//implementation of single linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};

    struct node *head=NULL,*last=NULL;
    void create();
    void insert();
    void delet();
    void display();
    void search();

void create()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    int n;
    printf("\nEnter an Element:");
    scanf("%d",&n);
    temp->data=n;
    temp->next=NULL;
    if(head==NULL)
    {
        head=temp;
        last=head;
    }

    else
    {
        last->next=temp;
        last=temp;
    }
}
void insert()
{
    struct node *prev,*cur,*temp;
    prev=NULL;
    cur=head;
    int count=1,pos,ch,n;
    temp=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter an Element:");
    scanf("%d",&n);
    temp->data=n;
    temp->next=NULL;
    printf("\nINSERT AS\n1:FIRSTNODE\n2:LASTNODE\n3:IN BETWEEN
FIRST&LAST NODES");
    printf("\nEnter Your Choice:");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1:
        temp->next=head;
        head=temp;
        break;
    case 2:
        last->next=temp;
        last=temp;
        break;
    case 3:
        printf("\nEnter the Position to Insert:");
        scanf("%d",&pos);
        printf("pos:%d,count=%d",pos,count);
        while(count!=pos)
        {
            prev=cur;
            cur=cur->next;
            count++;
        }
        if(count==pos)
        {
            prev->next=temp;
            temp->next=cur;
        }
        else
        {
            printf("\nNot Able to Insert");
        }
        break;

    }
}
void delet()
{
    struct node *prev=NULL,*cur=head;
    int count=1,pos,ch;
    printf("\nDELETE\n1:FIRSTNODE\n2:LASTNODE\n3:IN BETWEEN
FIRST&LAST NODES");
    printf("\nEnter Your Choice:");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1:
        if(head!=NULL)
        {
            printf("Deleted Element is %d",head->data);
            head=head->next;
        }
        else
            printf("Not Able to Delete");
        break;
```

```
case 2:
      if(head==NULL)
      {
      printf("Not Able to Delete");
      }
      else
      {

    while(cur!=last)
    {
        prev=cur;
        cur=cur->next;
    }
    if(cur==last)
    {
        printf("\nDeleted Element is:%d ",cur->data);
        prev->next=NULL;
        last=prev;
    }
    }
    break;
case 3:
    printf("\nEnter the Position of Deletion:");
   scanf("%d",&pos);
      if(head==NULL)
      {
      printf("\nNot Able to Delete");
      }
      else
      {
      while(count!=pos)
      {
                 prev=cur;
                cur=cur->next;
                count++;
```

```
                  }
          if(count==pos)
          {
                          printf("\nDeleted Element is:%d ",cur->data);
                  prev->next=cur->next;
          }
        }
        break;
    }
}
void display()
{
    struct node *temp=head;
    if(temp==NULL)
    {
        printf("\nList is Empty");
    }
    while(temp!=NULL)
    {
        printf("%d",temp->data);
        printf("-->");
        temp=temp->next;
    }
    printf("NULL\n");
}
void search()
{
    int value,pos=0;
    int flag=0;
    if(head==NULL)
    {
        printf("List is Empty");
        return;
    }
    printf("Enter the Value to be Searched:");
    scanf("%d",&value);
    struct node *temp;
    temp=head;
    while(temp!=NULL)
    {
        pos++;
        if(temp->data==value)
        {
            flag=1;
            printf("Element %d is Found at %d Position",value,pos);
            return;
        }
        temp=temp->next;
    }
    if(!flag)
    {
        printf("Element %d not Found in the List",value);
    }
}
int main()
{
    int ch;
    while(1)
    {
        printf("\n**** MENU ****");

printf("\n1:CREATE\n2:INSERT\n3:DELETE\n4:SEARCH\n5:DISPLAY\n6:EXI
T\n");
        printf("\nEnter Your Choice:");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1:
            create();
            break;
        case 2:
            insert();
            break;
        case 3:
            delet();
            break;
        case 4:
            search();
            break;
        case 5:
            display();
            break;
        case 6:
            return 0;
        default:
            printf("\n Invalid choice: Choose correct one");
            break;
        }
    }
    return 0;
}
```