UNIT-1 PROGRAMS

ARRAYS(all operations):

```
#include<iostream>
using namespace std;
class array
{
	public:
		int a[100],i,n,position,temp[100];
		cout<<"enter the no of elements:"<<endl;
		cin>>n;
		cout<<"enter the elements:"<<endl;
		for(i=0;i<n;i++)
		cin>>a[i];
		void insertatbegin(int x)
		{
			cout<<"enter the value to be inserted at beginning:"<<endl;
			cin>>x;
			for(i=n-1;i>0;i--)
			{
				a[i+1]=a[i];
				a[0]=x;
			}
		}
		void insertatend(int y)
		{
			cout<<"enter the value to be inserted at end:"<<endl;
			cin>>y;
			a[n]=y;
		}
		void insertatmiddle(int p)
		{
			cout<<"enter the position:"<<endl;
			cin>>position;
			cout<<"enter the value to be inserted at middle:"<<endl;
			cin>>x;
			for(i=n-1;i>=position-1;i--)
			{
				a[i+1]=a[i];
				a[position-1]=p;
			}
```

```
                    }
                    void deleteatbegin()
                    {
                              for(i=1;i<n;i++)
                              {
                                        a[i]=a[i+1];
                              }
                    }
                    void deleteatmiddle()
                    {
                              cout<<"enter the position:"<<endl;
                              cin>>position
                              if(position>n+1)
                              cout<<"deletion is not possible";
                              else
                              {
                              for(i=position-1;i<n-1;i++)
                              a[i]=a[i+1];
                              }
                    }
                    void deleteatend()
                    {
                              for(i=0;i<n-1;i++)
                              {
                              if(i<n)
                              temp[i]=a[i];
                              }
                    }

}
int main()
{
          array ob;
          int ch;
          while(1)
          {
                    cout<<endl<<"1.Insert at
begin"<<endl<<"2.Display"<<endl<<"3.Exit"<<endl<<"Enter Your Choice:";
                    cin>>ch;
                    switch(ch)
                    {
                              case 1:cout<<endl<<"Enter an element to insert:";
                                                  cin>>x;
                                                  ob.insertAtBegin(x);
```

```
                                                break;
                        case 2:ob.display();
                                                break;
                        case 3:return 0;
                        default:cout<<endl<<"WRONG CHOICE";
}
```

TWO-DIMENSIONAL ARRAYS:

MATRIX MULTIPLICATION:

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

int main()
{
   int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k;

   cout << "Enter rows and columns for first matrix: ";
   cin >> r1 >> c1;
   cout << "Enter rows and columns for second matrix: ";
   cin >> r2 >> c2;

   // If column of first matrix in not equal to row of second matrix,
   // ask the user to enter the size of matrix again.
   while (c1!=r2)
   {
      cout << "Error! column of first matrix not equal to row of second.";

      cout << "Enter rows and columns for first matrix: ";
      cin >> r1 >> c1;

      cout << "Enter rows and columns for second matrix: ";
      cin >> r2 >> c2;
   }

   // Storing elements of first matrix.
   cout << endl << "Enter elements of matrix 1:" << endl;
   for(i = 0; i < r1; ++i)
      for(j = 0; j < c1; ++j)
      {
         cout << "Enter element a" << i + 1 << j + 1 << " : ";
         cin >> a[i][j];
      }
```

```
    // Storing elements of second matrix.
    cout << endl << "Enter elements of matrix 2:" << endl;
    for(i = 0; i < r2; ++i)
        for(j = 0; j < c2; ++j)
        {
            cout << "Enter element b" << i + 1 << j + 1 << " : ";
            cin >> b[i][j];
        }

    // Initializing elements of matrix mult to 0.
    for(i = 0; i < r1; ++i)
        for(j = 0; j < c2; ++j)
        {
            mult[i][j]=0;
        }

    // Multiplying matrix a and b and storing in array mult.
    for(i = 0; i < r1; ++i)
        for(j = 0; j < c2; ++j)
            for(k = 0; k < c1; ++k)
            {
                mult[i][j] += a[i][k] * b[k][j];
            }

