**L1)**

**#include<iostream.h>**

**#include<conio.h>**

**#include<process.h>**

**class node{ int data;**

**node \*next, \*prev;**

**node \*head, \*tail;**

**public:**

**node()**

**{head = NULL;**

**tail = NULL;**

**}**

**node\* insert(int);**

**void disp(node\*);**

**node\* concat();**

**};node \*head1, \*head2;**

**void main()**

**{node obj;**

**node \*t = new node;**

**int n,m;**

**clrscr();**

**cout<<"enter no. of elements for 1st : ";cin>>n;**

**cout<<"enter no. of elements for 2nd : ";cin>>m;**

**head1=obj.insert(n);**

**head2=obj.insert(m);**

**t= obj.concat();**

**obj.disp(t);**

**getch();**

**}**

**node \*node::insert(int a)**

**{head=NULL;**

**tail=NULL;**

**for(int i=0 ; i<a ; i++)**

**{node \*temp = new node;**

**temp->next = NULL;**

**temp->prev = NULL;**

**cout<<"enter value"<<endl;**

**cin>>temp->data;**

**if(head!=NULL && tail!=NULL)**

**{ tail->next=temp;**

**temp->prev=tail;**

**tail=temp;**

**}**

**else**

**tail=head=temp;**

**}**

**return head;**

**}**

**void node::disp(node \*t)**

**{if(t==NULL)**

**cout<<"empty!"<<endl;**

**while(t!=NULL)**

**{cout<<t->data<<"\t";**

**t=t->next;**

**}**

**cout<<endl;**

**}**

**node\* node::concat()**

**{node \*h = new node;**

**if(head1==NULL && head2==NULL)**

**return NULL;**

**if(head1==NULL)**

**return head2;**

**if(head2==NULL)**

**return head1;**

**h=head1;**

**while(h->next!=NULL)**

**{h=h->next;}**

**h->next=head2;**

**head2->prev=h;**

**return head1;**

**}**

**L2)**

**#include<iostream.h>**

**#include<conio.h>**

**#include<process.h>**

**class node{ int data;**

**node \*next, \*prev;**

**node \*head, \*tail;**

**public:**

**node()**

**{head = NULL;**

**tail = NULL;**

**}**

**void insert();**

**void disp();**

**void delstack();**

**void delqueue();**

**};**

**void main()**

**{node obj;**

**int ch,n;**

**clrscr();**

**cout<<"enter no. of elements:-"<<endl;**

**cin>>n;**

**for(int i=0 ; i<n ; i++)**

**obj.insert();**

**obj.disp();**

**cout<<endl;**

**while(1)**

**{cout<<"1.Insert \n2.Delete as stack \n3.Delete as queue\n4.Exit"<<endl;**

**cout<<"enter your choice"<<endl;**

**cin>>ch;**

**switch(ch)**

**{case 1 : obj.insert();obj.disp();break;**

**case 2 : obj.delstack();obj.disp();break;**

**case 3 : obj.delqueue();obj.disp();break;**

**case 4 : exit(0);**

**default : cout<<"enter correctly!"<<endl;**

**}**

**getch();**

**}**

**}**

**void node::insert()**

**{node \*temp = new node;**

**temp->next = NULL;**

**cout<<"enter value"<<endl;**

**cin>>temp->data;**

**if(head!=NULL && tail!=NULL)**

**{tail->next=temp;**

**temp->prev=tail;**

**tail=temp;**

**}**

**else**

**tail=head=temp;**

**}**

**void node::disp()**

**{node \*temp=head;**

**if(temp==NULL)**

**cout<<"empty!"<<endl;**

**while(temp!=NULL)**

**{cout<<temp->data<<"\t";**

**temp=temp->next;**

**}**

**cout<<endl;**

**}**

**void node::delstack()**

**{node\* temp=tail;**

**If(head==tail)**

**head=NULL;**

**tail=temp->prev;**

**tail->next=NULL;**

**delete(temp);**

**}**

**void node::delqueue()**

**{**

**node\* temp=head;**

**head=temp->next;**

**head->prev=NULL;**

**delete(temp);**

**}**

**L3)**

#include<iostream.h>

#include<conio.h>

#include<process.h>

class node{ int data;

node \*next, \*prev;

node \*head, \*tail;

public:

node()

