```
#include<iostream>
using namespace std;

/*
@swasthiiiiii
*/

typedef struct node
{
    int data;
    struct node *left;
    struct node *right;
}NODE;

class dlist
{
    public:
    NODE *fninsertbegin(NODE *);
    NODE *fninsertend(NODE *);
    NODE *fninsertpos(NODE *,int);
    NODE *fndeletebegin(NODE *);
    NODE *fndeleteend(NODE *);
    NODE *fndeletepos(NODE *,int);
    void fndisplay(NODE *);
};

int main()
{
    NODE *head=NULL;
    dlist dl;
    int ch,pos;
    while(1)
    {
        cout<<"\n\nmenu\n";
        cout<<"1.insert begin\n";
        cout<<"2.insert end\n";
        cout<<"3.insert pos\n";
        cout<<"4.delete begin\n";
        cout<<"5.delete end\n";
        cout<<"6.delete position\n";
        cout<<"7.display\n";
        cout<<"8.exit\n";
        cout<<"enter your choice:\n";
        cin>>ch;
        switch (ch)
        {
            case 1:head=dl.fninsertbegin(head);
            break;
```

```
            case 2:head=dl.fninsertend(head);
            break;
            case 3:cout<<"enter the position:";
                   cin>>pos;
                   head=dl.fninsertpos(head,pos);
            break;
            case 4:head=dl.fndeletebegin(head);
            break;
            case 5:head=dl.fndeleteend(head);
            break;
            case 6:cout<<"enter the position:";
                   cin>>pos;
                   head=dl.fndeletepos(head,pos);
            break;
            case 7:dl.fndisplay(head);
            break;
            case 8:exit(0);
            }
       }
    return 0;
}
NODE *dlist::fninsertbegin(NODE *head)
{
    int num;
    NODE *newnode;
    newnode=(NODE *)malloc(sizeof(NODE));
    cout<<"enter the value for node:";
    cin>>num;

    newnode->data=num;
    newnode->right=NULL;
    newnode->left=NULL;

    if(head==NULL)
    {
        head=newnode;
    }
    else
    {
        newnode->right=head;
        head->left=newnode;
        head=newnode;
    }
    return head;
}

NODE *dlist::fninsertend(NODE *head)
{
```

```
    NODE *newnode;
    NODE *temp;
    int num;
    newnode=(NODE *)malloc(sizeof(NODE));

    cout<<"enter the value for node:";
    cin>>num;
    newnode->data=num;
    newnode->right=NULL;
    newnode->left=NULL;

    if(head==NULL)
    {
        head=newnode;
    }
    else
    {
     temp=head;
     while(temp->right!=NULL)
     {
         temp=temp->right;
     }
     newnode->left=temp;
     temp->right=newnode;
    }
    return head;
}

NODE *dlist::fninsertpos(NODE *head,int pos)
{
    NODE *newnode,*temp,*temp1,*temp2;
    int num,count=0;

    temp=head;
    while(temp!=NULL)
    {
    count++;
    temp=temp->right;
    }
    if(pos>count)
    {
        cout<<"invalid position\n";
    }
    else
    {
        newnode=(NODE *)malloc(sizeof(NODE));
        cout<<"enter the value for node:";
        cin>>num;
```

```
        newnode->data=num;
        newnode->right=NULL;
        newnode->left=NULL;
        if(pos==0 && head==NULL)
        {
            head=newnode;
        }
        else if(pos==0)
        {
            newnode->right=head;
            head->left=newnode;
            head=newnode;
        }
        else if(pos==count)
        {
            temp=head;
            while(temp->right!=NULL)
            {
                temp=temp->right;
            }
            temp->right=newnode;
            newnode->left=temp;
        }
        else
        {
            temp2=head;
            for(int i=0;i<pos;i++)
            {
                temp1=temp2;
                temp2=temp2->right;
            }
            temp1->right=newnode;
            newnode->right=temp2;
            temp2->left=newnode;
            newnode->left=temp1;
        }
    }
    return head;
}

NODE *dlist::fndeletebegin(NODE *head)
{
    NODE *temp;
    if(head==NULL)
    {
        cout<<"list is empty\n";
    }
    else if(head->right==NULL)
```

```
    {
        cout<<"deleted:"<<head->data;
        head=NULL;
        free(head);
    }
    else
    {
        temp=head;
        head=head->right;
        temp->right=NULL;
        cout<<"deleted:"<<temp->data;
        free(temp);
    }
    return head;
}

NODE *dlist::fndeleteend(NODE *head)
{
    NODE *temp1,*temp2;
    if(head==NULL)
    {
        cout<<"list is empty";
    }
    else if(head->right==NULL)
    {
        cout<<"deleted:"<<head->data;
        head=NULL;
        free(head);
    }
    else
    {
        temp2=head;
        while(temp2->right!=NULL)
        {
            temp1=temp2;
            temp2=temp2->right;
        }
        temp1->right=NULL;
        cout<<"deleted:"<<temp2->data;
        free(temp2);
    }
    return head;
}

NODE *dlist::fndeletepos(NODE *head,int pos)
{
    NODE *temp,*temp1,*temp2,*temp3;
    int count=0;
```

```
temp=head;
while(temp!=NULL)
{
    count++;
    temp=temp->right;
}
if(pos>count)
{
    cout<<"invalid position";
}
else
{
    if(pos==1 && head->right==NULL)
    {
        cout<<"deleted:"<<head->data;
        head=NULL;
        free(head);
    }
    else if(pos==1)
    {
        temp=head;
        head=head->right;
        temp->right=NULL;
        head->left=NULL;
        cout<<"deleted:"<<temp->data;
        free(temp);
    }
    else if(pos==count)
    {
        temp2=head;
        while(temp2->right!=NULL)
        {
        temp1=temp2;
        temp2=temp2->right;
        }
        temp1->right=NULL;
        cout<<"deleted:"<<temp2->data;
        free(temp2);
    }
    else
    {
        temp2=head;
        for(int i=1;i<pos;i++)
        {
            temp1=temp2;
            temp2=temp2->right;
        }
```

```
            temp3=temp2->right;
            temp1->right=temp3;
            temp3->left=temp1;

            temp2->left=NULL;
            temp2->right=NULL;
            cout<<"deleted:"<<temp2->data;
            free(temp2);
        }
    }
    return head;
}

void dlist::fndisplay(NODE *head)
{
    NODE *temp;
    temp=head;
    if(head==NULL)
    {
    cout<<"The list is empty"<<endl;
    }
    else
    {
    cout<<"The elements are :";
    while (temp!=NULL)
    {
    cout<<temp->data<<" ";
    temp=temp->right;
    }
    cout<<endl;
    }
}
```