    // Displaying the multiplication of two matrix.
    cout << endl << "Output Matrix: " << endl;
    for(i = 0; i < r1; ++i)
    for(j = 0; j < c2; ++j)
    {
        cout << " " << mult[i][j];
        if(j == c2-1)
            cout << endl;
    }

    return 0;
}
```

SINGLE LINKED LIST:

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

struct node
{
        public:
```

```
	int data;
	node *next;
};

class SLL
{
	node *head;
	public:
	SLL() {head=NULL;} //constructor to initialize
	void insertAtBegin(int x);
	void insertAtEnd(int x);
	void insertAtMiddle(int pos,int x);
	int removeAtBegin();
	int removeAtEnd();
	int removeAtMiddle(int pos);
	void display();
	void reverseList();
};
void SLL::insertAtBegin(int x)
{
	node *temp=new node;
	temp->data=x;
	temp->next=NULL;
	if(head==NULL)
	{
		head=temp;
	}
	else
	{
		temp->next=head;
		head=temp;
	}
}
void SLL::insertAtEnd(int x)
{
	node *temp=new node;
	node *p;
	temp->data=x;
	temp->next=NULL;
	if(head==NULL)
	{
		head=temp;
	}
	else
	{
		p=head;
		while(p->next!=NULL)
```

```
                    p=p->next;
            p->next=temp;
        }
}
void SLL::insertAtMiddle(int pos,int x)
{
        node *temp=new node;
        temp->data=x;
        temp->next=NULL;
        node *p;
        int i=0;
        for(i=1,p=head;i<pos-1 && p!=NULL;i++)
                    p=p->next;
        if(i==pos-1)
        {
            temp->next=p->next;
            p->next=temp;
        }
        else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
}
int SLL::removeAtBegin()
{
        node *temp;
        int x;
        if(head==NULL)    //When there are no elements in list
        {
            cout<<endl<<"LIST IS EMPTY";return 0;
        }
        else if(head->next==NULL) //when there is single element in list
        {
            temp=head;
            head=NULL;
            x=temp->data;
            delete temp;
            return x;
        }
        else
        {
            temp=head;
            head=temp->next;
            x=temp->data;
            delete temp;
            return x;
        }
}
int SLL::removeAtEnd()
{
	node *temp,*p;
	int x;
	if(head==NULL)    //When there are no elements in list
	{
		cout<<endl<<"LIST IS EMPTY";
		return 0;
	}
	else if(head->next==NULL) //when there is single element in list
	{
		temp=head;
		head=NULL;
		x=temp->data;
		delete temp;
		return x;
	}
	else
	{
		p=head;
		while(p->next->next!=NULL)
			p=p->next;
		temp=p->next;
		p->next=NULL;
		x=temp->data;
		delete temp;
		return x;
	}
}
int SLL::removeAtMiddle(int pos)
{
	node *temp,*p;
	int i,x;
	if(head==NULL)    //When there are no elements in list
	{
		cout<<endl<<"LIST IS EMPTY";
		return 0;
	}
	else
	{
		p=head;
		for(i=1,p=head;i<pos-1 && p!=NULL;i++)
			p=p->next;
		if(i==pos-1)
		{
			temp=p->next;
			p->next=temp->next;
			x=temp->data;
			delete temp;
```

```
                    return x;
                }
                else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
        }
}
void SLL::display()
{
        node *p;
        p=head;
        while(p!=NULL)
        {
                cout<<p->data<<"\t";
                p=p->next;
        }
}
void SLL::reverseList()
{
        SLL newone;
        while(head!=NULL)
        {
                int x=removeAtBegin();
                newone.insertAtBegin(x);
        }
        head=newone.head;
}
int main()
{
        int ch,x,p;
        SLL ob;
        while(1)
        {
                cout<<endl<<"1.Insert at begin"<<endl<<"2.Insert at end"<<endl<<"3.Insert
at middle";
                cout<<endl<<"4.Remove at begin"<<endl<<"5.Remove at
end"<<endl<<"6.Remove at middle";
                cout<<endl<<"7.Reverse the
List"<<endl<<"8.Display"<<endl<<"9.Exit"<<endl<<"Enter Your Choice:";
                cin>>ch;
                switch(ch)
                {
                        case 1:cout<<endl<<"Enter an element to insert:";
                                cin>>x;
                                ob.insertAtBegin(x);
                                break;
                        case 2:cout<<endl<<"Enter an element to insert:";
                                cin>>x;
                                ob.insertAtEnd(x);
```

```
				break;
			case 3:cout<<endl<<"Enter a position and an element to insert:";
				cin>>p>>x;
				ob.insertAtMiddle(p,x);
				break;
			case 4: cout<<endl<<"Removed element is:"<<ob.removeAtBegin();
				break;
			case 5:cout<<endl<<"Removed element is:"<<ob.removeAtEnd();
				break;
			case 6:cout<<endl<<"Enter position value";
				cin>>p;
				cout<<endl<<"Removed element
is:"<<ob.removeAtMiddle(p);
				break;
			case 7:ob.reverseList();
				break;
			case 8:ob.display();
				break;
			case 9:return 0;
			default:cout<<endl<<"WRONG CHOICE";
		}
	}
}
```

