

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class demo
{
    int a[10],i,j,n,item,k;
public:
    void get();
    void insert();
    void del();
    void dis();
};

void demo::get()
{
    cout<<"\nEnter n";
    cin>>n;
    cout<<"\nEnter Array Element:";
    for(i=1;i<=n;i++)
        cin>>a[i];
}

void demo::insert()
{
    cout<<"\nEnter Position:";
    cin>>k;
    cout<<"\nEnter Item:";
    cin>>item;
    j=n;
    while(j>=k)
    {
        a[j+1]=a[j];
        j--;
    }
    a[k]=item;
    n++;
}

void demo::del()
{
    cout<<"\nEnter Position:";
    cin>>k;
    j=k;
    while(j<=n-1)
    {
        a[j]=a[j+1];
        j++;
    }
    n--;
}

void demo::dis()
{
    cout<<"\n Elements are\n";
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
```

```

demo d;
int ch;
d.get();
cout<<"\n1. Insert 2.Del 3.Dis 4. Exit\n";
while(ch!=4)
{
    cout<<"\n Enter choice";
    cin>>ch;
    switch(ch)
    {
        case 1: d.insert(); break;
        case 2: d.del(); break;
        case 3: d.dis(); break;
        case 4: exit(0);
    }
}
getch();
}

```

*/ Output */

Enter n 3

Enter Array Element:1 2 4

1. Insert 2.Del 3.Dis 4. Exit

Enter choice 3

Elements are

1 2 4

Enter choice 1

Enter Position: 2

Enter Item: 6

Enter choice 3

Elements are

1 6 2 4

Enter choice 2

Enter Position: 3

Enter choice 3

Elements are

1 6 4

Enter choice 4

Assignment Name: Program for matrix addition, subtraction, multiplication and transpose of matrix
Class: MCA I

Lab: MCA 414 LAB ON DSA

```
#include<iostream.h>
#include<conio.h>
```

```
class matrix
{
    int a[5][5],b[5][5],c[5][5],d[5][5],e[5][5],f[5][5];
    int p,q,i,j,k,n,m;

public:
    void get();
    void add();
    void sub();
    void trans();
    void mul();

};
```

```
void matrix::get()
{
    cout<<"\nEnter Number of Row & Column : \t";
    cin>>n>>m;

    cout<<"\nEnter the first Matrix:\n";

    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
            cin>>a[i][j];
    }
    cout<<"\nEnter Number of Row & Column : \t";
    cin>>p>>q;

    cout<<"\nEnter the Second Matrix:\n";

    for(i=0;i<p;i++)
    {
        for(j=0;j<q;j++)
            cin>>b[i][j];
    }
}
```

```
void matrix::add()
{
    cout<<"\nThe addition of two matrix is : \n";
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            c[i][j]=a[i][j]+b[i][j];
            cout<<c[i][j]<<"\t";
        }
        cout<<"\n";
    }
}
```

```

}

void matrix::sub()
{
    cout<<"\nThe Subtraction of two matrix is :\n";
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            d[i][j]=a[i][j]-b[i][j];
            cout<<d[i][j]<<"\t";
        }
        cout<<"\n";
    }
}

void matrix::trans()
{
    cout<<"\nThe Transpose of first matrix is :\n";

    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            e[i][j]=a[j][i];
            cout<<e[i][j]<<"\t";
        }
        cout<<"\n";
    }
}

void matrix::mul()
{
    cout<<"\nThe Matrix Multiplication is : \n";

    if(m==p)
    {
        for(i=0;i<n;i++)
        {
            for(j=0;j<q;j++)
            {
                c[i][j]=0;
                for(k=0;k<p;k++)
                {
                    c[i][j]=c[i][j]+a[i][k]*b[k][j];
                }
                cout<<c[i][j]<<"\t";
            }
            cout<<"\n";
        }
    }
    else
        cout<<"\n Matrix Multiplication not possible";
}

void main()
{
    clrscr();
    matrix m;
    m.get();
}

```

```

        m.add();
        m.sub();
        m.trans();
        m.mul();
        getch();
    }

    /* Output */

Enter Number of Row & Column :  3 3

Enter the first Matrix:
1 2 3
4 5 6
7 8 9

Enter Number of Row & Column :  3 3

Enter the first Matrix:
1 2 3
4 5 6
7 8 9

The addition of two matrix is :
2      4      6
8      10     12
14     16     18

The Substraction of two matrix is :
0      0      0
0      0      0
0      0      0

The Transpose of first matrix is :
1      4      7
2      5      8
3      6      9

The Matrix Multiplication is :
30     36     42
66     81     96
102    126    150

```

Assignment Name: Implement Stack for Integer/character perform different
operation on stack (push, pop, peep, change).

Class: MCA I

Lab: MCA 414 LAB ON DSA

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
int n;
class stack
{
    private:
        int s[10],top,ele,i; // char s[10] for character
    public:
        stack()
        {
            top=-1;
        }
        void push();
        void dis();
        void pop();
        void peep();
        void change();
};

void stack::push()
{
    if(top>=n-1)
        cout<<"\nStack is overflow:";
    else
    {
        cout<<"\nEnter element:";
        cin>>ele;
        top++;
        s[top]=ele;
    }
}

void stack::dis()
{
    if(top==--1)
    {
        cout<<"\n Stack is Empty";
    }
    else
    {
        cout<<"\nElements in stack are:\n";
        for(i=top;i>=0;i--)
            cout<<s[i]<<"\t";
    }
}

void stack::pop()
{
    if(top==--1)
    {
        cout<<"\nUnderflow";
    }
    else
    {
        cout<<"\nPop ele is "<<s[top];
        top--;
    }
}
```

```

    }
}

void stack::peek()
{
    cout<<"\nEnter position:";
    cin>>i;
    if((top-i+1)<0)
    {
        cout<<"\nUnderflow";
    }
    else
    {
        cout<<"\nPeep ele is "<<s[top-i+1];
    }
}

void stack::change()
{
    cout<<"\nEnter position ";
    cin>>i;
    if((top-i+1)<0)
    {
        cout<<"\nUnderflow";
    }
    else
    {
        int n; //char n; for character
        cout<<"\nEnter element:";
        cin>>n;
        s[top-i+1]=n;
    }
}

void main()
{
    clrscr();
    stack s;
    cout<<"Enter size of stack";
    cin>>n;
    int ch;
    cout<<"\n1. Push 2.Display 3.Pop 4.Peep 5.Change 6.Exit\n";
    while(ch!=6)
    {
        cout<<"\nEnter ch :";
        cin>>ch;
        switch(ch)
        {
            case 1: s.push(); break;
            case 2: s.dis(); break;
            case 3: s.pop();break;
            case 4: s.peek(); break;
            case 5: s.change(); break;
            case 6: exit(0);
        }
    }
    getch();
}

```

*/ Output */

Enter size of stack 3

1. Push 2.Display 3.Pop 4.Peep 5.Change 6.Exit

Enter ch :1

Enter element:10

Enter ch :1

Enter element:20

Enter ch :1

Enter element:30

Enter ch :1

Stack is overflow:

Enter ch :2

Elements in stack are:

30 20 10

Enter ch :3

Pop ele is 30

Enter ch :2

Elements in stack are:

20 10

Enter ch :4

Enter position:1

Peep ele is 20

Enter ch :

2

Elements in stack are:

20 10

Enter ch :5

Enter position 1

Enter element:80

Enter ch :2

Elements in stack are:

80 10

Enter ch : 6


```
#include<conio.h>
#include<iostream.h>
#include<process.h>
class stack
{
    int info, ele;
    stack *node,*link,*top;
public:
    stack()
    {
        top=NULL;
    }
    void insert();
    void del();
    void dis();
};

void stack::insert()
{
    node=new stack;
    cout<<"\nEnter Info:";
    cin>>ele;
    node->info=ele;
    node->link=NULL;
    if(top==NULL)
    {
        top=node;
    }
    else
    {
        node->link=top;
        top=node;
    }
}

void stack::del()
{
    if(top==NULL)
    {
        cout<<"\n Underflow";
    }
    else
    {
        cout<<"\nDeleted Element is :"<<top->info;
        top=top->link;
    }
}

void stack::dis()
{
    stack *move;
    move=top;
    while(move!=NULL)
    {
        cout<<"\t"<<move->info;
        move=move->link;
    }
}
```

```

}

void main()
{
    clrscr();
    int ch;
    stack s;
    cout<<"\n1.Insert 2.Show 3.Delete 4.Exit";
    while(ch!=4)
    {
        cout<<"\nEnter Choice";
        cin>>ch;
        switch(ch)
        {
            case 1: s.insert(); break;
            case 2: s.dis(); break;
            case 3: s.del(); break;
            case 4:exit(0);
        }
    }
    getch();
}

```

*/ Output */

```

1.Insert 2.Show 3.Delete 4.Exit
Enter Choice1

Enter Info:23

Enter Choice1

Enter Info:55

Enter Choice1

Enter Info:66

Enter Choice1

Enter Info:77

Enter Choice2
    77      66      55      23
Enter Choice3

Deleted Element is :77
Enter Choice2
    66      55      23
Enter Choice

```

Assignment Name: Implement Infix to Postfix operation using stack.

Class: MCA I

Lab: MCA 414 LAB ON DSA

```
#include<iostream.h>
#include<conio.h>
#include<string.h>

class convert
{
    char infix[20],postfix[20],s[20];
    int i,p,top;
public:
    convert()
    {
        top=-1;
        i=p=0;
        cout<<"\nEnter infix Expression:";
        cin>>infix;
        strcat(infix,"");
        s[++top]='(';
    }
    int precedance(char);
    void post();
    void display();
};

int convert::precedance(char ch)
{
    switch(ch)
    {
        case '^':return 3;
        case '*':return 2;
        case '/':return 2;
        case '+':return 1;
        case '-':return 1;
        default: return 0;
    }
}

void convert::post()
{
    char ch;
    while(top!=-1)
    {
        ch=infix[i++];
        if((ch>='A'&&ch<='Z')||(ch>='a'&&ch<='z')||(ch>='1'&&ch<='9'))
            postfix[p++]=ch;
        else if(ch=='(')
            s[++top]=ch;
        else if(ch=='+'||ch=='-'||ch=='*'||ch=='/'||ch=='^')
        {
            while(precedance(ch)<=precedance(s[top]))
                postfix[p++]=s[top--];
            s[++top]=ch;
        }
        else if(ch==')')
        {
            while(s[top]!='(')
                postfix[p++]=s[top--];
            top--;
        }
    }
}
```

```

        else
            cout<<"\nWrong string";
    }
    postfix[p]='\0';
}

void convert::display()
{
    cout<<"\nPostfix Expression is :"<<postfix;
}

void main()
{
    clrscr();
    convert c;
    c.post();
    c.display();
    getch();
}

```

*/ Output */

Enter infix Expression: (a*b-(c+d/e^f)*h)

Postfix Expression is :ab*cdef^/+h*-

Enter infix Expression:a+2*5

Postfix Expression is :a25*+

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
int m;
class queue
{
    int f,r,q[10],n,i;    //char q[10],n  for character
public:
    queue()
    {
        f=r=0;
    }
    void insert();
    void del();
    void dis();
};

void queue::insert()
{
    if(r==m)
        cout<<"\nOverflow";
    else
    {
        cout<<"\nEnter Element in Queue=";
        cin>>n;
        if(f==0)
            f=1;
        r++;
        q[r]=n;
    }
}

void queue::del()
{
    if(f==0)
    {
        cout<<"\nUnderflow";
    }
    else
    {
        int n;
        n=q[f];
        if(f==r)
            f=r=0;
        else
            f++;
        cout<<"\nDeleted element is "<<n;
    }
}

void queue::dis()
{
    if(f==0)
        cout<<"\nUnderflow";
    else
```

```

    {
        cout<<"\nElements in queue are:";
        for(i=f;i<=r;i++)
            cout<<q[i]<<"\t";
    }
}

void main()
{
    clrscr();
    queue q;
    int ch;
    cout<<"Enter size of queue";
    cin>>m;
    cout<<"\n 1.insert 2.display 3.delete 4. exit \n";
    while(ch!=4)
    {
        cout<<"\nEnter ch:";
        cin>>ch;
        switch(ch)
        {
            case 1: q.insert(); break;
            case 2: q.dis(); break;
            case 3: q.del(); break;
            case 4:exit(0);
        }
    }
    getch();
}

```

*/ Output */

Enter size of queue 3

1.insert 2.display 3.delete 4. exit

Enter ch:3

Underflow

Enter ch:1

Enter Element in Queue=10

Enter ch:1

Enter Element in Queue=20

Enter ch:1

Enter Element in Queue=30

Enter ch:1

Overflow

Enter ch:2

Elements in queue are:10 20 30

Enter ch:3

Deleted element is 10

Enter ch:2

Elements in queue are:20 30

Enter ch:4

```
#include<conio.h>
#include<iostream.h>
#include<process.h>
class queue
{
    int info, ele, c;
    queue *node, *link, *start, *move;
public:
    queue()
    {
        start=NULL;
        c=0;
    }
    void insert();
    void del();
    void dis();
};

void queue::insert()
{
    node=new queue;
    if(c<3)
    {
        cout<<"\nEnter Info:";
        cin>>ele;
        node->info=ele;
        node->link=NULL;
        if(start==NULL)
        {
            start=node;
            c++;
            return;
        }
        else
        {
            move=start;
            while(move->link!=NULL)
            move=move->link;
            move->link=node;
            c++;
        }
    }
    else
        cout<<"\n Overflow";
}

void queue::del()
{
    move=start;
    if(move!=NULL)
    {
        move=move->link;
        cout<<"\nDeleted Element is :"<<start->info;
        start=move;
    }
    else
        cout<<"\nUnderflow";
}

void queue::dis()
```

```

{
    move=start;
    if (move==NULL)
    {
        cout<<"\n Queue is empty ";
        return;
    }
    else
    {
        while (move!=NULL)
        {
            cout<<move->info<<"\t";
            move=move->link;
        }
    }
}
void main()
{
    clrscr();
    int ch;
    queue s;
    cout<<"\n1.Insert 2.Show 3.Delete 4.Exit";
    while(ch!=4)
    {
        cout<<"\nEnter Choice";
        cin>>ch;
        switch(ch)
        {
            case 1: s.insert();break;
            case 2: s.dis();break;
            case 3: s.del();break;
            case 4:exit(0);
        }
    }
    getch();
}
*/ Output */
1.Insert 2.Show 3.Delete 4.Exit
Enter Choice2

```

Queue is empty
Enter Choice1

Enter Info:10

Enter Choice1

Enter Info:20

Enter Choice1

Enter Info:30

Enter Choice1

Overflow

Enter Choice2

10 20 30

Enter Choice3

Deleted Element is :10

Enter Choice2

20 30

Assignment Name: Implement Circular Queue, perform different operation of
circular queue (push ,pop, show)

Class: MCA I

Lab: MCA 414 LAB ON DSA

