```
//Complex:
using namespace std;
#include<iostream>
struct Complex
{
	int real,imag;
	Complex()
	{
		//cout<<"\nDefault constructor called\n";
		this->real=0;
		this->imag=0;
	}
	Complex(int real,int imag)
	{
		//cout<<"\nParameterised constructor called\n";
		this->real=real;
		this->imag=imag;
	}
	void setReal(int real)
	{
		this->real=real;
	}
	void setImag(int imag)
	{
		this->imag=imag;
```

```
}
int getReal()
{
	return this->real;
}
int getImag()
{
	return this->imag;
}
void display()
{
	cout<<"\ncomplex number: "<<this->real<<"+"<<this->imag<<"i"<<"\n";
}
Complex operator+(Complex c2)
{
	Complex temp;
	temp.real=c2.real+this->real;
	temp.imag=c2.imag+this->imag;
	return temp;
}
Complex operator+(int t)
{
	Complex temp;
	temp.real=this->real+t;
	temp.imag=this->imag+t;
```

```
        return temp;
}
Complex operator-(Complex c2)
{
        Complex temp;
        temp.real=c2.real-this->real;
        temp.imag=c2.imag-this->imag;
        return temp;
}
Complex operator-(int t)
{
        Complex temp;
        temp.real=this->real-t;
        temp.imag=this->imag-t;
        return temp;
}
Complex operator*(Complex c2)
{
        Complex temp;
        temp.real=c2.real*this->real;
        temp.imag=c2.imag*this->imag;
        return temp;
}
Complex operator*(int t)
{
```

```
        Complex temp;
        temp.real=this->real*t;
        temp.imag=this->imag*t;
        return temp;
}
Complex operator/(Complex c2)
{
        Complex temp;
        temp.real=c2.real/this->real;
        temp.imag=c2.imag/this->imag;
        return temp;
}
Complex operator/(int t)
{
        Complex temp;
        temp.real=this->real/t;
        temp.imag=this->imag/t;
        return temp;
}
int operator>(Complex c2)
{
        if(this->real>c2.real)
        return 1;
        else
        return 0;
```

```
}
int operator>(int t)
{
        if(this->real>t)
        return 1;
        else
        return 0;
}
int operator<(Complex c2)
{
        if(this->real<c2.real)
        return 1;
        else
        return 0;
}
int operator<(int t)
{
        if(this->real<t)
        return 1;
        else
        return 0;
}
int operator!=(Complex c2)
{
        if(this->real!=c2.real)
```

```
        return 1;
        else
        return 0;
}
int operator!=(int t)
{
        if(this->real!=t)
        return 1;
        else
        return 0;
}
int operator&&(Complex c2)
{
        if(this->real&&c2.real)
                return 1;
        else
        return 0;
}
int operator&&(int t)
{
        if(this->real&&t)
                return 1;
        else
        return 0;
}
```

```
        int operator||(Complex c2)
        {
                if(this->real||c2.real)
                        return 1;
                else
                return 0;
        }
        int operator||(int t)
        {
                if(this->real||t)
                        return 1;
                else
                return 0;
        }
};
Complex operator+(int,Complex);
Complex operator-(int,Complex);
Complex operator*(int,Complex);
Complex operator/(int,Complex);
int operator>(int,Complex);
int operator<(int,Complex);
int operator!=(int,Complex);
int operator&&(int,Complex);
int operator||(int,Complex);
int main()
```

```
{
        Complex c1,c3,c4,c5;
        int real,imag;
        cout<<"\nEnter real part of complex number:\n";
        cin>>real;
        cout<<"\nEnter imaginary part of complex number:\n";
        cin>>imag;
        c1.setReal(real);
        c1.setImag(imag);
        c1.display();
        Complex c2(10,5);
        c2.display();
        c3=c1+c2;
        cout<<"\nAddition of 2 complex number using member function:\n";
        c3.display();
        c4=c1+10;
        cout<<"\nAdd 10 to real and imaginary part of complex number using meber function:\n";
        c4.display();
        c5=5-c1;
        cout<<"\nAdd 5 to real and imaginary part of complex number using non member function:\n";
        c5.display();
        c3=c1-c2;
        cout<<"\nSubstraction of 2 complex number using member function:\n";
        c3.display();
        c4=c1-10;
```

```
