1. Employee:

**//employee.h**

```
#ifndef EMPLOYEE_H

#define EMPLOYEE_H

using namespace std;

#include<iostream>

#include<string.h>

class Employee              //Base class

{

        int emp_id;

        char name[20];

        double salary;

        public:

        Employee();

        Employee(int ,const char*,double );

        void setId(int );

        void setName(const char*);

        void setSalary(double );

        int getId();

        char* getName();

        double getSalary();

        virtual void display();

        virtual double calSal();

};

#endif EMPLOYEE_H
```

**//salesman.h**

```
#include "employee.h"

class SalesMan:public Employee

{

        int target;

        double intensive;

        public:

                SalesMan();

                SalesMan(int ,const char* ,double ,int ,int );

                void setTarget(int );

                void setIntense(double );

                int getTarget();

                double getIntense()       ;

                void display();

                double calSal();

};
```

**//admin.h**

```
#include "employee.h"

class Admin:public Employee

{

        double allowance;

        public:

                Admin();

                Admin(int ,const char* ,double ,double );
                void setAllow(double );

                double getAllow();

                void display();

                double calSal();

};
```

**//hr.h**

```
#include "employee.h"

class HrManager:public Employee          //step 1

{

        double commission;

        public:

        HrManager();

        HrManager(int ,const char* ,double ,double );

        void setComm(double );

        double getComm();

        void display();

        double calSal()   ;

};
```

**//Employee.cpp**

```
#include "employee.h"

#include "salesman.h"

#include "admin.h"
```

```
{

#include "hr.h"

Employee::	Employee()

	{

		cout<<"\n\nEmp default constructor called\n";

		this->emp_id=0;

		strcpy(this->name,"not_given");

		this->salary=0;

	}

Employee::	Employee(int i,const char* n,double s)

	{

		cout<<"\n\nEmp parameterised called\n";

		this->emp_id=i;

		strcpy(this->name,n);

		this->salary=s;

	}

	void Employee :: setId(int i)	//setters(mutators)

	{

		this->emp_id=i;

	}

	void Employee :: setName(const char* n)		//setters(mutators)

	{

		strcpy(this->name,n);

	}

	void Employee :: setSalary(double s)	//setters(mutators)

	{
```

```
                this->salary=s;

        }

        int Employee :: getId()   //getters(accessors)

        {

                return this->emp_id;

        }

        char* Employee :: getName()     //getters(accessors)

        {

                return this->name;

        }

        double Employee :: getSalary()   //getters(accessors)

        {

                return this->salary;

        }

        void Employee :: display()

        {

                cout<<"\nemployees detail: \nid: "<<this->emp_id<<"\tname: "<<this-
>name<<"\tsalary: "<<this->salary<<"\n";

        }

        double Employee :: calSal()

        {

                return salary;

        }

        SalesMan :: SalesMan():Employee()        //step 2(a)- default constructor of Sales manager and
call given to default constructor of Employee

        {
```

```
		cout<<"\n\nSM default constructor called\n";

		this->target=0;

		this->intensive=0;

	}

	SalesMan :: SalesMan(int i,const char* n,double s,int t,int in):Employee(i,n,s)			//step 
2(b) parameterised constructor of Sales manager and call given to parameterised constructor of 
Employee

	{

		cout<<"\n\nSM parameterised constructor called\n";

		this->target=t;

		this->intensive=in;

	}

	void SalesMan :: setTarget(int t)		//extra setters(mutator) required for Sales manager

	{

		this->target=t;

	}

	void SalesMan :: setIntense(double in)		//extra setters(mutator) required for Sales 
manager

	{

		this->intensive=in;

	}

	int SalesMan :: getTarget()			//extra getters(accessor) required for sales manager

	{

		return this->target;

	}

	double SalesMan :: getIntense()		//extra getters(accessor) required for sales manager

	{
		return this->intensive;

	}

	void SalesMan :: display()

	{

		Employee::display();	//step 3- call given to display function of Employee class

		cout<<"\ntarget: "<<this->target<<"\tintensive: "<<this->intensive;

	}

	double SalesMan :: calSal()			//different implementation

	{

		return getSalary()+intensive;

	}


	Admin :: Admin():Employee()	//step 2(a)-default constructor of Admin and call given to
default constructor of Employee

	{

		cout<<"\n\nAdmin default constructor called\n";

		this->allowance=0;

	}

	Admin :: Admin(int i,const char* n,double s,double a):Employee(i,n,s)	//step 2(b)-
parameterised constructor of Admin and call given to parameterised constructor of Employee

	{

		cout<<"\n\nAdmin parameterised constructor called\n";

		this->allowance=a;

	}

	void Admin :: setAllow(double a)			//extra setters(mutator) required for Admin

	{
		this->allowance=a;
	}
	double Admin :: getAllow()		//extra getters(accessor) required for Admin
	{
		return this->allowance;
	}
	void Admin :: display()	//step 3
	{
		Employee::display();		//call given to display function of Employee class
		cout<<"\tallowance: "<<this->allowance<<"\n";
	}
	double Admin :: calSal()		//different implementation
	{
		return getSalary()+allowance;
	}


	HrManager :: HrManager():Employee()	//step 2(a)-default constructor of HR manager and call
given to default constructor of Employee
	{
		cout<<"\n\nHR default constructor called\n";
		this->commission=0;
	}
	HrManager :: HrManager(int i,const char* n,double s,double c):Employee(i,n,s)	//step 2(b)-
parameterised constructor of HR manager and call given to parameterised constructor of Employee
	{
		cout<<"\n\nHR parameterised constructor called\n";
		this->commission=c;
	}
	void HrManager :: setComm(double c)		//extra setters(mutator) required for HR
manager
	{
		this->commission=c;
	}
	double HrManager :: getComm()		//extra getters(accessor) required for HR manager
	{
		return this->commission;
	}
	void HrManager :: display()
	{
		Employee::display();	//step 3- call given to display function of Employee class
		cout<<"\tCommission: "<<this->commission<<"\n";
	}
	double HrManager :: calSal()		//different implementation
	{
		return getSalary()+commission;
	}
```