```
#include<iostream.h>
#include<conio.h>
class queue
{
    int a[5],r,f;
public:

    queue()
    {
        f=r=-1;
    }
    void push();
    void pop();
    void show();
};

void queue::push()
{
    int item;

    if(f==0 && r==4 || f==r+1)
    {
        cout<<"\n Overflow";
    }
    else
    {
        if(r==4)
            r=-1;
        r++;
        cout<<"\nEnter item :";
        cin>>item;
        a[r]=item;

        if(f==-1)
        {
            f=0;
        }
    }
}

void queue::pop()
{
    if(f==-1)
    {
        cout<<"\n Underflow";
    }
    else
    {
        cout<<"\nDeleted element is :"<<a[f];
        if(f==r)
        {
            f=-1;
            r=-1;
        }
        else
        {

```

```

        if(f==4)
            f=0;
        else
            f++;
    }
}

void queue::show()
{
    if(f==-1)
    {
        cout<<"\nEmpty :";
    }
    else if(f<=r)
    {
        for(int i=f;i<=r;i++)
        {
            cout<<"\n"<<a[i];
        }
    }
    else
    {
        for(int i=f;i<=4;i++)
        {
            cout<<"\n"<<a[i];
        }
        for(int j=0;j<=r;j++)
        {
            cout<<"\n"<<a[j];
        }
    }
}

void main()
{
    queue s;
    int ch;
    clrscr();

    do
    {
        cout<<"\n 1: Push 2: Pop 3:show 4:exit ";
        cout<<"\nEnter choice";
        cin>>ch;

        switch(ch)
        {
            case 1: s.push(); break;
            case 2: s.pop(); break;
            case 3: s.show(); break;
            default: cout<<"\n Wrong Choice";
        }
    }while(ch<=3);
}

```

*/ Output */

1: Push 2: Pop 3:show 4:exit
Enter choice1

Overflow

1: Push 2: Pop 3:show 4:exit
Enter choice3

10
20
30
40
50

1: Push 2: Pop 3:show 4:exit
Enter choice2

Deleted element is :10

1: Push 2: Pop 3:show 4:exit
Enter choice2

Deleted element is :20

1: Push 2: Pop 3:show 4:exit
Enter choice3

30
40
50

1: Push 2: Pop 3:show 4:exit
Enter choice1

Enter item :44

1: Push 2: Pop 3:show 4:exit
Enter choice1

Enter item :55

1: Push 2: Pop 3:show 4:exit
Enter choice1

Overflow

1: Push 2: Pop 3:show 4:exit
Enter choice3

30
40
50
44
55

1: Push 2: Pop 3:show 4:exit
Enter choice 4

Assignment Name: Perform Insert, Display, delete, search, sum operation
on Linked list.

Class: MCA I

Lab: MCA 414 LAB ON DSA

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class node
{
    int info,item,s;
    node *link;
public:
    void insert();
    void dis();
    void del();
    void search();
    void sum();
};
node *move,*start=NULL,*temp;

void node::insert()
{
    cout<<"\nEnter the item:";
    cin>>item;
    node *nodel=new node;
    nodel->link=NULL;
    nodel->info=item;
    if(start==NULL)
        start=nodel;
    else
    {
        move=start;
        while(move->link!=NULL)
            move=move->link;
        move->link=nodel;
    }
}

void node::dis()
{
    node *x;
    x=start;
    cout<<"\n Elements in LL are:";
    while(x!=NULL)
    {
        cout<<"\t"<<x->info;
        x=x->link;
    }
}

void node::sum()
{
    node *x;
    x=start;
    s=0;
    while(x!=NULL)
    {
        s=s+x->info;
        x=x->link;
    }
}
```

```

        cout<<"\nSum of node is"<<s;
    }

void node::del()
{
    temp=start;
    if(temp!=NULL)
    {
        temp=temp->link;
        cout<<"\nDeleted node is"<<start->info;
        start=temp;
    }
    else
        cout<<"\n List is empty:";
}

void node::search()
{
    int c=0,f=0,d;
    cout<<"\nEnter item";
    cin>>item;
    temp=start;
    while(temp!=NULL)
    {
        c++;
        if(temp->info==item)
        {
            f=1;
            d=c;
            break;
        }
        temp=temp->link;
    }
    if(f==1)
        cout<<"\nElement is found at position "<<d;
    else
        cout<<"\nElement is not found";
}

void main()
{
    clrscr();
    node n;
    int ch;
    cout<<"\n1.Insert  2.Display 3. Delete 4.Search 5.Sum 6.Exit\n";

    do
    {
        cout<<"\nEnter choice";
        cin>>ch;
        switch(ch)
        {
            case 1: n.insert(); break;
            case 2: n.dis(); break;
            case 3: n.del(); break;
            case 4: n.search(); break;
            case 5: n.sum(); break;
            case 6: exit(0);
        }
    }while(ch!=6);
    getch();
}

```

*/ Output */

1.Insert 2.Display 3. Delete 4.Search 5.Sum 6.Exit

Enter choice1

Enter the item:10

Enter choice1

Enter the item:20

Enter choice1

Enter the item:30

Enter choice2

Elements in LL are: 10 20 30

Enter choice3

Deleted node is10

Enter choice2

Elements in LL are: 20 30

Enter choice5

Sum of node is50

Enter choice4

Enter item30

Element is found at position 2

Enter choice4

Enter item19

Element is not found

Enter choice 6

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class node
{
    int info,c,j;
    node *left,*right;
public:
    void insert();
    void display();
    void del();
};

node *start=NULL,*temp=NULL,*move=NULL, *temp1=NULL;
void node::insert()
{
    int item;
    node *p=new node;
    cout<<"\nEnter element:";
    cin>>item;
    p->info=item;
    p->left=NULL;
    p->right=NULL;
    if(start==NULL)
    {
        start=p;
        return;
    }
    else
    {
        temp=start;
        while(temp->right!=NULL)
            temp=temp->right;
        temp->right=p;
        p->left=temp;
    }
}

void node::display()
{
    move=start;
    if(move==NULL)
    {
        cout<<"\n LL Empty:";
        return;
    }
    else
    {
        cout<<"\n node in DLL are :";
        while(move!=NULL)
        {
            cout<<move->info<<"\t";
            move=move->right;
        }
    }
}
```

```

void node::del()
{
    if(start==NULL)
    {
        cout<<"\n LL Empty:";
        return;
    }
    temp=start;
    start=temp->right;
    start->left=NULL;
    temp->right=NULL;
    cout<<"\n deleted element is"<<temp->info;
}

void main()
{
    clrscr();
    node n;
    int ch;
    cout<<"\n1. Insert 2. Display 3.Delete 4. Exit";
    while(ch!=4)
    {
        cout<<"\nEnter choice";
        cin>>ch;
        switch(ch)
        {
            case 1: n.insert(); break;
            case 2: n.display(); break;
            case 3: n.del(); break;
            case 4: exit(0);
        }
    }
    getch();
}

```

*/ Output */