        cout<<"\nSubtract 10 from real and imaginary part of complex number using meber function:\n";

        c4.display();

        c5=5-c1;

        cout<<"\nSubtract real and imaginary part of complex number from 5 using non member function:\n";

        c5.display();

        c3=c1*c2;

        cout<<"\nMultiplication of 2 complex number using member function:\n";

        c3.display();

        c4=c1*10;

        cout<<"\nMultiply by 10 to real and imaginary part of complex number using meber function:\n";

        c4.display();

        c5=5*c1;

        cout<<"\nMultiply by 5 to real and imaginary part of complex number using non member function:\n";

        c5.display();

        c3=c1/c2;

        cout<<"\nDivision of 2 complex number using member function:\n";

        c3.display();

        c4=c1/10;

        cout<<"\nDivide by 10 to real and imaginary part of complex number using meber function:\n";

        c4.display();

        c5=5/c1;

        cout<<"\nDivide by real and imaginary part of complex number to 5 using non member function:\n";
```

```
c5.display();
if(c1>c2)
{
        cout<<"\n";
        c1.display();
        cout<<" is greater than ";
        c2.display();
        cout<<"\n";
}
else
{

        cout<<"\n";
        c1.display();
        cout<<" is less than ";
        c2.display();
        cout<<"\n";
}
if(c1>10)
{
        cout<<"\n";
        c1.display();
        cout<<" is greater than 10\n";
}
else
```

```
{
        cout<<"\n";
        c1.display();
        cout<<" is less than 10\n";
}
if(10>c1)
{
        cout<<"\n10 is greater than ";
        c1.display();
        cout<<"\n";
}
else
{
        cout<<"\n10 is less than ";
        c1.display();
        cout<<"\n";
}
if(c1<c2)
{
        cout<<"\n";
        c1.display();
        cout<<" is less than ";
        c2.display();
        cout<<"\n";
}
```

```
else
{
        cout<<"\n";
        c1.display();
        cout<<" is greater than ";
        c2.display();
        cout<<"\n";
}
if(c1<10)
{
        cout<<"\n";
        c1.display();
        cout<<" is less than 10\n";
}
else
{
        cout<<"\n";
        c1.display();
        cout<<" is greater than 10\n";
}
if(10<c1)
{
        cout<<"\n10 is less than ";
        c1.display();
        cout<<"\n";
```

```
}
else
{
        cout<<"\n10 is greater than ";
        c1.display();
        cout<<"\n";
}
if(c1!=c2)
{
        cout<<"\n";
        c1.display();
        cout<<" is not equal ";
        c2.display();
        cout<<"\n";
}
else
{
        cout<<"\n";
        c1.display();
        cout<<" is equal to ";
        c2.display();
        cout<<"\n";
}
if(c1!=10)
{
```

```
        cout<<"\n";
        c1.display();
        cout<<" is not equal to 10\n";
}
else
{
        cout<<"\n";
        c1.display();
        cout<<" is equal to 10\n";
}
if(10!=c1)
{
        cout<<"\n10 is not equal to ";
        c1.display();
        cout<<"\n";
}
else
{
        cout<<"\n10 is equal to ";
        c1.display();
        cout<<"\n";
}
if(c1&&c2)
cout<<"\nBoth are non zero value\n";
else
```

```
        cout<<"\nOne of the value is zero\n";
        if(c1&&0)
        cout<<"\nBoth are non zero value\n";
        else
        cout<<"\nOne of the value is zero\n";
        if(10&&c1)
        cout<<"\nBoth are non zero value\n";
        else
        cout<<"\nOne of the value is zero\n";
        if(c1||c2)
        cout<<"\nThere is non zero value\n";
        else
        cout<<"\nBoth the values are zero\n";
        if(c1||0)
        cout<<"\nThere is non zero value\n";
        else
        cout<<"\nBoth the value are zero\n";
        if(10||c1)
        cout<<"\nThere is non zero value\n";
        else
        cout<<"\nBoth the value are zero\n";
}
Complex operator+(int t,Complex c1)
{
        Complex temp;
```

```
        temp.setReal(t+c1.getReal());
        temp.setImag(t+c1.getImag());
        return temp;
}
Complex operator-(int t,Complex c1)
{
        Complex temp;
        temp.setReal(t-c1.getReal());
        temp.setImag(t-c1.getImag());
        return temp;
}
Complex operator*(int t,Complex c1)
{
        Complex temp;
        temp.setReal(c1.getReal()*t);