DOUBLY LINKED LIST:

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

struct node
{
	public:
	int data;
	node *prev;
	node *next;
};

class DLL
{
	node *head;
	public:
	DLL() {head=NULL;}//constructor to initialize
	void insertAtBegin(int x);
	void insertAtEnd(int x);
	void insertAtMiddle(int pos,int x);
	int removeAtBegin();
```

```
        int removeAtEnd();
        int removeAtMiddle(int pos);
        void display();
        void reverseList();
};
void DLL::insertAtBegin(int x)
{
        node *temp=new node;
        temp->data=x;
        temp->prev=temp->next=NULL;
        if(head==NULL)
        {
                head=temp;
        }
        else
        {
                head->prev=temp;
                temp->next=head;
                head=temp;
        }
}
void DLL::insertAtEnd(int x)
{
        node *temp=new node;
        node *p;
        temp->data=x;
        temp->prev=temp->next=NULL;
        if(head==NULL)
        {
                head=temp;
        }
        else
        {
                p=head;
                while(p->next!=NULL)
                        p=p->next;
                p->next=temp;
                temp->prev=p;
        }
}
void DLL::insertAtMiddle(int pos,int x)
{
        node *temp=new node;
        temp->data=x;
        temp->prev=temp->next=NULL;
        node *p;
        int i=0;
```

```
		for(i=1,p=head;i<pos-1 && p!=NULL;i++)
				p=p->next;
		if(i==pos-1)
		{
			temp->next=p->next;
			temp->prev=p;
			p->next=temp;
			p->next->prev=temp;
		}
		else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
}
int DLL::removeAtBegin()
{
		node *temp;
		int x;
		if(head==NULL)    //When there are no elements in list
		{
			cout<<endl<<"LIST IS EMPTY";return 0;
		}
		else if(head->next==NULL) //when there is single element in list
		{
			temp=head;
			head=NULL;
			x=temp->data;
			delete temp;
			return x;
		}
		else
		{
			temp=head;
			head=temp->next;
			head->prev=NULL;
			x=temp->data;
			delete temp;
			return x;
		}
}
int DLL::removeAtEnd()
{
		node *temp;
		int x;
		if(head==NULL)    //When there are no elements in list
		{
			cout<<endl<<"LIST IS EMPTY";
			return 0;
		}
		else if(head->next==NULL) //when there is single element in list
```