```

1. Insert 2. Display 3.Delete 4. Exit
Enter choice2

```

```

LL Empty:
Enter choice1

```

```

Enter element:10

```

```

Enter choice1

```

```

Enter element:20

```

```

Enter choice1

```

```

Enter element:30

```

```

Enter choice2

```

```

node in DLL are :10    20    30
Enter choice3

```

```

deleted element is10
Enter choice2

```



```
node in DLL are :20    30
Enter choice3
```

```
deleted element is20
Enter choice3
```

```
deleted element is30
Enter choice2
```

```
LL Empty:
Enter choice3
```

```
LL Empty:
Enter choice
```

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class node
{
    int info,c,i;
    node *link;
public:
    node()
    {
        c=0;
    }
    void insert();
    void display();
    void del();
};

node *start=NULL, *temp=NULL, *move=NULL, *temp1=NULL;
void node::insert()
{
    int item;
    node*p=new node;
    cout<<"\nEnter Element:";
    cin>>item;
    p->info=item;
    p->link=NULL;
    if(start==NULL)
    {
        start=p;
        p->link=start;
        c++;
    }
    else
    {
        temp=start;
        while(temp->link!=start)
            temp=temp->link;
        temp->link=p;
        p->link=start;
        c++;
    }
}

void node::display()
{
    if(start==NULL)
    {
        cout<<"\n LL empty";
        return;
    }
    node *temp;
    temp=start;
    move=start->link;
    cout<<temp->info;
    while(move!=start)
    {
```

```

        cout<<"->"<<move->info;
        move=move->link;
    }
    cout<<"\n Number of nodes in CLL are : "<<c;
}

void node::del()
{
    int pos;
    cout<<"\nEnter Position:";
    cin>>pos;
    if(c==1)
    {
        start=NULL;
    }
    if(start==NULL)
    {
        cout<<"\n LL Empty:";
        return;
    }

    if(pos>c||pos<1)
    {
        cout<<"\nInvalid Position";
        return;
    }

    if(pos==1)
    {
        temp=start;
        while(temp->link!=start)
        temp=temp->link;
        temp1=start;
        start=start->link;
        temp->link=start;
        cout<<"\nDeleted Element is "<<temp1->info;
        delete(temp1);
        c--;
    }
    else
    {
        temp=start;
        i=1;
        while(i<pos-1)
        {
            temp=temp->link;
            i++;
        }

        temp1=temp->link;
        temp->link=temp1->link;
        cout<<"\nDeleted element is"<<temp1->info;
        delete(temp1);
        c--;
    }
}

void main()
{
    clrscr();
    node n;
    int ch;

```

```

cout<<"\n 1.Insert 2.Display 3.Delete 4.Exit";
while(ch!=4)
{
    cout<<"\n Enter Choice";
    cin>>ch;
    switch(ch)
    {
        case 1: n.insert(); break;
        case 2: n.display(); break;
        case 3: n.del(); break;
        case 4: exit(0);
    }
    }getch();
}

*/ Output */

```

```

1.Insert 2.Display 3.Delete 4.Exit
Enter Choice1

```

```

Enter Element:10

```

```

Enter Choice1

```

```

Enter Element:20

```

```

Enter Choice2

```

```

10->20

```

```

Number of nodes in CLL are :2

```

```

Enter Choice3

```

```

Enter Position:2

```

```

Deleted element is20

```

```

Enter Choice2

```

```

10

```

```

Number of nodes in CLL are :1

```

```

Enter Choice3

```

```

Enter Position:1

```

```

LL Empty:

```

```

Enter Choice2

```

```

LL empty

```

```

Enter Choice 4

```

Assignment Name: Implementation of Polynomial Addition / Subtraction (using Array)

Class: MCA I

Lab: MCA 414 LAB ON DSA

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>

class polyexpr
{
    int pe1[10],pe2[10],pe3[10];
    int order;
public:
    polyexpr(int);
    void read_polyexpr1();
    void read_polyexpr2();
    void add_polyexpr();
    void sub_polyexpr();
    void view_polyexpr();
};

polyexpr::polyexpr(int para)
{
    order = para;
}

void polyexpr::read_polyexpr1()
{
    cout<<endl<<"Enter poly exp 1 : ";
    for (int i=order;i>=0;i--)
    {
        cout<<endl<<"Enter coeff of x^"<<i<<" : ";
        cin>>pe1[i];
    }
}

void polyexpr::read_polyexpr2()
{
    cout<<endl<<"Enter poly exp 2 : ";
    for (int i=order;i>=0;i--)
    {
        cout<<endl<<"Enter coeff of x^"<<i<<" : ";
        cin>>pe2[i];
    }
}

void polyexpr::add_polyexpr()
{
    for (int i=order;i>=0;i--)
        pe3[i]=pe1[i]+pe2[i];
    view_polyexpr();
}

void polyexpr::sub_polyexpr()
{
    for (int i=order;i>=0;i--)
        pe3[i]=pe1[i]-pe2[i];
    view_polyexpr();
}
```

```

void polyexpr::view_polyexpr()
{
    cout<<endl<<"poly exp  : ";
    for (int i=order;i>=0;i--)
    {
        if(i>=2 && pe3[i] !=0)
        {
            if(pe3[i]==1)
                cout<<"x^"<<i<<" + ";
            else
                cout<<pe3[i]<<"x^"<<i<<" + ";
        }
        else
        {
            if(i==1 &&pe3[i] !=0)
            {
                if(pe3[i]==1)
                    cout<<"x + ";
                else
                    cout<<pe3[i]<<"x + ";
            }
            else
            {
                if(pe3[i] !=0)
                    cout<<pe3[i];
            }
        }
    }
}

void main()
{
    int ord,ch;
    clrscr();
    cout<<endl<<"Enter max order of poly expression : ";
    cin>>ord;
    polyexpr obj(ord);
    obj.read_polyexpr1();
    obj.read_polyexpr2();
    cout<<"1: poly add  2: poly sub  3: exit";
    while(ch!=3)
    {
        cout<<"\nEnter your choice";
        cin>>ch;
        switch(ch)
        {
            case 1:obj.add_polyexpr();
                    break;
            case 2:obj.sub_polyexpr();
                    break;
            case 3: exit(0);
        }
    }
    getch();
}

```

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class demo
{
    int a[10],i,j,n,f,temp,ele,demo,mid,low,high;
public:
    void get();
    void sort();
    void linear();
    void binary();
    void dis();
};

void demo::get()
{
    cout<<"\n Enter n:";
    cin>>n;
    cout<<"\nEnter array Elements:";
    for(i=1;i<=n;i++)
        cin>>a[i];
}

void demo::linear()
{
    int ele;
    cout<<"\nEnter the element to be search";
    cin>>ele;
    for(i=1;i<=n;i++)
    {
        if(a[i]==ele)
        {
            cout<<"\nSuccessful search";
            cout<<"\nElement is found at position "<<i;
            return;
        }
    }
    if(i>n)
    {
        cout<<"\nUnsuccessful search:";
        cout<<"\nElement is not found ";
    }
}

void demo::sort()
{
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n-1;j++)
        {
            if(a[j]<a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
}
```

```

        }
    }
}

void demo::binary()
{
    cout<<"\nEnter element to be search ";
    cin>>ele;
    f=0;
    low=1;
    high=n;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(a[mid]==ele)
        {
            f=1;
            cout<<"\nElement is found at :"<<mid;
            return;
        }
        else if(a[mid]<ele)
            low=mid+1;
        else if(a[mid]>ele)
            high=mid-1;
    }
    if(f==0)
        cout<<"\n Element is not found:";
}

void demo::dis()
{
    cout<<"\n Element are \n";
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
    demo d;
    int ch;
    d.get();
    d.dis();
    cout<<"\n 1:Linear 2:Binary 3:exit\n";
    while(ch!=3)
    {
        cout<<"\nEnter Choice:";
        cin>>ch;
        switch(ch)
        {
            case 1: d.linear(); break;
            case 2: d.sort();
                    d.dis();
                    d.binary(); break;
            case 3: exit(0); break;
        }
    }
    getch();
}

