```
/* Program 1: Create a Singly Linked List and perform following operation
on it-
 Insert a node at middle of the linked list.
 Insert a node at the last of the linked list.
 Delete first node of a linked list.
 Delete last node of a linked list.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node{
    int val;
    struct node *next;
}*start=NULL,*tmp;
void display(){
    tmp=start;
    printf("\n");
    if(tmp!=NULL)
    {
        do{
            printf("%d-->",tmp->val);
            tmp=tmp->next;
        }while(tmp!=NULL);
    printf("END");
    }
    else
    printf("NULL");
return ;
}
void ins_mid(){
    int after,val;
    printf("\nAfter : ");
    scanf("%d",&after);
    printf("\nValue : ");
    scanf("%d",&val);
    node *move;
    move=start;
    while(move->val!=after)
        move=move->next;
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=move->next;
    move->next=tmp;
    display();
}
void ins_last(){
    int val;
    printf("\nValue : ");
    scanf("%d",&val);
    node *move;
    move = start;
    while(move->next!=NULL)
        move=move->next;
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=NULL;
    move->next=tmp;
    display();
```

```
return ;
}
void del_first(){
    tmp=start;
    start=start->next;
    free(tmp);
    display();
return ;
}
void del_mid(){
    int val;
    printf("\nNode value : ");
    scanf("%d",&val);
    node *p,*move;
    move=p=start;
    while(move->val!=val)
    {
        p=move;
        move=move->next;
    }
    p->next=move->next;
    free(move);
    display();
return;
}
void del_last(){
    node *p,*move=start;
    while(move->next!=NULL){
        p=move;
        move=move->next;
    }
    p->next=NULL;
    free(move);
    display();
return;
}
void ins_first(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    tmp->next=(start==NULL?NULL:start);
    start=tmp;
    display();
return;
}

int main(){
    int ch=1;

    while(TRUE){
        printf("\n0.Insert at First\n1.Insert at middle\n2.Insert at
last\n3.Delete first node\n4.Delete middle node\n5.Delete last
node\n6.Diplay\n7.Exit\nEnter your choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 0:ins_first();
                   break;
            case 1:ins_mid();
                   break;
```

```
            case 2:ins_last();
                   break;
            case 3:del_first();
                   break;
            case 4:del_mid();
                   break;
            case 5:del_last();
                   break;
            case 6:display();
                   break;
            case 7:exit(0);
                   break;
            default:printf("Wrong choice retry....");
                   break;
        }
    }
return 0;
}
```

Output:

0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 0 Node value:85 85-->END 74-->232-->100-->99-->14-->95-->18-->41-->85-->END	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 1 After : 100 Value : 55 74-->232-->100-->55-->99-->14-->95-->18-->41-->85-->END	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 5 74-->232-->100-->55-->99-->14-->95-->18-->41-->END

```
/* Program 2: Create a Doubly Linked List and perform following operation
on it-
 Insert a node at middle of the doubly linked list.
 Insert a node at the last of the doubly linked list.
 Delete first node of a doubly linked list.
 Delete middle node of a doubly linked list.
 Delete last node of a doubly linked list.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node{
    int val;
    struct node *next,*prev;
}*start=NULL,*tmp,*last;

void display(){
    tmp=start;
    printf("\n");
    if(tmp!=NULL)
    {
        do
        {
            printf("%d<==>",tmp->val);
            tmp=tmp->next;
        }while(tmp!=NULL);
    printf("END");
    }
    else
    printf("NULL");
return ;
}
void ins_mid(){
    int after,val;
    printf("\nAfter : ");
    scanf("%d",&after);
    printf("\nValue : ");
    scanf("%d",&val);
    node *move;
    move=start;
    while(move->val!=after)
        move=move->next;
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=move->next;
    tmp->prev=move;
    move->next->prev=tmp;
    move->next=tmp;
    display();
}
void ins_last(){
    int val;
    printf("\nValue : ");
    scanf("%d",&val);
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=NULL;
    tmp->prev=last;
```

```
    last->next=tmp;
    last=tmp;
    display();
return ;
}
void del_first(){
    tmp=start;
    start=start->next;
    start->prev=NULL;
    free(tmp);
    display();
return ;
}
void del_mid(){
    int val;
    printf("\nNode value : ");
    scanf("%d",&val);
    node *p,*q,*move;
    move=start;
    while(move->val!=val)
        move=move->next;
    p=move->prev;
    q=move->next;
    p->next=q;
    q->prev=p;
    free(move);
    display();
return;
}
void del_last(){
    tmp=last;
    last=last->prev;
    last->next=NULL;
    free(tmp);
    display();
return;
}
void ins_first(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    if(start==NULL)
    {
        tmp->next=NULL;
        tmp->prev=NULL;
        start=last=tmp;
    }
    else
    {
        tmp->next=start;
        start->prev=tmp;
        start=tmp;
    }
    display();
return;
}

int main(){
    int ch=1;
```

```
    while(TRUE){
        printf("\n0.Insert at First\n1.Insert at middle\n2.Insert at
last\n3.Delete first node\n4.Delete middle node\n5.Delete last
node\n6.Diplay\n7.Exit\nEnter your choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 0:ins_first();
                   break;
            case 1:ins_mid();
                   break;
            case 2:ins_last();
                   break;
            case 3:del_first();
                   break;
            case 4:del_mid();
                   break;
            case 5:del_last();
                   break;
            case 6:display();
                   break;
            case 7:exit(0);
                   break;
            default:printf("Wrong choice retry....");
                   break;
        }
    }
return 0;
}
```