```
	{
		temp=head;
		head=NULL;
		x=temp->data;
		delete temp;
		return x;
	}
	else
	{
		temp=head;
		while(temp->next!=NULL)
			temp=temp->next;
		temp->prev->next=NULL;
		x=temp->data;
		delete temp;
		return x;
	}
}
int DLL::removeAtMiddle(int pos)
{
	node *temp;
	int i,x;
	if(head==NULL)    //When there are no elements in list
	{
		cout<<endl<<"LIST IS EMPTY";
		return 0;
	}
	else
	{
		temp=head;
		for(i=1,temp=head;i<pos && temp!=NULL;i++)
			temp=temp->next;
		if(i==pos)
		{
			temp->next->prev=temp->prev;
			temp->prev->next=temp->next;
			x=temp->data;
			delete temp;
			return x;
		}
		else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
	}
}
void DLL::display()
{
	node *p;
	p=head;
	while(p!=NULL)
	{
		cout<<p->data<<"\t";
		p=p->next;
	}
}
int main()
{
	int ch,x,p;
	DLL ob;
	while(1)
	{
		cout<<endl<<"1.Insert at begin"<<endl<<"2.Insert at end"<<endl<<"3.Insert 
at middle";
		cout<<endl<<"4.Remove at begin"<<endl<<"5.Remove at 
end"<<endl<<"6.Remove at middle";
		cout<<endl<<"7.Display"<<endl<<"8.Exit"<<endl<<"Enter Your Choice:";
		cin>>ch;
		switch(ch)
		{
			case 1:cout<<endl<<"Enter an element to insert:";
					cin>>x;
					ob.insertAtBegin(x);
					break;
			case 2:cout<<endl<<"Enter an element to insert:";
					cin>>x;
					ob.insertAtEnd(x);
					break;
			case 3:cout<<endl<<"Enter a position and an element to insert:";
					cin>>p>>x;
					ob.insertAtMiddle(p,x);
					break;
			case 4: cout<<endl<<"Removed element is:"<<ob.removeAtBegin();
					break;
			case 5:cout<<endl<<"Removed element is:"<<ob.removeAtEnd();
					break;
			case 6:cout<<endl<<"Enter position value";
					cin>>p;
					cout<<endl<<"Removed element 
is:"<<ob.removeAtMiddle(p);
					break;
			case 7:ob.display();
					break;
			case 8:return 0;
			default:cout<<endl<<"WRONG CHOICE";
		}
	}
```

```
}
```

CIRCULAR SINGLE LINKED LIST:

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

struct node
{
	public:
	int data;
	node *next;
};

class CLL
{
	node *last;
	public:
	CLL() {last=NULL;}//constructor to initialize
	void insertAtBegin(int x);
	void insertAtEnd(int x);
	void insertAtMiddle(int pos,int x);
	int removeAtBegin();
	int removeAtEnd();
	int removeAtMiddle(int pos);
	void display();
};
void CLL::insertAtBegin(int x)
{
	node *temp=new node;
	temp->data=x;
	temp->next=NULL;
	if(last==NULL)
	{
		last=temp;
		temp->next=temp;
	}
	else
	{
		temp->next=last->next;
		last->next=temp;
	}
}
void CLL::insertAtEnd(int x)
{
	node *temp=new node;
```

```
        node *p;
        temp->data=x;
        temp->next=NULL;
        if(last==NULL)
        {
                last=temp;
                temp->next=temp;
        }
        else
        {
                temp->next=last->next;
                last->next=temp;
                last=temp;
        }
}
void CLL::insertAtMiddle(int pos,int x)
{
        node *temp=new node;
        temp->data=x;
        temp->next=NULL;
        node *p;
        int i=0;
        for(i=1,p=last->next;i<pos-1 && p->next!=last;i++)
                        p=p->next;
        if(i==pos-1)
        {
                temp->next=p->next;
                p->next=temp;
        }
        else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
}
int CLL::removeAtBegin()
{
        node *temp;
        int x;
        if(last==NULL)    //When there are no elements in list
        {
                cout<<endl<<"LIST IS EMPTY";return 0;
        }
        else if(last->next==last) //when there is single element in list
        {
                temp=last;
                last=NULL;
                x=temp->data;
                delete temp;
                return x;
        }
```

```
	else
	{
		temp=last->next;
		last->next=last->next->next;
		x=temp->data;
		delete temp;
		return x;
	}
}
int CLL::removeAtEnd()
{
	node *temp,*p;
	int x;
	if(last==NULL)    //When there are no elements in list
		cout<<endl<<"LIST IS EMPTY";
	else if(last->next==last) //when there is single element in list
	{
		temp=last;
		last=NULL;
		x=temp->data;
		delete temp;
		return x;
	}
	else
	{
		p=last->next;
		while(p->next!=last)
			p=p->next;
		temp=last;
		p->next=temp->next;
		last=p;
		x=temp->data;
		delete temp;
		return x;
	}
}
int CLL::removeAtMiddle(int pos)
{
	node *temp,*p;
	int i,x;
	if(last==NULL)    //When there are no elements in list
		cout<<endl<<"LIST IS EMPTY";
	else
	{
		p=last->next;
		for(i=1,p=last->next;i<pos-1 && p!=last;i++)
			p=p->next;
```

```
                if(i==pos-1)
                {
                        temp=p->next;
                        p->next=temp->next;
                        x=temp->data;
                        delete temp;
                        return x;
                }
                else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
        }
}
void CLL::display()
{
        node *p;
        p=last->next;
        while(p!=last)
        {
                cout<<p->data<<"\t";
                p=p->next;
        }
        cout<<p->data<<"\t";
}
int main()
{
        int ch,x,p;
        CLL ob;
        while(1)
        {
                cout<<endl<<"1.Insert at begin"<<endl<<"2.Insert at end"<<endl<<"3.Insert
at middle";
                cout<<endl<<"4.Remove at begin"<<endl<<"5.Remove at
end"<<endl<<"6.Remove at middle";
                cout<<endl<<"7.Display"<<endl<<"8.Exit"<<endl<<"Enter Your Choice:";
                cin>>ch;
                switch(ch)
                {
                        case 1:cout<<endl<<"Enter an element to insert:";
                                        cin>>x;
                                        ob.insertAtBegin(x);
                                        break;
                        case 2:cout<<endl<<"Enter an element to insert:";
                                        cin>>x;
                                        ob.insertAtEnd(x);
                                        break;
                        case 3:cout<<endl<<"Enter a position and an element to insert:";
                                        cin>>p>>x;
                                        ob.insertAtMiddle(p,x);
                                        break;
                        case 4: cout<<endl<<"Removed element is:"<<ob.removeAtBegin();
                                        break;
                        case 5:cout<<endl<<"Removed element is:"<<ob.removeAtEnd();
                                        break;
                        case 6:cout<<endl<<"Enter position value";
                                        cin>>p;
                                        cout<<endl<<"Removed element
is:"<<ob.removeAtMiddle(p);
                                        break;
                        case 7:ob.display();
                                        break;
                        case 8:return 0;
                        default:cout<<endl<<"WRONG CHOICE";
                }
        }
}
```