**//main.cpp**

```
#include "employee.h"
#include "salesman.h"
#include "admin.h"
#include "hr.h"
```

```
int main()

{

	int id,target;

	char name[20];

	double salary,intensive,allowance,commission;

	Employee *e;

	Employee e1;

	Employee e2(42,"pragati",60000);

	e=&e2;

	e->display();

	SalesMan m1;

	cout<<"enter sale managers id:\n";

	cin>>id;

	cout<<"\nenter the name of sales manager:\n";

	cin>>name;

	cout<<"\nenter salary of sales manager:\n";

	cin>>salary;

	cout<<"\nenter target of sales manager:\n";

	cin>>target;

	cout<<"\nenter intensive for target completion:\n";

	cin>>intensive;

	m1.setId(id);

	m1.setName(name);

	m1.setSalary(salary);
```

```
m1.setTarget(target);

m1.setIntense(intensive);

e=&m1;

e->display();

cout<<"\nCalculated salary is: "<<e->calSal();

SalesMan m2(22,"pragati",50000,45,4500);

e=&m2;

e->display();

cout<<"\nCalculated salary is: "<<e->calSal();

Admin a1;

cout<<"enter admin id:\n";

cin>>id;

cout<<"\nenter name of the admin:\n";

cin>>name;

cout<<"\nenter salary of admin:\n";

cin>>salary;

cout<<"\nallowance for admin:\n";

cin>>allowance;

a1.setId(id);

a1.setName(name);

a1.setSalary(salary);

a1.setAllow(allowance);

e=&a1;

e->display();

cout<<"\nCalculated salary is: "<<e->calSal();
```

```
Admin a2(101,"pragati",50000,4500);

e=&a2;

e->display();

cout<<"\nCalculated salary is: "<<e->calSal();

HrManager h1;

cout<<"\nenter hr managers id:\n";

cin>>id;

cout<<"\nenter name of hr manager:\n";

cin>>name;

cout<<"\nenter salary of hr manager:\n";

cin>>salary;

cout<<"\nenter commission for hr manager:\n";

cin>>commission;

h1.setId(id);

h1.setName(name);

h1.setSalary(salary);

h1.setComm(commission);

e=&h1;

e->display();

cout<<"\nCalculated salary is: "<<e->calSal();

HrManager h2(202,"pragati",50000,5000);

e=&h2;

e->display();

cout<<"\nCalculated salary is: "<<e->calSal();

return 0;
```

```
}
```

2. Complex:

**//Complex.h**

```
using namespace std;

#include<iostream>

class Complex

{

        int real,imag;

        public:

        Complex();

        Complex(int ,int);

        void setReal(int);

        void setImag(int);

        int getReal();

        int getImag();

        void display();

        Complex add(Complex);

        Complex add(int);

        Complex sub(Complex );

        Complex sub(int );

        Complex mult(Complex);

        Complex mult(int);

        Complex div(Complex);
```

```
	}


		Complex div(int );

	};

	Complex add(Complex,int);

	Complex sub(Complex,int);

	Complex mult(Complex,int);

	Complex div(Complex,int);
```

**//complex.cpp**

```
#include "complex.h"

	Complex :: Complex()

	{

		//cout<<"\nDefault constructor called\n";

		this->real=0;

		this->imag=0;

	}

	Complex :: Complex(int real,int imag)

	{

		//cout<<"\nParameterised constructor called\n";

		this->real=real;

		this->imag=imag;

	}

	void Complex :: setReal(int real)

	{

		this->real=real;

	}
```

```
void Complex :: setImag(int imag)

{

        this->imag=imag;

}

int Complex :: getReal()

{

        return this->real;

}

int Complex :: getImag()

{

        return this->imag;

}

void Complex :: display()

{

        cout<<"\ncomplex number: "<<this->real<<"+"<<this->imag<<"i"<<"\n";

}

Complex Complex :: add(Complex c2)

{

        Complex temp;

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

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

        return temp;

}

Complex Complex :: add(int t)

{
        Complex temp;

        temp.real=this->real+t;

        temp.imag=this->imag+t;

        return temp;

}

Complex Complex :: sub(Complex c2)

{

        Complex temp;

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

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

        return temp;

}

Complex Complex :: sub(int t)

{

        Complex temp;

        temp.real=this->real-t;

        temp.imag=this->imag-t;

        return temp;

}

Complex Complex :: mult(Complex c2)

{

        Complex temp;

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

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

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

}
```

**//main.cpp**

```
#include "complex.h"

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();

}
```