Output:

0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 0 Node value:32 32<==>END 121<==>13<==>85<==>441<==> 220<==>41<==>63<==>32<==>9 5<==>32<==>	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 1 After : 85 Value : 23 121<==>13<==>85<==>23<==>4 41<==>220<==>41<==>63<==>3 2<==>95<==>32<==>END	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 4 Node value : 41 121<==>13<==>85<==>23<==>4 41<==>220<==>63<==>32<==>9 5<==>32<==>E

```
/* Program 3: Create a Circular Singly Linked List and perform following
operation on it-
 Insert a node at middle of the Circular Singly linked list.
 Insert a node at the last of the Circular Singly linked list.
 Delete first node of a Circular Singly linked list.
 Delete middle node of a Circular Singly linked list.
 Delete last node of a Circular Singly linked list.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node{
    int val;
    struct node *next;
}*start=NULL,*tmp,*last;

void display(){
    tmp=start;
    printf("\n");
    if(tmp!=NULL)
    {
        do
        {
            printf("%d-->",tmp->val);
            tmp=tmp->next;
        }while(tmp!=start);
    printf("END");
    }
    else
    printf("NULL");

return ;
}
void ins_mid(){
    int after,val;
    printf("\nAfter : ");
    scanf("%d",&after);
    printf("\nValue : ");
    scanf("%d",&val);
    node *move;
    move=start;
    while(move->val!=after)
        move=move->next;
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=move->next;
    move->next=tmp;
    display();
}
void ins_last(){
    int val;
    printf("\nValue : ");
    scanf("%d",&val);
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=start;
    last->next=tmp;
    last=tmp;
```

```
    display();
return ;
}
void del_first(){
    tmp=start;
    start=start->next;
    last->next=start;
    free(tmp);
    display();
return ;
}
void del_mid(){
    int val;
    printf("\nNode value : ");
    scanf("%d",&val);
    node *p,*move;
    move=p=start;
    while(move->val!=val)
    {
        p=move;
        move=move->next;
    }
    p->next=move->next;
    free(move);
    display();
return;
}
void del_last(){
    node *move=start;
    while(move->next!=last)
        move=move->next;
    tmp=last;
    move->next=start;
    last=move;
    free(tmp);
    display();
return;
}
void ins_first(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    if(start==NULL)
    {
        tmp->next=tmp;
        start=last=tmp;
    }
    else
    {
        tmp->next=start;
        last->next=tmp;
        start=tmp;
    }
    display();
return;
}

int main(){
    int ch=1;
```

```
    while(TRUE){
        printf("\n0.Insert at First\n1.Insert at middle\n2.Insert at
last\n3.Delete first node\n4.Delete middle node\n5.Delete last
node\n6.Diplay\n7.Exit\nEnter your choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 0:ins_first();
                   break;
            case 1:ins_mid();
                   break;
            case 2:ins_last();
                   break;
            case 3:del_first();
                   break;
            case 4:del_mid();
                   break;
            case 5:del_last();
                   break;
            case 6:display();
                   break;
            case 7:exit(0);
                   break;
            default:printf("Wrong choice retry....");
                   break;
        }
    }
return 0;
}
```

Output:

0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 0 Node value:32 32-->END 23-->44-->63-->85-->23-->11-->960-->32-->END	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 2 Value : 51 23-->44-->63-->85-->23-->11-->960-->32-->51-->END	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 3 44-->63-->85-->23-->11-->960-->32-->51-->END

```
/* Program 4: Create a Circular Doubly Linked List and perform following
operation on it-
 Insert a node at middle of the Circular Doubly linked list.
 Insert a node at the last of the Circular Doubly linked list.
 Delete first node of a Circular Doubly linked list.
 Delete middle node of a Circular Doubly linked list.
 Delete last node of a Circular Doubly linked list.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node{
    int val;
    struct node *next,*prev;
}*start=NULL,*tmp,*last;

void display(){
    tmp=start;
    printf("\n");
    if(tmp!=NULL)
    {
        do
        {
            printf("%d<==>",tmp->val);
            tmp=tmp->next;
        }while(tmp!=start);
    printf("END");
    }
    else
    printf("NULL");
return ;
}
void ins_mid(){
    int after,val;
    printf("\nAfter : ");
    scanf("%d",&after);
    printf("\nValue : ");
    scanf("%d",&val);
    node *move;
    move=start;
    while(move->val!=after)
        move=move->next;
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=move->next;
    tmp->prev=move;
    move->next->prev=tmp;
    move->next=tmp;
    display();
}
void ins_last(){
    int val;
    printf("\nValue : ");
    scanf("%d",&val);
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=start;
    tmp->prev=last;
```

```
    last->next=tmp;
    last=tmp;
    start->prev=last;
    display();
return ;
}
void del_first(){
    tmp=start;
    start=start->next;
    start->prev=last;
    last->next=start;
    free(tmp);
    display();
return ;
}
void del_mid(){
    int val;
    printf("\nNode value : ");
    scanf("%d",&val);
    node *p,*q,*move;
    move=start;
    while(move->val!=val)
        move=move->next;
    p=move->prev;
    q=move->next;
    p->next=q;
    q->prev=p;
    free(move);
    display();
return;
}
void del_last(){
    tmp=last;
    last=last->prev;
    last->next=start;
    start->prev=last;
    free(tmp);
    display();
return;
}
void ins_first(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    if(start==NULL)
    {
        tmp->next=tmp;
        tmp->prev=tmp;
        start=last=tmp;
    }
    else
    {
        tmp->next=start;
        tmp->prev=last;
        start->prev=tmp;
        last->next=tmp;
        start=tmp;
    }
    display();
```

```
return;
}

int main(){
    int ch=1;
    while(TRUE){
        printf("\n0.Insert at First\n1.Insert at middle\n2.Insert at
last\n3.Delete first node\n4.Delete middle node\n5.Delete last
node\n6.Diplay\n7.Exit\nEnter your choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 0:ins_first();
                   break;
            case 1:ins_mid();
                   break;
            case 2:ins_last();
                   break;
            case 3:del_first();
                   break;
            case 4:del_mid();
                   break;
            case 5:del_last();
                   break;
            case 6:display();
                   break;
            case 7:exit(0);
                   break;
            default:printf("Wrong choice retry....");
                   break;
        }
    }
return 0;
}
```

Output:

0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 0 Node value:33 3<==>850<==>65<==>45<==>123<==>65<==>41<==>33<==>END	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 2 Value : 88 3<==>850<==>65<==>45<==>123<==>65<==>41<==>33<==>88<==>END	0.Insert at First 1.Insert at middle 2.Insert at last 3.Delete first node 4.Delete middle node 5.Delete last node 6.Diplay 7.Exit Enter your choice: 6 3<==>850<==>65<==>45<==>123<==>65<==>41<==>33<==>88<==>END

```
/* Program 5: given a linked list and two integers M and N. Traverse the
linked list such that you retain M nodes then delete next N nodes, continue
the same until end of the linked list.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node {
    int val;
    struct node *next;
}*start = NULL, *tmp;

void display() {
    tmp = start;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d-->", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");

    return;
}
void insert(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    tmp->next=(start==NULL?NULL:start);
    start=tmp;
    display();
return;
}
void del(int m, int n) {
    int l, c;
    node *move = start, *p, *q;
    while (move != NULL) {
        for (l = m; l > 0 && move != NULL; --l) {
            p = move;
            move = move->next;
        }
        for (c = n; c > 0 && move != NULL; --c) {
            q = move->next;
            if (move == start) {
                tmp = start;
                start = q;
                free(tmp);
                move = q;
            } else {
                tmp = move;
                p->next = q;
                free(tmp);
                move = q;
            }
```

```
        }
    }
    display();
}
int main() {
    int ch = 1;
    while (TRUE) {
        printf("\n0.Insert\n1.Del\n2.Dispaly\n3.Exit\nEnter your choice:
");
        scanf("%d", &ch);
        switch (ch){
        case 0:
            insert();
            break;
        case 1:
            printf("\nEnter M(nodes to leave) : ");
            scanf("%d",&m);
            printf("\nEnter N(nodes to delete) : ");
            scanf("%d",&n);
            del(m,n);
            break;
        case 2:
            display();
            break;
        case 3:
            exit(0);
        default:
            printf("Wrong choice retry....");
            break;
        }
    }
    return 0;
}
```