CIRCULAR DOUBLE LINKED LIST(USING HEAD & LAST)

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

struct node
{
        public:
        int data;
        node *prev;
        node *next;
};

class DCLL
{
        node *head;
        node *last;
        public:
        DCLL() {head=NULL;
        last=NULL;}
        void insertAtBegin(int x);
        void insertAtEnd(int x);
        void insertAtMiddle(int pos,int x);
        int removeAtBegin();
        int removeAtEnd();
        int removeAtMiddle(int pos);
        void display();
        void reverseList();
```

```
};
void DCLL::insertAtBegin(int x)
{
        node *temp=new node;
        temp->data=x;
        temp->prev=temp->next=NULL;
        if(head==NULL&&head==last)
        {
                head=temp;
                last=temp;
                temp->prev=temp->next=NULL;
        }
        else
        {
                temp->next=head;
                head->prev=temp;
                temp->prev=last;
                last->next=temp;
                head=temp;
        }
}
void DCLL::insertAtEnd(int x)
{
        node *temp=new node;
        node *p;
        temp->data=x;
        temp->prev=temp->next=NULL;
        if(head==NULL&&head==last)
        {
                head=temp;
                last=temp;
                temp->prev=temp->next=NULL;
        }
        else
        {
                last->next=temp;
                temp->prev=last;
                last=temp;
                head->prev=last;
                last->next=head;
        }
}
void DCLL::insertAtMiddle(int pos,int x)
{
        node *temp=new node;
        temp->data=x;
        temp->prev=temp->next=NULL;
        node *p;
        int i=1,p=head;
        while(i<pos-1)
        {
                p=p->next;
                i++;
        }
        if(i==pos-1)
        {
        temp->prev=p;
        temp->next=p->next;
        p->next->prev=temp;
        p->next=temp;
        }
        else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
}
int DCLL::removeAtBegin()
{
        node *temp;
        int x;
        if(head==NULL&&head==last)    //When there are no elements in list
        {
                cout<<endl<<"LIST IS EMPTY";return 0;
        }
        else if(head->next==head) //when there is single element in list
        {
                temp=NULL;
                head=NULL;
                x=temp->data;
                delete temp;
                return x;
        }
        else
        {
                head=head->next;
                head->prev=last;
                last->next=head;
                x=temp->data;
                delete temp;
                return x;
        }
}
int DCLL::removeAtEnd()
{
        node *temp;
        int x;
        if(head==NULL&&head==last)    //When there are no elements in list
```