{head = NULL;

tail = NULL;

}

void insert();

void disp();

void in\_before();

void in\_after();

void del();

void traverse();

void rev();

};

void main()

{node obj;

int ch,n;

clrscr();

cout<<"enter no. of elements:-"<<endl;

cin>>n;

for(int i=0 ; i<n ; i++)

obj.insert();

obj.disp();

cout<<endl;

while(1)

{cout<<"1.Insert before \n 2.Insert after \n 3.Delete \n 4.Traverse"<<endl;

cout<<"5.Reverse \n6.Exit \n";

cout<<"enter your choice"<<endl;

cin>>ch;

switch(ch)

{case 1 : obj.in\_before();obj.disp();break;

case 2 : obj.in\_after();obj.disp();break;

case 3 : obj.del();obj.disp();break;

case 4 : obj.traverse();break;

case 5 : obj.rev();obj.disp();break;

case 6 : exit(0);

default : cout<<"enter correctly!"<<endl;

}

getch();

}

}

void node::insert()

{node \*temp = new node;

temp->next = NULL;

temp->prev = NULL;

cout<<"enter value"<<endl;

cin>>temp->data;

if(head!=NULL && tail!=NULL)

{tail->next=temp;

temp->prev=tail;

tail=temp;

}

else

tail=head=temp;

}

void node::disp()

{node \*temp=head;

if(temp==NULL)

cout<<"empty!"<<endl;

while(temp!=NULL)

{cout<<temp->data<<"\t";

temp=temp->next;

}

cout<<endl;

}

void node::rev()

{node \*temp=head, \*r;

head=NULL;

while(temp!=NULL)

{

r=temp->next;

temp->prev=temp->next;

temp->next=head;

head=temp;

temp=r;

}

}

void node::traverse()

{ node \*temp=head;

if(temp==NULL)

{ cout<<"list empty!"<<endl;

return;

}

cout<<"contents :-"<<endl;

while(temp!=NULL)

{ cout<<temp->data<<endl;

temp=temp->next;

}

}

void node::del()

{ cout<<"enter element to be deleted"<<endl;

int item;

cin>>item;

node \*temp=head;

if(head->data==item)

{head=head->next;

head->prev=NULL;

delete (temp);

}

else

{ while(temp!=NULL)

{ if(temp->data==item)

{ temp->prev->next=temp->next;

temp->next->prev=temp->prev;

delete(temp);

}

temp=temp->next;

}

}

}

void node::in\_after()

{cout<<"enter element :-"<<endl;

int a,pos=0,c=1;

cin>>a;

node \*temp=head;

while(temp!=NULL)

{ if(temp->data == a)

{ pos = c;

break;

}

temp=temp->next;

c++;

}

if(pos==0)

{ cout<<"element not present!"<<endl;

return;

}

node \*link=new node;

cout<<"enter value"<<endl;

cin>>link->data;

link->next=temp->next;

temp->next->prev=link;

temp->next=link;

link->prev=temp;

}

void node::in\_before()

{

cout<<"element:-"<<endl;

int a,pos=0,c=1;

cin>>a;

node \*temp=head;

node \*link=new node;

link->next=NULL;

link->prev=NULL;

while(temp!=NULL)

{ if(temp->data == a)

{ pos = c;

break;

}

temp=temp->next;

c++;

}

if(pos==0)

{ cout<<"element not present!"<<endl;

return;

}

cout<<"enter value"<<endl;

cin>>link->data;

if(pos==1)

{head->prev=link;

link->next=head;

head=link;

}

else

{

temp->prev->next=link;

link->prev=temp->prev;

link->next=temp;

temp->prev=link;

}

}