```



```
*/ Output */
```

```
Enter n:3
```

```
Enter array Elements:12 3 45
```

```
Element are
```

```
12      3      45
```

```
1:Linear 2:Binary 3:exit
```

```
Enter Choice:1
```

```
Enter the element to be search 3
```

```
Successful search
```

```
Element is found at position 2
```

```
Enter Choice:2
```

```
Element are
```

```
45      12      3
```

```
Enter element to be search 12
```

```
Element is found at :2
```

```
Enter Choice:2
```

```
Element are
```

```
45      12      3
```

```
Enter element to be search 56
```

```
Element is not found:
```

```
Enter Choice:3
```

```
#include<iostream.h>
#include<conio.h>
class demo
{
    int a[10],temp;          //For string char a[10][10],temp[10];
    int,i,last,exch,j,n,temp;
public:
    void get();
    void asc_sort();
    void dec_sort();
    void disp();
};

void demo::get()
{
    cout<<"\n Enter the array size:";
    cin>>n;
    cout<<"\nEnter the array element:";
    for(i=1;i<=n;i++)
        cin>>a[i];
}

void demo::asc_sort()
{
    last=n;
    for(i=1;i<=n-1;i++)
    {
        exch=0;
        for(j=1;j<=last-1;j++)          // for string
        {
            if(a[j]>a[j+1])              // if(strcmp(a[j],a[j+1])>0)
            {
                temp=a[j];  // strcpy(temp,a[j]);
                a[j]=a[j+1]; // strcpy(a[j],a[j+1]);
                a[j+1]=temp; // strcpy(a[j+1],temp);
            }
            exch=exch+1;
        }
    }

    if(exch==0)
        return;
    else
        last=last-1;
}

void demo::dec_sort()
{
    last=n;
    for(i=1;i<=n-1;i++)
    {
        exch=0;
        for(j=1;j<=last-1;j++)          //for string
        {
            if(a[j]<a[j+1])              // if(strcmp(a[j],a[j+1])<0)
```

```

        {
            temp=a[j];    // strcpy(temp,a[j]);
            a[j]=a[j+1];  // strcpy(a[j],a[j+1]);
            a[j+1]=temp;  // strcpy(a[j+1],temp);
        }
        exch=exch+1;
    }
}

if(exch==0)
return;
else
last=last-1;
}

void demo::disp()
{
    cout<<"\nThe array element are";
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
    demo d;
    d.get();
    d.disp();
    d.asc_sort();
    cout<<"\nAfter Ascending Sort:";
    d.disp();
    d.dec_sort();
    cout<<"\nAfter Descending Sort:";
    d.disp();
    getch();
}

*/ Output */

```

Enter the array size: 3

Enter the array element: 12 3 45

The array element are12 3 45

After Ascending Sort:

The array element are3 12 45

After Descending Sort:

The array element are45 12 3

```
#include<iostream.h>
#include<conio.h>

class demo
{
    int a[10],temp;        // int a[10][10],temp[10] for string
    int i, min_index,j,n;
public:
    void get();
    void asc_sort();
    void dsc_sort();
    void disp();
};

void demo::get()
{
    cout<<"\nEnter the array size:";
    cin>>n;
    cout<<"\nEnter the array element:";
    for(i=1;i<=n;i++)
        cin>>a[i];
}

void demo::asc_sort()
{
    for(i=1;i<=n-1;i++)
    {
        min_index=i;
        for(j=i+1;j<=n;j++)            // for string
        {
            if(a[j]<a[min_index])    // if(strcmp(a[j],a[min_index])<0)
                min_index=j;
        }

        if(min_index!=i)
        {
            temp=a[min_index];        // strcpy(temp,a[min_index]);
            a[min_index]=a[i];        // strcpy(a[min_index],a[i]);
            a[i]=temp;                // strcpy(a[i],temp);
        }
    }
}

void demo::dsc_sort()
{
    for(i=1;i<=n;i++)
    {
        min_index=i;
        for(j=i+1;j<=n;j++)            // for string
        {
            if(a[j]>a[min_index])    // if(strcmp(a[j],a[min_index])>0)
                min_index=j;
        }

        if(min_index!=i)
        {

```

```

        temp=a[min_index];    // strcpy(temp,a[min_index]);
        a[min_index]=a[i];    // strcpy(a[min_index],a[i]);
        a[i]=temp;            // strcpy(a[i],temp);
    }
}

void demo::disp()
{
    cout<<"\n The array element are";
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
    demo d;
    d.get();
    d.disp();
    d.asc_sort();
    cout<<"\nAfter ascending sort:";
    d.disp();
    d.dsc_sort();
    cout<<"\n After Descending sort:";
    d.disp();
    getch();
}

*/ Output */

```

Enter the array size:4

Enter the array element:12 3 -45 -6

The array element are	12	3	-45	-6
After ascending sort:				
The array element are	-45	-6	3	12
After Descending sort:				
The array element are	12	3	-6	-45

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
class insert
{
    int n,a[10],temp,ptr,q,i,j,k,key;
public:
    void get();
    void sort();
    void display();
};

void insert::get()
{
    cout<<"\nEnter Range:";
    cin>>n;
    for(i=1;i<=n;i++)
        a[i]=random(1000);
    cout<<"\nElements are :";
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}

void insert::sort()
{
    a[0]=-9999;
    for(i=2;i<=n;i++)
    {
        temp=a[i];
        ptr=i-1;
        while(temp<a[ptr])
        {
            a[ptr+1]=a[ptr];
            ptr--;
        }
        a[ptr+1]=temp;
    }
}

void insert::display()
{
    cout<<"\nSorted Element using Insertion Sort:";
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
    insert h;
    h.get();
    h.sort();
    h.display();
    getch();
}

*/ Output */
Enter Range:5
Elements are :10      3      335      33      355
Sorted Element using Insertion Sort:3      10      33      335      355
```

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>

class demo
{
    int b[20][20],i,j,k,l,z,c,n,a[20];

public:
    void get();
    void sort();
    void disp();
};

void demo::get()
{
    cout<<"\nEnter the array size ";
    cin>>n;
    for(i=0;i<=9;i++)
    for(j=0;j<=9;j++)
    b[i][j]=-1;

    cout<<"\nEnter the array element:";
    for(i=0;i<n;i++)
        a[i]=random(1000);

    cout<<"\nThe array element are:";
    for(i=0;i<n;i++)
        cout<<a[i]<<"\t";
    l=0;
    for(i=0;i<n;i++)
    {
        k=0;
        c=a[i];
        while(c>0)
        {
            k++;
            c=c/10;
        }
        if(l<k)
            l=k;
    }
}

void demo::sort()
{
    for(j=1;j<=l;j++)
    {
        cout<<"\n";
        for(i=0;i<n;i++)
        {
            z=0;
            k=int(a[i]/pow(10,j-1))%10;
            cout<<k<<"\t";
            while(b[k][z]!=-1)
```

```

        z++;
        b[k][z]=a[i];
    }
    i=0;
    for(k=9;k>=0;k--)
    {
        z=0;
        while(b[k][z]!=-1)
        {
            a[i]=b[k][z];
            b[k][z]=-1;
            i++;
            z++;
        }
    }
}

void demo::disp()
{
    cout<<"\n The array element are ";
    for(i=0;i<n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
    demo d;
    d.get();
    cout<<"\nAfter ascending sort";
    d.sort();
    d.disp();
    getch();
}

*/ Output */

Enter the array size 5

Enter the array element:
The array element are:10      3      335      33      355
After ascending sort
0      3      5      3      5
3      5      0      3      1
3      3      0      0      0
The array element are 355      335      33      10      3