        temp.setImag(c1.getImag()*t);
        return temp;
}
Complex operator/(int t,Complex c1)
{
        Complex temp;
        temp.setReal(t/c1.getReal());
        temp.setImag(t/c1.getImag());
        return temp;
}
```

```
int operator>(int t,Complex c1)
{
	if(t>c1.getReal())
	return 1;
	else
	return 0;
}
int operator<(int t,Complex c1)
{
	if(t<c1.getReal())
	return 1;
	else
	return 0;
}
int operator!=(int t,Complex c1)
{
	if(t!=c1.getReal())
	return 1;
	else
	return 0;
}
int operator&&(int t,Complex c1)
{
	if(t&&c1.getReal())
		return 1;
```

```
	else
	return 0;
}
int operator||(int t,Complex c1)
{
	if(t||c1.getReal())
		return 1;
	else
	return 0;
}

//Distance:
#include<iostream>
using namespace std;
struct Distance
{
	int feet,inch;
	Distance()
	{
	//	cout<<"\nDefault constructor called\n";
		this->feet=0;
		this->inch=0;
	}
	Distance(int f,int i)
	{
```

```
//          cout<<"\nParameterised constructor called\n";
            this->feet=f;
            this->inch=i;
}
void setFeet(int f)
{
            this->feet=f;
}
void setInch(int i)
{
            this->inch=i;
}
int getFeet()
{
            return this->feet;
}
int getInch()
{
            return this->inch;
}
void display()
{
            cout<<"\nDistance:\t"<<this->feet<<" feet "<<this->inch<<" inches\n";
}
Distance operator+(Distance d2)
```

```
{
        Distance temp;
        temp.feet=this->feet+d2.feet;
        temp.inch=this->inch+d2.inch;
        return temp;
}
Distance operator+(int t)
{
        Distance temp;
        temp.feet=this->feet+t;
        temp.inch=this->inch+t;
        return temp;
}
Distance operator-(Distance d2)
{
        Distance temp;
        temp.feet=this->feet-d2.feet;
        temp.inch=this->inch-d2.inch;
        return temp;
}
Distance operator-(int t)
{
        Distance temp;
        temp.feet=this->feet-t;
        temp.inch=this->inch-t;
```

```
        return temp;
}
Distance operator*(Distance d2)
{
        Distance temp;
        temp.feet=this->feet*d2.feet;
        temp.inch=this->inch*d2.inch;
        return temp;
}
Distance operator*(int t)
{
        Distance temp;
        temp.feet=this->feet*t;
        temp.inch=this->inch*t;
        return temp;
}
Distance operator/(Distance d2)
{
        Distance temp;
        temp.feet=this->feet/d2.feet;
        temp.inch=this->inch/d2.inch;
        return temp;
}
Distance operator/(int t)
{
```

```
        Distance temp;
        temp.feet=this->feet/t;
        temp.inch=this->inch/t;
        return temp;
}
int operator>(Distance d2)
{
        if(this->feet>d2.feet)
        return 1;
        else
        return 0;
}
int operator>(int t)
{
        if(this->feet>t)
        return 1;
        else
        return 0;
}
int operator<(Distance d2)
{
        if(this->feet<d2.feet)
                return 1;
        else
                return 0;
```

```
        }
        int operator<(int t)
        {
                if(this->feet<t)
                        return 1;
                else
                        return 0;
        }
        int operator!=(Distance d2)
        {
                if(this->feet!=d2.feet)
                return 1;
                else
                return 0;
        }
        int operator!=(int t)
        {
                if(this->feet!=t)
                return 1;
                else
                return 0;
        }
};
Distance operator+(int,Distance);
Distance operator-(int,Distance);
```

```
Distance operator*(int,Distance);
Distance operator/(int,Distance);
int operator>(int,Distance);
int operator<(int,Distance);
int operator!=(int,Distance);
int main()
{
        int feet,inch;
        Distance d1,d3,d4,d5;
        cout<<"\nEnter distance in feet and inches\n";
        cin>>feet>>inch;
        d1.setFeet(feet);
        d1.setInch(inch);
        d1.display();
        Distance d2(5,2);
        d2.display();
        d3=d1+d2;
        cout<<"\nAddition of 2 distance using member function:\n";
        d3.display();
        d4=d1+10;
        cout<<"\nAdd 10 to feet and inches of distance using meber function:\n";
        d4.display();
        d5=5+d1;
