```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *head=NULL;
void  create(void);
void display(void);
void ins_at_beg(int);
void ins_at_end(int);
void ins_at_pos(int);
int length(void);
void del_at_beg(void);
void del_at_end(void);
void del_at_pos(void);
void reverse(void);
void middlenode(void);
void checkloop(void);
void main()
{
    int choice,info;
    while(1)
    {
        printf("1.Create\n");
        printf("2.Insert at the beginning\n");
        printf("3.Insert at the ending\n");
        printf("4.Insert at the specific position\n");
        printf("5.Delete at beginning\n");
        printf("6.Delete at ending\n");
        printf("7.Delete at specific position\n");
        printf("8.Display\n");
        printf("9.Reverse\n");
        printf("10.Middlenode\n");
        printf("11.Checkloop\n");
        printf("12.Exit\n");
        printf("Enter your choice ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: create();
                    break;
            case 2: printf("Enter your data ");
                    scanf("%d",&info);
                    ins_at_beg(info);
                    break;
            case 3: printf("Enter your data ");
                    scanf("%d",&info);
                    ins_at_end(info);
                    break;
            case 4: printf("Enter your data ");
                    scanf("%d",&info);
                    ins_at_pos(info);
                    break;
            case 5: del_at_beg();
                    break;
            case 6: del_at_end();
```

```
                    break;
            case 7: del_at_pos();
                    break;
            case 8: display();
                    break;
            case 9: reverse();
                    break;
            case 10: middlenode();
                    break;
            case 11: checkloop();
                    break;
            case 12: exit(0);
            default: printf("Incorrect Input\n");
        }
    }
}
void create()
{
    struct node *newnode,*temp;
    int info,ch=1;
    while(ch)
    {
        newnode=(struct node *)malloc(sizeof(struct node));
        if(newnode==NULL)
        {
            printf("Malloc error\n");
            return;
        }
        printf("Enter your data ");
        scanf("%d",&info);
        newnode->data=info;
        newnode->next=NULL;
        if(head==NULL)
        {
            head=temp=newnode;
        }
        else
        {
            temp->next=newnode;
            temp=newnode;
        }
        printf("Do you want to continue ");
        scanf("%d",&ch);
    }
}
void ins_at_beg(int info)
{
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Malloc Error\n");
        return;
    }
    newnode->data=info;
    newnode->next=NULL;
    if(head==NULL)
    {
        head=newnode;
    }
    else
```

```
    {
        newnode->next=head;
        head=newnode;
    }
}
void ins_at_end(int info)
{
    struct node *newnode,*temp;
    temp=head;
    newnode=(struct node *)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Malloc Error\n");
        return;
    }
    newnode->data=info;
    newnode->next=NULL;
    if(temp==NULL)
    {
        head=temp=newnode;
    }
    else
    {
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
    }
}
void ins_at_pos(int info)
{
    int pos,i=1,l;
    struct node *newnode,*temp;
    temp=head;
    printf("Enter your position ");
    scanf("%d",&pos);
    l=length();
    if(pos<=0||pos>l)
    {
        printf("Invalid Position\n");
        return;
    }
    newnode=(struct node *)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Malloc Error\n");
        return;
    }
    newnode->data=info;
    newnode->next=NULL;
    if(temp==NULL)
    {
        head=temp=newnode;
    }
    else
    {
        while(i<pos-1)
```

```
        {
            temp=temp->next;
            i++;
        }
        newnode->next=temp->next;
        temp->next=newnode;
    }
}
void del_at_beg()
{
    struct node *temp;
    temp=head;
    if(temp==NULL)
    {
        printf("List is empty\n");
        return;
    }
    head=head->next;
    free(temp);
}
void del_at_end()
{
    struct node *temp,*prev;
    temp=head;
    if(temp==NULL)
    {
        printf("List is empty\n");
        return;
    }
    while(temp->next!=NULL)
    {
        prev=temp;
        temp=temp->next;
    }
    prev->next=NULL;
    free(temp);
}
void del_at_pos()
{
    struct node *temp,*prev;
    temp=head;
    prev=NULL;
    int pos,l,i=1;
    printf("Enter your position ");
    scanf("%d",&pos);
    l=length();
    if(pos<=0||pos>l)
    {
        printf("Invalid position\n");
        return;
    }
    while(i<pos)
    {
        prev=temp;
        temp=temp->next;
        i++;
    }
    prev->next=temp->next;
    free(temp);
}
```

```
void display()
{
    struct node *temp;
    temp=head;
    if(temp==NULL)
    {
        printf("List is empty\n");
        return;
    }
    while(temp!=NULL)
    {
        printf("%d\n",temp->data);
        temp=temp->next;
    }
}

void reverse()
{
    struct node *prev,*current,*nextnode;
    prev=NULL;
    current=nextnode=head;
    while(nextnode!=NULL)
    {
        nextnode=current->next;
        current->next=prev;
        prev=current;
        current=nextnode;
    }
    head=prev;
}

void middlenode()
{
    struct node *temp,*prev;
    temp=prev=head;
    if(temp==NULL)
    {
        printf("List is empty\n");
        return;
    }
    while(temp!=NULL && temp->next!=NULL)
    {
        prev=prev->next;
        temp=temp->next->next;
    }
    printf("%d\n",prev->data);
}
void checkloop()
{
    struct node *fast,*slow;
    slow=fast=head;
    while(fast!=NULL&&fast->next!=NULL)
    {
        slow=slow->next;
        fast=fast->next->next;
        if(slow==fast)
        {
            printf("Loop found\n");
            return;
        }
    }
    printf("Loop not found\n");
}
```

```
int length()
{
    struct node *temp;
    int count=0;
    temp=head;
    while(temp!=NULL)
    {
        count++;
        temp=temp->next;
    }
    return count;
}
```