```



```
#include<iostream.h>
#include<conio.h>
#include<string.h>

class demo
{
    int x[20],temp;
    int a,n,i,j,left,right;
public:

    void get();
    void asort(int,int);
    int partition(int,int);
    void disp();
};

void demo::get()
{
    cout<<"\nEnter the array size:";
    cin>>n;
    cout<<"\nEnter the array element:";
    for(i=1;i<=n;i++)
        cin>>x[i];
    asort(1,n);
}

void demo::asort(int p,int q)
{
    if(p<q)
    {
        j=partition(p,q);
        asort(p,j-1);
        asort(j+1,q);
    }
}

int demo::partition(int lb, int ub)
{
    a=x[lb];
    left=lb+1;
    right=ub;
    do
    {
        while(x[left]<a)
            left++;
        while(x[right]>a)
            right--;
        if(left<right)
        {
            temp=x[left];
            x[left]=x[right];
            x[right]=temp;
        }
    }while(left<=right);

    x[lb]=x[right];
}
```

```

        x[right]=a;
        return(right);
    }

void demo::disp()
{
    cout<<"\nThe array element are:";
    for(i=1;i<=n;i++)
        cout<<x[i]<<"\t";
}

void main()
{
    clrscr();
    demo d;
    d.get();
    cout<<"\nAfter Ascending sort";           // Descending
    d.disp();
    getch();
}

*/ Output */

```

Enter the array size: 5

Enter the array element:12 3 -45 -67 8

After Ascending sort

The array element are:-67 -45 3 8 12

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>

int n; //remember that n should be declare global

class merge
{
    int a[10],b[10],i,j;
public:

    void read();
    void merge_sort(int l,int h);
    void merge1(int l,int m, int h);
    void disp();
};

void merge::read()
{
    for(i=0;i<n;i++)
        cin>>a[i];
}

void merge::merge_sort(int l,int h)
{
    int mid;
    if(l<h)
    {
        mid=int((l+h)/2);
        merge_sort(l,mid);
        merge_sort(mid+1,h);
        merge1(l,mid,h);
    }
}

void merge::merge1(int low,int m,int high)
{
    int h=low;
    int i=low;
    j=m+1;

    while((h<=m) && (j<=high))
    {
        if(a[h]<=a[j]) //Change descending order if(a[h]>=a[j])
        {
            b[i]=a[h];
            i++;
            h++;
        }
        else
        {
            b[i]=a[j];
            i++;
            j++;
        }
    }
}
```

```

    }

    if(h<=m)
    {
        while(h<=m)
        {
            b[i]=a[h];
            i++;
            h++;
        }
    }
    else
    {
        while(j<=h)
        {
            b[i]=a[j];
            i++;
            j++;
        }
    }

    for(int k=low;k<=high;k++)
        a[k]=b[k];
}

void merge::disp()
{
    for(i=0;i<n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
    int l,h;
    merge m;
    cout<<"\nEnter Elements";
    cin>>n;
    h=n-1;
    l=0;

    m.read();
    cout<<"\n\nDisplay the array elements\n";
    m.disp();
    m.merge_sort(l,h);
    cout<<"\nAfter Sorting\n";
    m.disp();
    getch();
}

```

*/ Output */

Enter Elements5
12 -34 5 67 -8

Display the array elements
12 -34 5 67 -8
After Sorting
-34 -8 5 12 67

```
#include<iostream.h>
#include<conio.h>
class heap
{
    int n,a[10],q,i,j,k,key;
public:
    void get();
    void create();
    void display();
};
void heap::get()
{
    cout<<"\nEnter Range:";
    cin>>n;
    cout<<"\nEnter the element:";
    for(i=1;i<=n;i++)
        cin>>a[i];
}
void heap::create()
{
    for(q=2;q<=n;q++)
    {
        i=q;
        key=a[q];
        j=i/2;
        while(i>1 && key>a[j]) //change Min heap while(i>1 && key<a[j])
        {
            a[i]=a[j];
            i=j;
            j=i/2;

            if(j<1)
                j=1;
        }
        a[i]=key;
    }
}
void heap::display()
{
    cout<<"\nHeap Tree:";
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}
void main()
{
    clrscr();
    heap h;
    h.get();
    h.create();
    h.display();
    getch();
}
*/ Output */
```

Enter Range:7

Enter the element:80 45 70 40 35 50 90

Heap Tree:90 45 80 40 35 50 70

```
#include<iostream.h>
#include<conio.h>

class heap
{
    int n,a[10],q,i,j,k,key,temp;
public:
    void get();
    void create();
    void sort();
    void display();
};

void heap::get()
{
    cout<<"\nEnter range:";
    cin>>n;
    cout<<"\nEnter the elements\n";
    for(i=1;i<=n;i++)
        cin>>a[i];
}

void heap::create()
{
    for(q=2;q<=n;q++)
    {
        i=q;
        key=a[q];
        j=i/2;
        while(i>1 && key >a[j]) //Change descending order key<a[j]
        {
            a[i]=a[j];
            i=j;
            j=i/2;
            if(j<1)
                j=1;
        }
        a[i]=key;
    }
}

void heap::sort()
{
    create();
    cout<<"\nMax Heap Tree";
    display();
    for(q=n;q>=2;q--)
    {
        temp=a[1];
        a[1]=a[q];
        a[q]=temp;
        i=1;
        key=a[1];
        j=2;
        if(j+1<q)
            if(a[j+1]>a[j])
                j++;
    }
}
```

```

        while(j<=q-1 && a[j]>key)
        {
            a[i]=a[j];
            i=j;
            j=i*2;
            if(j+1<q)
                if(a[j+1]>a[j])
                    j++;
            else
                if(j>n)
                    j=n;
            a[i]=key;
        }
    }
}

void heap::display()
{
    for(i=1;i<=n;i++)
        cout<<a[i]<<"\t";
}

void main()
{
    clrscr();
    heap h;
    h.get();
    h.sort();
    cout<<"\nSorted element are:";
    h.display();
    getch();
}

*/ Output */

Enter range: 5

Enter the elements
12 3 45 6 18

Max Heap Tree45 18      12      3      6
Sorted element are:3      6      12      18      45

```

```
#include<iostream.h>
#include<process.h>
#include<conio.h>

struct ver
{
    int data;
    ver *left,*right;
};

class tree
{
public:
    ver* create(int,ver*);
    void in(ver*);
    void post(ver*);
    void pre(ver*);
};

ver *tree::create(int c, ver *node)
{
    if (node==NULL)
    {
        node=new ver;
        node->data=c;
        node->left=NULL;
        node->right=NULL;
        return node;
    }
    else
    {
        if (c<node->data)
            node->left=create(c,node->left);
        else
            node->right=create(c,node->right);
        return node;
    }
}

void tree::in(ver * node)
{
    if (node)
    {
        in(node->left);
        cout<<node->data<<"\t";
        in(node->right);
    }
}

void tree::pre(ver * node)
{
    if (node)
    {
        cout<<node->data<<"\t";
        pre(node->left);
        pre(node->right);
    }
}
```