```
	{
		cout<<endl<<"LIST IS EMPTY";
		return 0;
	}
	else if(head->next==head) //when there is single element in list
	{
		temp=NULL;
		head=NULL;
		x=temp->data;
		delete temp;
		return x;
	}
	else
	{
		temp=last;
		last=temp->prev;
		last->next=head;
		head->prev=last;
		x=temp->data;
		delete temp;
		return x;
	}
}
int DCLL::removeAtMiddle(int pos)
{
	node *temp;
	int i,x;
	if(head==NULL&&head==last)    //When there are no elements in list
	{
		cout<<endl<<"LIST IS EMPTY";
		return 0;
	}
	else
	{
		temp=head;
		for(i=1,temp=head;i<pos;i++)
			temp=temp->next;
		if(i==pos)
		{
			p->prev->next=p->next;
			p->next->pev=p->prev;
			x=temp->data;
			delete temp;
			return x;
		}
		else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
	}
```

```
}
void DCLL::display()
{
	node *p;
	p=head;
	while(p->next!=p)
	{
		cout<<p->data<<"\t";
		p=p->next;
	}
}
int main()
{
	int ch,x,p;
	DCLL ob;
	while(1)
	{
		cout<<endl<<"1.Insert at begin"<<endl<<"2.Insert at end"<<endl<<"3.Insert 
at middle";
		cout<<endl<<"4.Remove at begin"<<endl<<"5.Remove at 
end"<<endl<<"6.Remove at middle";
		cout<<endl<<"7.Display"<<endl<<"8.Exit"<<endl<<"Enter Your Choice:";
		cin>>ch;
		switch(ch)
		{
			case 1:cout<<endl<<"Enter an element to insert:";
					cin>>x;
					ob.insertAtBegin(x);
					break;
			case 2:cout<<endl<<"Enter an element to insert:";
					cin>>x;
					ob.insertAtEnd(x);
					break;
			case 3:cout<<endl<<"Enter a position and an element to insert:";
					cin>>p>>x;
					ob.insertAtMiddle(p,x);
					break;
			case 4: cout<<endl<<"Removed element is:"<<ob.removeAtBegin();
					break;
			case 5:cout<<endl<<"Removed element is:"<<ob.removeAtEnd();
					break;
			case 6:cout<<endl<<"Enter position value";
					cin>>p;
					cout<<endl<<"Removed element 
is:"<<ob.removeAtMiddle(p);
					break;
			case 7:ob.display();
```