Output:

43-->228-->332-->96-->44-->99-->12-->74-->99-->41-->85-->32-->END
0.Insert
1.Del
2.Dispaly
3.Exit
Enter your choice: 1

Enter M(nodes to leave) : 2

Enter N(nodes to delete) : 3

0.Insert
1.Del
2.Dispaly
3.Exit
Enter your choice: 2

43-->228-->99-->12-->85-->32-->END

```
/* Program 6: Write a program to print elements of a singly linked list in
reverse.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node {
    int val;
    struct node *next;
}*start = NULL, *tmp;

void display() {
    tmp = start;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d-->", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");
    return;
}
void insert(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    tmp->next=(start==NULL?NULL:start);
    start=tmp;
    display();
return;
}
void reverse(){
    node *p,*q;
    p=NULL;
    q=start;
    if(!q||(!(q->next))){
        return;
    }
    while(q!=NULL){
        tmp=q->next;
        q->next=p;
        p=q;
        q=tmp;
    }
    start=p;
    display();
}
int main() {
    int ch = 1;
    while (TRUE) {
        printf("\n0.Insert\n1.Reverse\n2.Display\n3.Exit\nEnter your
choice: ");
        scanf("%d", &ch);
        switch (ch){
```

```
            case 0:
                insert();
                break;
            case 1:
                reverse();
                break;
            case 2:
                display();
                break;
            case 3:
                exit(0);
            default:
                printf("Wrong choice retry....");
                break;
            }
        }
        return 0;
}
```

Output:

```
36-->95-->68-->77-->49-->62-->45-->66-->98-->74-->11-->62-->32-->END
0.Insert
1.Reverse
2.Display
3.Exit
Enter your choice: 1

32-->62-->11-->74-->98-->66-->45-->62-->49-->77-->68-->95-->36-->END
```

```
/* Program 7: write a program that will delete any duplicates node from the
linked list.*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node {
    int val;
    struct node *next;
}*start = NULL, *tmp;

void display() {
    tmp = start;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d-->", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");

    return;
}
void insert(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    tmp->next=(start==NULL?NULL:start);
    start=tmp;
    display();
return;
}
void remove_dup(){
    node* move,*chk,*p;
    chk=start;
    while(chk!=NULL){
        p=chk;
        move=chk->next;
        while(move!=NULL){
            if(chk->val==move->val){
                    tmp=move;
                    p->next=move->next;
                    move=move->next;
                    free(tmp);
                    continue;
            }
            p=move;
            move=move->next;
        }
        chk=chk->next;
    }
    display();
}

int main() {
```

```
    int ch = 1;
    while (TRUE) {
        printf("\n0.Insert\n1.Remove duplicates\n2.Display\n3.Exit\nEnter
your choice: ");
        scanf("%d", &ch);
        switch (ch){
        case 0:
            insert();
            break;
        case 1:
            remove_dup();
            break;
        case 2:
            display();
            break;
        case 3:
            exit(0);
        default:
            printf("Wrong choice retry....");
            break;
        }
    }
    return 0;
}
```

Output:

85-->11-->11-->23-->62-->23-->66-->96-->330-->66-->END
0.Insert
1.Remove duplicates
2.Display
3.Exit
Enter your choice: 1

85-->11-->23-->62-->66-->96-->330-->END

```
/* Program 8: given a linked list and a number k. Reverse every k nodes in
the list.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node {
    int val;
    struct node *next;
}*start = NULL, *tmp;

void display() {
    tmp = start;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d-->", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");
    return;
}
void insert(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->val=val;
    tmp->next=(start==NULL?NULL:start);
    start=tmp;
    display();
return;
}

node *reverse(node *head,int k){
    int i;
    node *p,*q;
    if(head==NULL||head->next==NULL)
        return head;
    p=NULL;
    q=head;
    tmp=start;
    for(i=0;i<k&&tmp;i++){
        tmp=q->next;
        q->next=p;
        p=q;
        q=tmp;
    }
    head->next=reverse(tmp,k);
    return p;
}

int main() {
    int ch = 1,k;
    while (TRUE) {
```

```
        printf("\n0.Insert\n1.Reverse\n2.Display\n3.Exit\nEnter your
choice: ");
        scanf("%d", &ch);
        switch (ch){
        case 0:
            insert();
            break;
        case 1:
            printf("\nEnter k: ");
            scanf("%d",&k);
            start=reverse(start,k);
            display();
            break;
        case 2:
            display();
            break;
        case 3:
            exit(0);
        default:
            printf("Wrong choice retry....");
            break;
        }
    }
    return 0;
}
```