```

void tree::post(ver * node)
{
    if(node)
    {
        post(node->left);
        post(node->right);
        cout<<node->data<<"\t";
    }
}

void main()
{
    clrscr();
    tree t;
    ver *r=new ver;
    r=NULL;
    int n,ch;
    cout<<"\n 1:insert 2:inorder 3:preorder 4:postorder 5:exit :";
    while(ch!=5)
    {
        cout<<"\nEnter Choice:";
        cin>>ch;
        switch(ch)
        {
            case 1: cout<<"\nEnter Node:";
                     cin>>n;
                     r=t.create(n,r);
                     break;
            case 2: cout<<"\nInorder Traversal:";
                     t.in(r);
                     break;
            case 3: cout<<"\nPreorder Traversal:";
                     t.pre(r);
                     break;
            case 4: cout<<"\nPostorder Traversal:";
                     t.post(r);
                     break;
            case 5: exit(0);
        }
    }
    getch();
}

*/ Output */

```

```

1:insert 2:inorder 3:preorder 4:postorder 5:exit :
Enter Choice:1

Enter Node:18

Enter Choice:1

Enter Node:5

Enter Choice:1

Enter Node:20

Enter Choice:1

```

Enter Node:16

Enter Choice:1

Enter Node:30

Enter Choice:2

Inorder Traversal:5	16	18	20	30
---------------------	----	----	----	----

Enter Choice:3

Preorder Traversal:18	5	16	20	30
-----------------------	---	----	----	----

Enter Choice:4

Postorder Traversal:16	5	30	20	18
------------------------	---	----	----	----

Enter Choice:5

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>f
class NODE
{
    public:
        int data;
        NODE *left, *right;
};
class TREE
{
    // data
    public:
        NODE *root;
        // operations
        TREE();
        void ADD(int); // 1
        void DEL(int); // 2
        void PRE_ORD(NODE *); // 3
        void IN_ORD(NODE *); // 4
        void POST_ORD(NODE *); // 5
        NODE * FIND_IIO_SUCCESOR(NODE *);
        void MENU();
};
////////////////////////////////////
TREE::TREE()
{
    // def of function
    root = NULL;
}
////////////////////////////////////
void TREE::ADD(int ele)
{
    // def of function
    //----- (A) create new node -----
    NODE *NN;
    NN= new NODE(); // allocate new node
    //----- (B) fill up new node -----
    NN->data = ele;
    NN->left = NULL;
    NN->right = NULL;
    //---- (C) set the links -----
    if(root==NULL) //case - II Not Full - Empty
        root=NN;
    else //case - III Not Full - Not Empty
    {
        NODE *par = NULL;
        NODE *ptr = root;
        while(ptr != NULL)
        {
            par = ptr;
            if(ele < ptr->data)
                ptr = ptr->left;
            else
                ptr = ptr->right;
        }
    }
}
```

```

        if(ele < par->data)
            par->left = NN;
        else
            par->right = NN;
    }
}
////////////////////////////////////
NODE * TREE:: FIND IIO SUCCESSOR(NODE *ptr)
{
    NODE *par_ssr = ptr;
    NODE *ssr = ptr->right;
    while(ssr->left != NULL)
    {
        par_ssr = ssr;
        ssr = ssr->left;
    }
    // delete ssr
    if(par_ssr == ptr)
        par_ssr->right = ssr->right;
    else
        par_ssr->left = ssr->right;
    return ssr;
}
////////////////////////////////////
void TREE::DEL(int ele)
{
    if(root == NULL)
        cout<<endl<<"Tre is empty";
    else // Tree not empty
    {
        NODE *ptr=root;
        NODE *par=NULL;
        // find the node to be deleted with his parent
        while(ptr!=NULL)
        {
            if(ptr->data==ele)
                break; // node found
            else
            {
                par = ptr;
                if(ele<ptr->data)
                    ptr=ptr->left;
                else
                    ptr=ptr->right;
            }
        }
        if(ptr == NULL) // node not found
            cout<<"Element Not Found";
        else // node found
        {
            NODE *TEMP=ptr;
            if(ptr->left==NULL && ptr->right==NULL) // zero child
            {
                if(par == NULL) // ptr is root of tree
                    root = NULL;
                else
                {
                    if(ele<par->data)
                        par->left=NULL;
                    else
                        par->right=NULL;
                }
            }
            else

```

```

        {
            if(ptr->left == NULL || ptr->right == NULL) // 1 child
            {
                // find out child
                NODE *ch;
                if(ptr->left==NULL)
                    ch = ptr->right;
                else
                    ch=ptr->left;
                // set links
                if(par == NULL) // ptr is root of tree
                    root = ch;
                else
                {
                    if(ele<par->data)
                        par->left=ch;
                    else
                        par->right=ch;
                }
            }
            else // 2 children
            {
                NODE *IIOS = FIND_IIO_SUCCESSOR(ptr);

                IIOS->left = ptr->left;
                IIOS->right = ptr->right;

                if( ele < par->data )
                    par->left = IIOS;
                else
                    par->right = IIOS;
            }
        }
        delete TEMP;
    }
}

////////////////////////////////////
void TREE::PRE_ORD( NODE *ptr)
{
    // def of function
    if(ptr != NULL)
    {
        cout<<ptr->data<<" ";
        PRE_ORD(ptr->left);
        PRE_ORD(ptr->right);
    }
}

////////////////////////////////////
void TREE::IN_ORD( NODE *ptr)
{
    // def of function
    if(ptr != NULL)
    {
        IN_ORD(ptr->left);
        cout<<ptr->data<<" ";
        IN_ORD(ptr->right);
    }
}

////////////////////////////////////
void TREE::POST_ORD( NODE *ptr)
{

```

```

// def of function
if(ptr != NULL)
{
    POST_ORD(ptr->left);
    POST_ORD(ptr->right);
    cout<<ptr->data<<" ";
}
}
////////////////////////////////////
void TREE::MENU()
{
    int ele, opt;
    do
    {
        cout<<endl<<"=====\n";
        cout<<endl<<"1 Add Node";
        cout<<endl<<"2 Delete Node";
        cout<<endl<<"3 Pre-Order Traversal";
        cout<<endl<<"4 In-Order Traversal";
        cout<<endl<<"5 Post-Order Traversal";
        cout<<endl<<"6 Exit";
        cout<<endl<<"=====\n";
        cout<<endl<<"Enter your choice : ";
        cin>>opt;
        switch(opt)
        {
            case 1:
                cout<<endl<<"Enter element : ";
                cin>>ele;
                ADD(ele);
                IN_ORD(root);
                break;
            case 2:
                cout<<endl<<"Enter element : ";
                cin>>ele;
                DEL(ele);
                if(root != NULL)
                    IN_ORD(root);
                else
                    cout<<endl<<"Tree empty";
                break;
            case 3:
                if(root != NULL)
                    PRE_ORD(root);
                else
                    cout<<endl<<"Tree empty";
                break;
            case 4:
                if(root != NULL)
                    IN_ORD(root);
                else
                    cout<<endl<<"Tree empty";
                break;
            case 5:
                if(root != NULL)
                    POST_ORD(root);
                else
                    cout<<endl<<"Tree empty";
                break;
            case 6:
                exit(0);
            default:

```

```

        cout<<endl<<"Invalid input";
    }
}while(1);
}
////////////////////////////////////
void main()
{
    TREE obj;
    clrscr();
    obj.MENU();
    getch();
}
////////////////////////////////////

```

```
#include<iostream.h>
#include<conio.h>

class dfstree
{
    int a[20][20], visited[20],n,i,j;
public:
    void dfs(int);
    void get();
};

void dfstree::get()
{
    cout<<"\nEnter the number of node";
    cin>>n;
    for(i=0;i<n;i++)
        visited[i]=0;
    cout<<"\nEnter the adjacency matrix:";
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            cin>>a[i][j];
        }
    }
    dfs(0);
}

void dfstree::dfs(int v)
{
    int k;
    visited[v]=1;
    cout<<"\t"<<v+1;
    for(k=1;k<n;k++)
        if(a[v][k]==1)
            if(visited[k]==0)
                dfs(k);
}

void main()
{
    clrscr();
    dfstree d;
    d.get();
    getch();
}

*/ Output */
```

```
Enter the number of node5
Enter the adjacency matrix:
0 1 1 0 0
1 0 0 1 1
1 0 0 1 0
0 1 1 0 1
0 1 0 1 0
```

```
1         2         4         3         5
```



```
#include<iostream.h>
```

```
#include<conio.h>
```

```
class bfstree
```

```
{
    int reach[20],a[20][20],q[20],n,i,j,f,r,index;
public:
    bfstree()
    {
        f=r=0;
        index=1;
    }
    void get();
    void bfs();
};
```

```
void bfstree::get()
```

```
{
    cout<<"\nEnter number of vertices:";
    cin>>n;
    cout<<"\nEnter Adjacency matrix:";
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            reach[i]=0;
            cin>>a[i][j];
        }
}
```

```
void bfstree::bfs()
```

```
{
    reach[1]=1;
    f++;
    r++;
    q[r]=index;
    cout<<"\nBFS is ";
    while(f<=r)
    {
        index=q[f];
        f++;
        cout<<index<<"\t";
        for(j=1;j<=n;j++)
        {
            if(a[index][j]==1 && reach[j]!=1)
            {
                reach[j]=1;
                r++;
                q[r]=j;
            }
        }
    }
}
```

```
void main()
```

```
{
    clrscr();
    bfstree b;
```

```
        b.get();
        b.dbfs();
        getch();
    }

    /* Output */

    Enter number of vertices:6

    Enter Adjacency matrix:
    0 1 1 0 0 0
    1 0 0 1 0 0
    1 0 0 0 0 1
    0 1 0 0 1 1
    0 0 0 1 0 0
    0 0 1 1 0 0

    BFS is 1          2          3          4          6          5
```

```

-----
Assignment Name: Implement All Pair Shortest Path (Floyd-Warshall)
Class: MCA I                                     Lab: MCA 414 LAB ON
DSA-----
#include<iostream.h>
#include<conio.h>

class path
{
    int a[5][5],i,j,k,n,s,d;
public:
    void insert();
    void display();
};

void path::insert()
{
    cout<<"\nEnter the no. of vertices";
    cin>>n;
    cout<<"\nEnter the matrix:";
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            cin>>a[i][j];
            if(a[i][j]==-1)
                a[i][j]=9999;
        }

    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            for(k=1;k<=n;k++)
                if(a[i][j]>(a[i][k]+a[k][j]))
                    a[i][j]=a[i][k]+a[k][j];
                else
                    a[i][j]=(a[i][k]+a[k][j]);
}

void path::display()
{
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            cout<<"\t"<<a[i][j];
        cout<<"\n";
    }
    cout<<"\nEnter the source vertex:";
    cin>>s;
    cout<<"\nEnter the destination vertex:";
    cin>>d;
    cout<<"\nPath from Source "<<s<<" to destination "<<d<<" is ";
    cout<<a[s][d];
}

void main()
{
    clrscr();
    path p;
    p.insert();
    cout<<"\n Shortest path is \n";
    p.display();
    getch();
}

```

```
}
```

```
*/ Output */
```

```
Enter the no. of vertices 3
```

```
Enter the matrix:0 4 11
                  6 0 2
                  3 -1 0
```

```
Shortest path is
```

0	4	6
5	0	2
3	7	0

```
Enter the source vertex:3
```

```
Enter the destination vertex:2
```

```
Path from Source 3 to destination 2 is 7
```

Assignment Name: Minimum Cost Spanning tree using Prims Algorithm
Class: MCA ILab: MCA 414 LAB ON DSA

```
#include<iostream.h>
#include<conio.h>
int n;
class single
{
    int
v,cost[10][10],i,j,s[10],e[10],near1[10],t[10][3],m,minedge,k,l,mincost;
    int jindex;
    float dist[10];
public:
    void get();
    void prim();
    void display();
};

void single::get()
{
    m=1;
    minedge=9999;
    cout<<"\nEnter the no. of vertices\n";
    cin>>n;
    cout<<"\nEnter the Adjacency matrix\n";
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            cin>>cost[i][j];
            if(cost[i][j]==-1)
                cost[i][j]=9999;
            else
            {
                e[m]=cost[i][j];
                if(e[m]<minedge)
                {
                    minedge=e[m];
                    k=i;
                    l=j;
                }
            }
        }
}

void single::prim()
{
    t[1][1]=k;
    t[1][2]=l;
    mincost=cost[k][l];
    for(i=1;i<=n;i++)
    {
        if(cost[i][l]<cost[i][k])
            near1[i]=l;
        else
            near1[i]=k;
    }

    near1[k]=near1[l]=0;
    int minj=9999;
    for(i=2;i<=n-1;i++)
```

```

{
    minj=9999;
    for(j=1;j<=n;j++)
    {
        if(near1[j]!=0)
        {
            if(cost[j][near1[j]]<minj)
            {
                minj=cost[j][near1[j]];
                jindex=j;
            }
        }
    }
    t[i][1]=jindex;
    t[i][2]=near1[jindex];
    mincost=mincost+cost[jindex][near1[jindex]];
    near1[jindex]=0;

    for(int k1=1;k1<=n;k1++)
    {
        if(near1[k1]!=0 && cost[k1][near1[k1]]>cost[k1][jindex])
            near1[k1]=jindex;
    }
}
cout<<"\n Mincost ="<<mincost;
}

```

```

void single::display()
{
    cout<<endl;
    cout<<"\nMinimum Spanning Tree Path as follow\n";
    cout<<t[1][1]<<"->"<<t[1][2];
    for(i=2;i<n;i++)
    {
        cout<<"->";
        cout<<t[i][1];
    }
}

```

```

void main()

```

```

{
    single d;
    clrscr();
    d.get();
    d.prim();
    d.display();
    getch();
}

```

```

*/ Output */

```

```

Enter the no. of vertices

```

```

7

```

```

Enter the Adjacency matrix

```

```

-1 28 -1 -1 -1 10 -1

```

```

28 -1 16 -1 -1 -1 14

```

```

-1 16 -1 12 -1 -1 -1

```

```

-1 -1 12 -1 22 -1 18

```

```

-1 -1 -1 22 -1 25 24

```

```

10 -1 -1 -1 25 -1 -1

```

```

-1 14 -1 18 24 -1 -1

```

```

Mincost =99

```

```

Minimum Spanning Tree Path as follow

```

```

1->6->5->4->3->2->7

```