```
                                        break;
                        case 8:return 0;
                        default:cout<<endl<<"WRONG CHOICE";
                }
        }
}
```

CIRCULAR DOUBLE LINKED LIST(USING LAST)

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

struct node
{
        public:
        int data;
        node *prev;
        node *next;
};
int counter = 0;
class DCLL
{
        node *last;
        public:
        DCLL() {
        last=NULL;}
        node *create_node(int);
        void insertAtBegin();
        void insertAtEnd();
        void insertAtMiddle();
        int removeAtBegin();
        int removeAtEnd();
        int removeAtMiddle();
        void display();
        void reverseList();
};
node *DCLL::create_node(int x)
{
  counter++;
  struct node *temp;
  temp = new(struct node);
  temp->data = x;
  temp->next = NULL;
  temp->prev = NULL;
  return temp;
```

```
}
void DCLL::insertAtBegin()
{
        int x;
        cout<<endl<<"Enter an element to insert:";
        cin>>x;
        struct node *temp;
   temp = create_node(x);
        if(last==NULL)
        {
                last=temp;
                temp->prev=temp->next=temp;
        }
        else
        {
                temp->next=last->next;
                temp->prev=last;
                temp->next->prev=temp;
        }
}
void DCLL::insertAtEnd()
{
        int x;
        cout<<endl<<"Enter an element to insert:";
        cin>>x;
        struct node *temp;
   temp = create_node(x);
        if(last==NULL)
        {
                last=temp;
                temp->prev=temp->next=temp;
        }
        else
        {
                temp->next=last->next;
                temp->prev=last;
                last->next=temp;
                last=temp;
        }
}
void DCLL::insertAtMiddle()
{
        int x,pos;
        cout<<endl<<"Enter a position and an element to insert:";
        cin>>pos>>x;
        struct node *temp,*p;
   temp = create_node(x);
```

```
        int i=1;
        p=last->next;
        while(i<pos-1)
        {
                p=p->next;
                i++;
        }

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

}
int DCLL::removeAtBegin()
{
        node *temp;
        int x;
        if(last==NULL)    //When there are no elements in list
        {
                cout<<endl<<"LIST IS EMPTY";return 0;
        }
        else if(last->next==last) //when there is single element in list
        {
                temp=NULL;
                head=NULL;
                x=temp->data;
                delete temp;
                return x;
        }
        else
        {
                temp=last->next;
                temp->next->prev=last;
                last->next=temp->next;
                x=temp->data;
                delete temp;
                return x;
        }
}
int DCLL::removeAtEnd()
{
        node *temp;
        int x;
        if(last==NULL)    //When there are no elements in list
        {
                cout<<endl<<"LIST IS EMPTY";
                return 0;
        }
        else if(last->next==last) //when there is single element in list
        {
                temp=NULL;
                last=NULL;
                x=temp->data;
                delete temp;
                return x;
        }
        else
        {
                temp=last;
                last=temp->prev;
                last->next=temp->next;
                temp->next->prev=last;
                x=temp->data;
                delete temp;
                return x;
        }
}
int DCLL::removeAtMiddle()
{
        int pos;
        cout<<endl<<"Enter position value";
        cin>>pos;
        node *p;
        int i,x;
        if(last==NULL)    //When there are no elements in list
        {
                cout<<endl<<"LIST IS EMPTY";
                return 0;
        }
        else
        {
                p=last->next;
                for(i=1,p=last->next;i<pos;i++)
                        p=p->next;
                if(i==pos)
                {
                        p->prev->next=p->next;
                        p->next->prev=p->prev;
                        x=p->data;
                        delete p;
                        return x;
                }
                else cout<<endl<<"WRONG POSITION NUMBER"<<endl;
```

```
		}
}
void DCLL::display()
{
	struct node *p;
	int i;
	p=last->next;
	for (i = 0;i < counter-1;i++)
	{
		cout<<p->data<<"<->";
		p=p->next;
	}
	cout<<p->data;
}
int main()
{
	int ch;
	DCLL ob;
	while(1)
	{
		cout<<endl<<"1.Insert at begin"<<endl<<"2.Insert at end"<<endl<<"3.Insert
at middle";
		cout<<endl<<"4.Remove at begin"<<endl<<"5.Remove at
end"<<endl<<"6.Remove at middle";
		cout<<endl<<"7.Display"<<endl<<"8.Exit"<<endl<<"Enter Your Choice:";
		cin>>ch;
		switch(ch)
		{
			case 1:ob.insertAtBegin();
					break;
			case 2:
					ob.insertAtEnd();
					break;
			case 3:
					ob.insertAtMiddle();
					break;
			case 4: cout<<endl<<"Removed element is:"<<ob.removeAtBegin();
					break;
			case 5:cout<<endl<<"Removed element is:"<<ob.removeAtEnd();
					break;
			case 6:
					cout<<endl<<"Removed element
is:"<<ob.removeAtMiddle();
					break;
			case 7:ob.display();
					break;
			case 8:return 0;
```

```
                              default:cout<<endl<<"WRONG CHOICE";
                    }
          }
}
```

