LINKED LIST

Program to create a SLL and perform all insertion and deletion cases

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *link;
};
struct node *root=NULL;
void create();
int display(struct node*);
void addat_begin();
void append();
int length(struct node*);
void insert_after();
void delete_beg();
void delete_specified();
void main()
{
    int op,len;
    while(1)
    {
        printf("***menu**\n");
        printf("1.create\n");
        printf("2.display\n");
        printf("3.addat_begin\n");
        printf("4.append\n");
        printf("5.length\n");
        printf("6.insert_after\n");
        printf("7.delete_begin\n");
        printf("8.delete_specified\n");
        printf("9.exit\n");
        printf("enter option\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                    create();
                    break;
            case 2:
                    display(root);
                    break;
            case 3:
                    addat_begin();
                    break;
                case 4:
                    append();
                    break;
                case 5:
                    printf("length:%d\n",length(root));
                    break;
                case 6:
                    insert_after();
                    break;
                case 7:
                  delete_beg();
                  break;
                case 8:
                    delete_specified();
                    break;
                case 9:
                    exit(1);
                    break;
            }
        }
    }
    void create()
    {
        struct node *temp,*p;
        int num;
        printf("enter -1 to exit\n");
        printf("enter data\n");
        scanf("%d",&num);
        while(num!=-1)
        {
            temp=(struct node*)malloc(sizeof(struct node));
            temp->data=num;
            temp->link=NULL;
            if(root==NULL)
            {
                root=temp;
            }
            else
            {
                p=root;
                while(p->link!=NULL)
                {
                    p=p->link;
                }
                p->link=temp;

            }
            printf("enter data\n");
            scanf("%d",&num);
        }

    }
    int display(struct node *temp)
    {

        if(temp==NULL)
        {
            printf("no nodes to display\n");
        }
        else
        {
            printf("list elements\n");
            while(temp!=NULL)
            {
                printf("%d\n",temp->data);
                temp=temp->link;
            }
        }

    }
    void addat_begin()
    {
        struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        printf("enter temp data\n");
        scanf("%d",&temp->data);
        temp->link=NULL;
        if(root==NULL)
        {
            root=temp;
        }
        else
        {
            temp->link=root;
            root=temp;
        }

    }
    void append()
    {
        struct node *temp,*p;
        temp=(struct node*)malloc(sizeof(struct node));
        printf("enter temp data\n");
        scanf("%d",&temp->data);
        temp->link=NULL;
        p=root;
        while(p->link!=NULL)
        {
            p=p->link;
        }
        p->link=temp;


    }
    int length(struct node *temp)
    {
        int c=0;
        while(temp!=NULL)
        {
            c++;
            temp=temp->link;
        }
        return c;
    }
    void insert_after()
    {
        int loc,i=1;
        printf("enter location\n");
        scanf("%d",&loc);
        if(loc>length(root))
        {
            printf("invalid location\n");

        }
        else
        {
            struct node *p=root;
            struct node *temp;
            while(i<loc)
            {
                p=p->link;
                i++;
            }
            temp=(struct node*)malloc(sizeof(struct node));
            printf("enter data\n");
            scanf("%d",&temp->data);
            temp->link=NULL;
            temp->link=p->link;
            p->link=temp;
        }
    }
    void delete_beg()
    {
        struct node *temp=root;
        root=temp->link;
        temp->link=NULL;
        free(temp);
    }
    void delete_specified()
    {
        int loc,i=1;
        printf("enter location\n");
        scanf("%d",&loc);
        if(loc>length(root))
        {
            printf("invalid location\n");
        }
        else
        {
            struct node *p,*q;
            p=root;
            while(i<loc-1)
            {
                p=p->link;
                i++;
            }
            q=p->link;
            p->link=q->link;
            q->link=NULL;
            free(q);

        }
    }
```

Output:

```
***menu**
1.create
2.display
3.addat_begin
4.append
5.length
6.insert_after
7.delete_begin
8.delete_specified
9.exit
enter option
1
enter -1 to exit
enter data
10
enter data
```

```
20
enter data
30
enter data
-1
***menu**
1.create
2.display
3.addat_begin
4.append
5.length
6.insert_after
7.delete_begin
8.delete_specified
9.exit
enter option
2
list elements
10
20
30
***menu**
1.create
2.display
3.addat_begin
4.append
5.length
6.insert_after
7.delete_begin
8.delete_specified
9.exit
enter option
9
```

Program to create a DLL and perform all insertion and deletion cases