Output:

73-->91-->82-->77-->12-->340-->850-->420-->62-->12-->65-->END
0.Insert
1.Reverse
2.Display
3.Exit
Enter your choice: 1

Enter k: 3

82-->91-->73-->340-->12-->77-->62-->420-->850-->65-->12-->END

```
/* Program 9: Given two linked lists, insert nodes of second list into
first list at alternate positions of first list.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node{
    int val;
    struct node *next;
}*start1=NULL,*start2=NULL,*tmp1=NULL,*tmp2=NULL;

void display(node **source){
    tmp1=*source;
    printf("\n");
    if(tmp1!=NULL)
    {
        do
        {
            printf("%d-->",tmp1->val);
            tmp1=tmp1->next;
        }while(tmp1!=NULL);
    printf("END");
    }
    else
    printf("NULL");
return ;
}
void insert(node **source){
    char ch='Y';
    node *tmp;
    while(ch=='Y'||ch=='y'){
        int val;
        printf("\nNode value:");
        scanf("%d",&val);
        tmp=(node *)malloc(sizeof(node));
        tmp->val=val;
        tmp->next=(*source==NULL?NULL:*source);
        *source=tmp;
        display(source);
        printf("\nMore(Y/N) :");
        scanf("%s",&ch);
    }
return;
}
void merge_list(node **s1,node **s2){
    node *m1,*m2;
    m1=tmp1=*s1;
    m2=tmp2=*s2;
    while(tmp1&&tmp2){
        tmp1=m1->next;
        tmp2=m2->next;
        m1->next=m2;
        m2->next=tmp1;
        m1=tmp1;
        m2=tmp2;
    }
   display();
}
```

```
int main(){
    int ch=1;
    while(TRUE){
        printf("\n0.Insert\n1.Merge lists\n2.Display\n3.Exit\nEnter your
choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 0:printf("\nlist 1:");
                   insert(&start1);
                   printf("\nlist 2:");
                   insert(&start2);
                   break;
            case 1:merge_list(&start1,&start2);
                   break;
            case 2:display(&start1);
                   break;
            case 3:exit(0);
                   break;
            default:printf("Wrong choice retry....");
                   break;
        }
    }
return 0;
}
```

Output:

76-->321-->98-->42-->96-->36-->END
More(Y/N) :n

list 2:
Node value:55

97-->13-->71-->93-->36-->56-->55-->END
More(Y/N) :n

0.Insert
1.Merge lists
2.Display
3.Exit
Enter your choice: 1

76-->97-->321-->13-->98-->71-->42-->93-->96-->36-->36-->56-->END

```
/* Program 10: Write a C function that moves last element to front in a
given Singly Linked List.*/

#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
struct node {
    int val;
    struct node *next;
}*start = NULL, *tmp;

void display() {
    tmp = start;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d-->", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");
    return;
}
void insert(){
    int val;
    char ch='Y';
    while(ch=='Y'||ch=='y'){
        printf("\nNode value:");
        scanf("%d",&val);
        tmp=(struct node *)malloc(sizeof(struct node));
        tmp->val=val;
        tmp->next=(start==NULL?NULL:start);
        start=tmp;
        display();
        printf("\nMore(Y/N) : ");
        scanf("%s",&ch);
    }
return;
}
void exchange(){
    node *p,*move=start;
    while(move->next!=NULL){
            p=move;
            move=move->next;
    }
    p->next=start;
    move->next=start->next;
    start->next=NULL;
    start=move;
    display();
}
int main(){
    int ch=1;
    while(TRUE){
        printf("\n0.Insert\n1.Exchange\n2.Display\n3.Exit\nEnter your
choice: ");
        scanf("%d",&ch);
```

```
        switch(ch){
            case 0:insert();
                    break;
            case 1:exchange();
                    break;
            case 2:display();
                    break;
            case 3:exit(0);
                    break;
            default:printf("Wrong choice retry....");
                    break;
        }
    }
return 0;
}
```

Output:

78-->99-->0-->290-->41-->110-->36-->96-->END
More(Y/N) : n

0.Insert
1.Exchange
2.Display
3.Exit
Enter your choice: 1

96-->99-->0-->290-->41-->110-->36-->78-->END