REVERSING A DOUBLE LINKED LIST:

```
#include<iostream>
#include<cstdlib>

using namespace std;

struct node
{
   int data;
   struct node *nptr; //next pointer
   struct node *pptr; //previous pointer
};

class RDLL
{
          node *hptr;
          public:
          RDLL() {hptr=NULL;}//constructor to initialize
          void insertNode(int pos,int x);
          void deleteNode(int pos);
          void reverseList();
          void print();
};

void RDLL::insertNode(int pos, int x)
{
   struct node *temp=new node;
   if(temp==NULL)
      cout<<"Insertion not possible\n";
   temp->data=x;
   if(pos==1 && hptr==NULL)
   {
      temp->pptr=NULL;
      temp->nptr=NULL;
      hptr=temp;
   }
   else if(pos==1)
   {
      temp->nptr=hptr;
      hptr=temp;
```

```
        temp->nptr->pptr=temp;
        temp->pptr=NULL;
    }
    else
    {
        int i=1;
        struct node *thptr=hptr;
        while(i<pos-1)
        {
            thptr=thptr->nptr;
            i++;
        }
        temp->nptr=thptr->nptr;
        temp->pptr=thptr;
        thptr->nptr=temp;
        thptr->nptr->pptr=thptr;
    }
}

void RDLL::deleteNode(int pos)
{
    if(hptr==NULL)
        cout<<"Deletion not possible\n";
    else
    {

        if(pos==1)
        {
            hptr=hptr->nptr;
        }
        else
        {
            int i=1;
            struct node *thptr=hptr;
            while(pos<i-1)
            {
                thptr=thptr->nptr;
            }
            thptr->nptr=thptr->nptr->nptr;
            if(thptr->nptr!=NULL)
                thptr->nptr->pptr=thptr;
        }

    }
}

void RDLL::reverseList()
```

```
{
   struct node *current=hptr;
   struct node *prev=NULL;
   while(current!=NULL)
   {
      current->pptr=current->nptr; //line 1
      current->nptr=prev;         //line 2
      prev=current;               //line 3
      current=current->pptr;      //line 4
   }
      hptr=prev;
}

void RDLL::print()
{
   struct node *thptr=hptr;
   while(thptr!=NULL)
   {
      cout<<thptr->data<<"\n";
      thptr=thptr->nptr;
   }
}

int main()
{
   int ch,p,x;
   RDLL ob;
   while(1)
          {
                    cout<<"1.INSERT A
NODE"<<"\n2.REVERSE"<<"\n3.DISPLAY"<<"\n4.EXIT"<<endl;
                    cout<<"Enter your choice:";
                    cin>>ch;
                    switch(ch)
                    {
                              case 1:cout<<endl<<"Enter a position and an element to insert:";
                                                            cin>>p>>x;
                                                            ob.insertNode(p,x);
                                                            break;
                                        case 2:ob.reverseList();
                                                            break;
                                        case 3:ob.print();
                                                            break;
                                        case 4:return 0;
                              }

                    }
}
```

REVERSING CIRCULAR LINKED LIST:

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

// Linked list node
struct Node {
   int data;
   Node* next;
};

// function to get a new node
Node* getNode(int data)
{
   // allocate memory for node
   Node* newNode = new Node;

   // put in the data
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

// Function to reverse the circular linked list
void reverse(Node** head_ref)
{
   // if list is empty
   if (*head_ref == NULL)
      return;

   // reverse procedure same as reversing a
   // singly linked list
   Node* prev = NULL;
   Node* current = *head_ref;
   Node* next;
   do {
      next = current->next;
      current->next = prev;
      prev = current;
      current = next;
   } while (current != (*head_ref));

   // adjusting the links so as to make the
   // last node point to the first node
   (*head_ref)->next = prev;
```

```
    *head_ref = prev;
}

// Function to print circular linked list
void printList(Node* head)
{
    if (head == NULL)
        return;

    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
}

// Driver program to test above
int main()
{
    // Create a circular linked list
    // 1->2->3->4->1
    Node* head = getNode(1);
    head->next = getNode(2);
    head->next->next = getNode(3);
    head->next->next->next = getNode(4);
    head->next->next->next->next = head;

    cout << "Given circular linked list: ";
    printList(head);

    reverse(&head);

    cout << "\nReversed circular linked list: ";
    printList(head);

    return 0;
}
```

RECURSION:
FACTORIAL OF A NUMBER EXAMPLE :

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

int factorial(int n)
{
    if (n == 0)
        return 1;
    return n * factorial(n - 1);
}

int main()
{
    int num;
            cout<<"Enter a number: "<<endl;
            cin>>num;
    cout << "Factorial of "
         << num << " is " << factorial(num) << endl;
    return 0;
}
```

FIBONACCI SERIES EXAMPLE:

```
#include <iostream>
using namespace std;
int fib(int x) {
   if((x==1)||(x==0)) {
      return(x);
   }else {
      return(fib(x-1)+fib(x-2));
   }
}
int main() {
   int x , i=0;
   cout << "Enter the number of terms of series : ";
   cin >> x;
   cout << "\nFibonnaci Series : ";
   while(i < x) {
      cout << " " << fib(i);
      i++;
   }
   return 0;
}
```