```
//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 add(Complex c2)

{

        Complex temp;

        temp.real=c2.real+this->real;

        temp.imag=c2.imag+this->imag;

        return temp;

}

Complex add(int t)

{

        Complex temp;

        temp.real=this->real+t;

        temp.imag=this->imag+t;
        return temp;
}
Complex sub(Complex c2)
{
        Complex temp;
        temp.real=c2.real-this->real;
        temp.imag=c2.imag-this->imag;
        return temp;
}
Complex sub(int t)
{
        Complex temp;
        temp.real=this->real-t;
        temp.imag=this->imag-t;
        return temp;
}
Complex mult(Complex c2)
{
        Complex temp;
        temp.real=c2.real*this->real;
        temp.imag=c2.imag*this->imag;
        return temp;
}
Complex mult(int t)
{
        Complex temp;

        temp.real=this->real*t;

        temp.imag=this->imag*t;

        return temp;

    }

    Complex div(Complex c2)

    {

        Complex temp;

        temp.real=c2.real/this->real;

        temp.imag=c2.imag/this->imag;

        return temp;

    }

    Complex div(int t)

    {

        Complex temp;

        temp.real=this->real/t;

        temp.imag=this->imag/t;

        return temp;

    }

};

Complex add(Complex,int);

Complex sub(Complex,int);

Complex mult(Complex,int);

Complex div(Complex,int);

int main()
```

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

	c3=c1.sub(c2);

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

	c3.display();

	c4=c1.sub(10);

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

	c4.display();

	c5=sub(c1,5);

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

	c5.display();

	c3=c1.mult(c2);

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

	c3.display();

	c4=c1.mult(10);

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

	c4.display();

	c5=mult(c1,5);

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

	c5.display();

	c3=c1.div(c2);

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

	c3.display();

	c4=c1.div(10);
	cout<<"\nDivide by 10 to real and imaginary part of complex number using meber function:\n";

	c4.display();

	c5=div(c1,5);

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

	c5.display();

}

Complex add(Complex c1,int t)

{

	Complex temp;

	temp.setReal(c1.getReal()+t);

	temp.setImag(c1.getImag()+t);

	return temp;

}

Complex sub(Complex c1,int t)

{

	Complex temp;

	temp.setReal(c1.getReal()-t);

	temp.setImag(c1.getImag()-t);

	return temp;

}

Complex mult(Complex c1,int t)

{

	Complex temp;

	temp.setReal(c1.getReal()*t);

	temp.setImag(c1.getImag()*t);
	return temp;
}
Complex div(Complex c1,int t)
{
	Complex temp;
	temp.setReal(c1.getReal()/t);
	temp.setImag(c1.getImag()/t);
	return temp;
}

//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 add(Distance d2)

{
        Distance temp;

        temp.feet=this->feet+d2.feet;

        temp.inch=this->inch+d2.inch;

        return temp;

}

Distance add(int t)

{

        Distance temp;

        temp.feet=this->feet+t;

        temp.inch=this->inch+t;

        return temp;

}

Distance sub(Distance d2)

{

        Distance temp;

        temp.feet=this->feet-d2.feet;

        temp.inch=this->inch-d2.inch;

        return temp;

}

Distance sub(int t)

{

        Distance temp;

        temp.feet=this->feet-t;

        temp.inch=this->inch-t;

        return temp;
```

```
}

Distance mult(Distance d2)

{

        Distance temp;

        temp.feet=this->feet*d2.feet;

        temp.inch=this->inch*d2.inch;

        return temp;

}

Distance mult(int t)

{

        Distance temp;

        temp.feet=this->feet*t;

        temp.inch=this->inch*t;

        return temp;

}

Distance div(Distance d2)

{

        Distance temp;

        temp.feet=this->feet/d2.feet;

        temp.inch=this->inch/d2.inch;

        return temp;

}

Distance div(int t)

{

        Distance temp;
```

```
                temp.feet=this->feet/t;
                temp.inch=this->inch/t;
                return temp;
        }
};
Distance add(Distance,int);
Distance sub(Distance,int);
Distance mult(Distance,int);
Distance div(Distance,int);
int main()
{
        int feet,inch;
        Distance d1,d2,d3,d4,d5;
        cout<<"\nEnter distance in feet and inches\n";
        cin>>feet>>inch;
        d1.setFeet(feet);
        d1.setInch(inch);
        d1.display();
        cout<<"\nEnter distance in feet and inches\n";
        cin>>feet>>inch;
        d2.setFeet(feet);
        d2.setInch(inch);
        d2.display();
        d3=d1.add(d2);
        cout<<"\nAddition of 2 distance using member function:\n";
```

```
d3.display();

d4=d1.add(10);

cout<<"\nAdd 10 to feet and inches of distance using meber function:\n";

d4.display();

d5=add(d1,5);

cout<<"\nAdd 5 to feet and inches of distance using non meber function:\n";

d5.display();

d3=d1.sub(d2);

cout<<"\nSubstraction of 2 distance using member function:\n";

d3.display();

d4=d1.sub(10);

cout<<"\nSubtract 10 from feet and inches of distance using meber function:\n";

d4.display();

d5=sub(d1,5);

cout<<"\nSubtract 5 from feet and inches of distance using non meber function:\n";

d5.display();

d3=d1.mult(d2);

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

d3.display();

d4=d1.mult(10);

cout<<"\nMultiply by 10 to feet and inches of distance using meber function:\n";

d4.display();

d5=mult(d1,5);

cout<<"\nMultiply by 5 to feet and inches of distance using non meber function:\n";

d5.display();
```

```
        d3=d1.div(d2);

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

        d3.display();

        d4=d1.div(10);

        cout<<"\nDivide by 10 to feet and inches of distance using meber function:\n";

        d4.display();

        d5=div(d1,5);

        cout<<"\nDivide by 5 to feet and inches of distance using non meber function:\n";

        d5.display();

}

Distance add(Distance d1,int t)

{

        Distance temp;

        temp.setFeet(d1.getFeet()+t);

        temp.setInch(d1.getInch()+t);

        return temp;

}

Distance sub(Distance d1,int t)

{

        Distance temp;

        temp.setFeet(d1.getFeet()-t);

        temp.setInch(d1.getInch()-t);

        return temp;

}

Distance mult(Distance d1,int t)
```

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