```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
    struct node *left;
    int data;
    struct node *right;
};
struct node *root=NULL;
void create();
int display();
```

```
void insert_big();
void insert_end();
int length();
void add_after();
void delete_big();
void delete_specified();
void delete_end();
void main()
{
    int op;
    printf("***menu***\n");
    printf("1.create\n");
    printf("2.display\n");
    printf("3.insert_big\n");
    printf("4.insert_end\n");
    printf("5.length\n");
    printf("6.add_after\n");
    printf("7.delete_big\n");
    printf("8.delete_specified\n");
    printf("9.delete_end\n");
    Printf( “10.exit\n” );
    while(1)
    {

        printf("enter option\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                    create();
                    break;
            case 2:
                    display();
                    break;
            case 3:
                    insert_big();
                    break;
            case 4:
                    insert_end();
                    break;
            case 5:
                    printf("length:%d nodes\n",length());
                    break;
            case 6:
                    add_after();
                    break;
            case 7:
                    delete_big();
```

```
                break;
            case 8:
                delete_specified();
                break;
            case 9:
                delete_end();
                break;
            case 10:
                exit(1);
                break;



        }
    }
}
void create()
{
    struct node *temp,*p;
    int num;
    printf("enter -1 to exit\n");
    printf("enter data\n");
    scanf("%d",&num);
    while(num!=-1)
    {
        temp=(struct node*)malloc(sizeof(struct node));
        temp->left=NULL;
        temp->data=num;
        temp->right=NULL;
        if(root==NULL)
        {
            root=temp;
        }
        else
        {
            p=root;
            while(p->right!=NULL)
            {
                p=p->right;
            }
            temp->left=p;
            p->right=temp;
        }
        printf("enter data\n");
        scanf("%d",&num);
    }
}
int display()
```

```
{
    struct node *temp=root;
    if(temp==NULL)
    {
        printf("no nodes in list\n");

    }
    else
    {
        printf("elements in list\n");
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->right;
        }
    }
}
void insert_big()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp->left=NULL;
    printf("enter new data\n");
    scanf("%d",&temp->data);
    temp->right=NULL;
    if(root==NULL)
    {
        root=temp;
    }
    temp->right=root;
    root->left=temp;
    root=temp;
}
void insert_end()
{
    struct node *temp,*p;
    temp=(struct node*)malloc(sizeof(struct node));
    temp->left=NULL;
    printf("enter new data\n");
    scanf("%d",&temp->data);
    temp->right=NULL;
    p=root;
    while(p->right!=NULL)
    {
        p=p->right;
    }
    p->right=temp;
    temp->left=p;
}
int length()
{
    struct node *temp=root;
    int c=0;
    while(temp!=NULL)
    {
        c++;
        temp=temp->right;

    }
    return c;
}
void add_after()
{
    struct node *temp,*p;
    int loc,len,i=1;
    printf("enter location\n");
    scanf("%d",&loc);
    len=length();
    if(loc>len)
    {
        printf("invalid location\n");
        printf("list contain %d nodes",len);
    }
    else
    {
        temp=(struct node*)malloc(sizeof(struct node));
        printf("enter node data\n");
        scanf("%d",&temp->data);
        temp->left=NULL;
        temp->right=NULL;
        p=root;
        while(i<loc)
        {
            p=p->right;
            i++;

        }
        temp->right=p->right;
        p->right->left=temp;
        temp->left=p;
        p->right=temp;
    }

}
void delete_big()
```

```
{
    struct node *temp=root;
    root=temp->right;
    root->left=NULL;
    free(temp);
}
void delete_specified()
{
    struct node *q,*p;
    int loc,len,i=1;
    printf("enter location\n");
    scanf("%d",&loc);
    len=length();
    if(loc>len)
    {
        printf("invalid location\n");
        printf("list contain %d nodes",len);
    }
    else
    {
        p=root;
        while(i<loc-1)
        {
            p=p->right;
            i++;
        }
        q=p->right;
        p->right=q->right;
        q->right->left=p;
    }
}
void delete_end()
{
    struct node *temp=root;
    while(temp->right!=NULL)
    {
        temp=temp->right;

    }
    temp->left->right=NULL;
}
```
Output:

```
***menu***
1.create
2.display
3.insert_big
4.insert_end
5.length
```

```
6.add_after
7.delete_big
8.delete_specified
9.delete_end
10.exit
enter option
1
enter -1 to exit
enter data
10
enter data
20
enter data
30
enter data
40
enter data
-1
enter option
2
elements in list
10
20
30
40
enter option
5
length:4 nodes
enter option
10
```