temp->data);

            break;

        }

        temp=temp->next;

        i++;

    }

    if(!flag){

        printf("Element not found\n");

    }

}

void search\_data(){

    int data,i=0,flag=0;

    struct node \*temp=NULL;

    temp=head;

    printf("enter the data\n");

    scanf("%d",&data);

    while(temp!=NULL){

        if(temp->data==data){

            flag=1;

            printf("%d\n",temp->data);

            printf("found in %d position\n",i+1);

            break;

        }

        temp=temp->next;

        i++;

    }

    if(!flag){

        printf("not found\n");

    }

}

void insertion\_big(){

    struct node \*temp1=head;

    newnode=(struct node \*)malloc(sizeof(struct node));

    printf("enter the new element\n");

    scanf("%d",&newnode->data);

    if(head==NULL){

        head=newnode;

        head->prev=NULL;

        head->next=NULL;

    }

    else{

        newnode->next=head;

        head=newnode;

        newnode->prev=NULL;

        newnode->next->prev=newnode;

    }

    temp1=head;

    while(temp1!=NULL){

        printf("%d\t",temp1->data);

        temp1=temp1->next;

    }

}

void insertion\_end(){

    struct node \*temp1=head,\*temp2;

    newnode=(struct node \*)malloc(sizeof(struct node));

    printf("enter the new element\n");

    scanf("%d",&newnode->data);

    newnode->next=NULL;

    if(head==NULL){

        head=newnode;

        head->prev=NULL;

        head->next=NULL;

    }

    else{

        while(temp1->next!=NULL){

            temp1=temp1->next;

        }

        newnode->prev=temp1;

        temp1->next=newnode;

        temp1=head;

        while(temp1!=NULL){

            printf("%d\t",temp1->data);

            temp1=temp1->next;

            }

    }

}

void insertion\_data(){

    struct node \*temp1=head;

    int d;

    newnode=(struct node \*) malloc(sizeof(struct node));

    printf("enter the element to be inserted\n");

    scanf("%d",&newnode->data);

    printf("The data added to the front of the current data");

    scanf("%d",&d);

    if(head==NULL){

        printf("insert element in beginning or end");

        return 0;

    }

    else{

        while(temp1->next!=NULL){

            if(temp1->data==d){

                temp1->next->prev=newnode;

                newnode->prev=temp1;

                newnode->next=temp1->next;

                temp1->next=newnode;

            }

            temp1=temp1->next;

        }

    }

    temp1=head;

    while(temp1!=NULL){

        printf("%d\t",temp1->data);

        temp1=temp1->next;

        }

    }

void delete\_big(){

    if(head==NULL){

        printf("deletion is not possible");

        return 0;

    }

    head->next->prev=NULL;

    head=head->next;

    temp2=head;

    while(temp1!=NULL){

        printf("%d\t",temp1->data);

        temp1=temp1->next;

        }

}

void delete\_end(){

struct node \*temp1=head;

if(head==NULL){

    printf("deletion is not possible\n");

    return 0;

}

else{

    while(temp1->next->next!=NULL){

        temp1=temp1->next;

    }

    temp1->next=NULL;

    temp1=head;

    while(temp1!=NULL){

    printf("%d\t",temp1->data);

    temp1=temp1->next;

    }

}

}

void delete\_data()

{

    struct node \*temp1, \*temp2;

    int val;

    printf("Enter the value");

    scanf("%d",&val);

    temp2 = head;

   if(head==NULL){

        printf("deletion is not possible\n");

        return 0;

    }

    else if(head->data==val){

        delete\_big();

    }

    else{

        temp1=head;

        while(temp1->next->data!=val){

            temp1=temp1->next;

        }

        if(temp1->next->next==NULL){

            temp1->next=NULL;

            temp1=head;

            return 0;

        }

        temp2=temp1->next;

        temp1->next=temp2->next;

        temp2->next->prev=temp1;

        free(temp2);

    }

    temp1=head;

    while(temp1!=NULL){

        printf("%d\t",temp1->data);

        temp1=temp1->next;

        }

}