        cout<<"\nAdd 5 to feet and inches of distance using non meber function:\n";
        d5.display();
```

```
d3=d1-d2;
cout<<"\nSubstraction of 2 distance using member function:\n";
d3.display();
d4=d1-10;
cout<<"\nSubtract 10 from feet and inches of distance using meber function:\n";
d4.display();
d5=10-d1;
cout<<"\nSubtract feet and inches of distance from 10 using non meber function:\n";
d5.display();
d3=d1*d2;
cout<<"\nMultiplication of 2 distance using member function:\n";
d3.display();
d4=d1*10;
cout<<"\nMultiply by 10 to feet and inches of distance using meber function:\n";
d4.display();
d5=5*d1;
cout<<"\nMultiply by 5 to feet and inches of distance using non meber function:\n";
d5.display();
d3=d1/d2;
cout<<"\nDivision of 2 distance using member function:\n";
d3.display();
d4=d1/10;
cout<<"\nDivide by 10 to feet and inches of distance using meber function:\n";
d4.display();
d5=15/d1;
```

```
cout<<"\nDivide by feet and inches of distance to 15 using non meber function:\n";
d5.display();
if(d1>d2)
{
        cout<<"\n";
        d1.display();
        cout<<" is greater than ";
        d2.display();
        cout<<"\n";
}
else
{

        cout<<"\n";
        d1.display();
        cout<<" is less than ";
        d2.display();
        cout<<"\n";
}
if(d1>10)
{
        cout<<"\n";
        d1.display();
        cout<<" is greater than 10\n";
}
```

```
else
{
        cout<<"\n";
        d1.display();
        cout<<" is less than 10\n";
}
if(10>d1)
{
        cout<<"\n10 is greater than ";
        d1.display();
        cout<<"\n";
}
else
{
        cout<<"\n10 is less than ";
        d1.display();
        cout<<"\n";
}
if(d1<d2)
{
        cout<<"\n";
        d1.display();
        cout<<" is less than ";
        d2.display();
        cout<<"\n";
```

```
}
else
{
        cout<<"\n";
        d1.display();
        cout<<" is greater than ";
        d2.display();
        cout<<"\n";
}
if(d1<10)
{
        cout<<"\n";
        d1.display();
        cout<<" is less than 10\n";
}
else
{
        cout<<"\n";
        d1.display();
        cout<<" is greater than 10\n";
}
if(10<d1)
{
        cout<<"\n10 is less than ";
        d1.display();
```

```
        cout<<"\n";
}
else
{
        cout<<"\n10 is greater than ";
        d1.display();
        cout<<"\n";
}
if(d1!=d2)
{
        cout<<"\n";
        d1.display();
        cout<<" is not equal to ";
        d2.display();
        cout<<"\n";
}
else
{
        cout<<"\n";
        d1.display();
        cout<<" is equal to ";
        d2.display();
        cout<<"\n";
}
if(d1!=10)
```

```
        {
                cout<<"\n";
                d1.display();
                cout<<" is not equal to 10\n";
        }
        else
        {
                cout<<"\n";
                d1.display();
                cout<<" is equal to 10\n";
        }
        if(10!=d1)
        {
                cout<<"\n10 is not equal to ";
                d1.display();
                cout<<"\n";
        }
        else
        {
                cout<<"\n10 is equal to ";
                d1.display();
                cout<<"\n";
        }
}
Distance operator+(int t,Distance d1)
```

```
{
        Distance temp;
        temp.setFeet(t+d1.getFeet());
        temp.setInch(t+d1.getInch());
        return temp;
}
Distance operator-(int t,Distance d1)
{
        Distance temp;
        temp.setFeet(t-d1.getFeet());
        temp.setInch(t-d1.getInch());
        return temp;
}
Distance operator*(int t,Distance d1)
{
        Distance temp;
        temp.setFeet(d1.getFeet()*t);
        temp.setInch(d1.getInch()*t);
        return temp;
}
Distance operator/(int t,Distance d1)
{
        Distance temp;
        temp.setFeet(t/d1.getFeet());
        temp.setInch(t/d1.getInch());
```

```
        return temp;
}
int operator>(int t,Distance d1)
{
        if(t>d1.getFeet())
        return 1;
        else
        return 0;
}
int operator<(int t,Distance d1)
{
        if(t<d1.getFeet())
        return 1;
        else
        return 0;
}
int operator!=(int t,Distance d1)
{
        if(t!=d1.getFeet())
        return 1;
        else
        return 0;
}
```