Singly Linked List Program in C with all cases

```
#include <stdio.h>
#include<conio.h>
void create();
void display();
void begin_insert();
void end_insert();
void pos_insert();
void begin_delete();
void end_delete();
void pos_delete();
 struct node
{
        int info;
        struct node *next;
};
struct node *start=NULL;
int main()
{
        int choice;
        while(1){

                printf("\n MENU  ");
printf("\n 1.Create");
printf("\n 2.Display");
printf("\n 3.Insert at the beginning");
printf("\n 4.Insert at the end");
printf("\n 5.Insert at specified position");
printf("\n 6.Delete from beginning ");
printf("\n 7.Delete from the end ");
printf("\n 8.Delete from specified position");
printf("\n 9.Exit ");
printf("\n------------------------------------------n");
printf("\nEnter your choice:");
scanf("%d",&choice);
switch(choice)
{
        case 1:
                        create();
                        break;
        case 2:
                        display();
                        break;
        case 3:
                        begin_insert();
                        break;
        case 4:
                        end_insert();
                        break;
        case 5:
                        pos_insert();
                        break;
        case 6:
                        begin_delete();
                        break;
        case 7:
                        end_delete();
                        break;
        case 8:
                        pos_delete();
                        break;


        case 9:
                        exit(0);
                        break;


        default:
                        printf("n Wrong Choice:n");
                        break;
            }
        }
```

```
	return 0;
}
void create()
{
	struct node *temp,*ptr;
	temp=(struct node *)malloc(sizeof(struct node));
	if(temp==NULL)
	{
		printf("nOut of Memory Space:n");
		exit(0);
	}
	printf("Enter the data value for the node:");
	scanf("%d",&temp->info);
	temp->next=NULL;
	if(start==NULL)
	{
		start=temp;
	}
	else
	{
		ptr=start;
		while(ptr->next!=NULL)
		{
			ptr=ptr->next;
```

```
                }
                ptr->next=temp;
        }
}
void display()
{
        struct node *ptr;
        if(start==NULL)
        {
                printf("nList is empty:n");
                return;
        }
        else
        {
                ptr=start;
                printf("The List elements are:");
                while(ptr!=NULL)
                {
                        printf("%dt",ptr->info );
                        ptr=ptr->next ;
                }
        }
}
void begin_insert()
```

```
{
        struct node *temp;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
                printf("nOut of Memory Space:n");
                return;
        }
        printf("Enter the data value for the node:" );
        scanf("%d",&temp->info);
        temp->next =NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                temp->next=start;
                start=temp;
        }
}
void end_insert()
{
        struct node *temp,*ptr;
}
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
                printf("nOut of Memory Space:n");
                return;
        }
        printf("Enter the data value for the node:" );
        scanf("%d",&temp->info );
        temp->next =NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                ptr=start;
                while(ptr->next !=NULL)
                {
                        ptr=ptr->next ;
                }
                ptr->next =temp;
        }
}
void pos_insert()
```

```
{
    struct node *ptr,*temp;
    int i,pos;
    temp=(struct node *)malloc(sizeof(struct node));
    if(temp==NULL)
    {
        printf("nOut of Memory Space:n");
        return;
    }
    printf("Enter the position for the new node to be inserted:");
    scanf("%d",&pos);
    printf("Enter the data value of the node:");
    scanf("%d",&temp->info) ;

    temp->next=NULL;
    if(pos==0)
    {
        temp->next=start;
        start=temp;
    }
    else
    {
        for(i=0,ptr=start;i<pos-1;i++) { ptr=ptr->next;
            if(ptr==NULL)
```

```
                {
                    printf("Position not found:");
                    return;
                }
            }
            temp->next =ptr->next ;
            ptr->next=temp;
        }
}
void begin_delete()
{
        struct node *ptr;
        if(ptr==NULL)
        {
            printf("nList is Empty:n");
            return;
        }
        else
        {
            ptr=start;
            start=start->next ;
            printf("The deleted element is :%d",ptr->info);
            free(ptr);
        }
```

```
}
void end_delete()
{
        struct node *temp,*ptr;
        if(start==NULL)
        {
                printf("List is Empty:");
                exit(0);
        }
        else if(start->next ==NULL)
        {
                ptr=start;
                start=NULL;
                printf("The deleted element is:%d",ptr->info);
                free(ptr);
        }
        else
        {
                ptr=start;
                while(ptr->next!=NULL)
                {
                        temp=ptr;
                        ptr=ptr->next;
                }
        temp->next=NULL;
        printf("The deleted element is:%d",ptr->info);
        free(ptr);
    }
}
void pos_delete()
{
    int i,pos;
    struct node *temp,*ptr;
    if(start==NULL)
    {
        printf("nThe List is Empty:n");
        exit(0);
    }
    else
    {
        printf("Enter the position of the node to be deleted:");
        scanf("%d",&pos);
        if(pos==0)
        {
            ptr=start;
            start=start->next ;
            printf("The deleted element is:%d",ptr->info  );
            free(ptr);
```

```
                }
                else
                {
                        ptr=start;
                        for(i=0;i<pos;i++) { temp=ptr; ptr=ptr->next ;
                                if(ptr==NULL)
                                {
                                        printf("nPosition not Found:n");
                                        return;
                                }
                        }
                        temp->next =ptr->next ;
                        printf("The deleted element is:%d",ptr->info );
                        free(ptr);
                }
        }
}
```