/\*

#include<iostream>

using namespace std;

class ComplexNo{

int real,img;

public:

void setComplexNo(){

cout<<"Enter real value: "; cin>>real;

cout<<"Enter imaginary value: "; cin>>img;

}

void showComplexNo(){

cout<<real<<(img>=0?"+":"")<<img<<"i"<<endl;

}

int getReal(){return real;}

int getImg(){return img;}

void setReal(int val){real = val;}

void setImg(int img){this->img = img;}

ComplexNo add(ComplexNo c){

ComplexNo temp;

temp.real = real + c.real;

temp.img = img + c.img;

return temp;

}

ComplexNo operator+(ComplexNo c){

ComplexNo temp;

temp.real = real + c.real;

temp.img = img + c.img;

return temp;

}

ComplexNo operator+(int val){

ComplexNo temp;

temp.real = real + val;

temp.img = img;

return temp;

}

friend ComplexNo operator-(int, ComplexNo);

//friend void operator>>(istream&, ComplexNo&);

friend istream& operator>>(istream&, ComplexNo&);

};

ComplexNo operator+(int val, ComplexNo c){

ComplexNo temp;

temp.setReal( c.getReal() + val );

temp.setImg( c.getImg() + val );

return temp;

}

ComplexNo operator-(int val, ComplexNo c){

ComplexNo temp;

temp.real = c.real - val;

temp.img = c.img - val;

return temp;

}

//void operator>>(istream& in, ComplexNo& c){

istream& operator>>(istream& in, ComplexNo& c){

cout<<"Enter real value: "; in>>c.real;

cout<<"Enter imaginary value: "; in>>c.img;

return in;

}

//void operator<<(ostream& out, ComplexNo c){

ostream& operator<<(ostream& out, ComplexNo c){

out<<c.getReal()<<(c.getImg()>=0?"+":"")

<<c.getImg()<<"i";//<<endl;

return out;

}

int main(){

ComplexNo c1, c2, c3, c4, c5, c6, c7;

//cin>>c1;

//cin>>c2;

//cin>>c3;

cin>>c1>>c2>>c3;

c4 = c1 + c2 + c3;

cout<<"c1="<<c1<<endl;

cout<<"c2="<<c2<<endl;

cout<<"c3="<<c3<<endl;

cout<<"after c4=c1+c2, c4:"<<c4<<endl;

c5=c4+10; //increase real by 10

cout<<"after c5=c4+10, c5:"<<c5<<endl;

c6=2+c5; //increase both real & img by 2

cout<<"after c6=2+c5, c6:"<<c6<<endl;

c7=1-c5; //decrese both real & img by 1

cout<<"after c7=1-c5, c7:"<<c7<<endl;

return 0;

}

\*/

//-------------------------

#include<iostream>

using namespace std;

class ComplexNo{

int real,img;

public:

void setComplexNo(){

cout<<"Enter real value: "; cin>>real;

cout<<"Enter imaginary value: "; cin>>img;

}

void showComplexNo(){

cout<<real<<(img>=0?"+":"")<<img<<"i"<<endl;

}

int getReal(){return real;}

int getImg(){return img;}

void setReal(int val){real = val;}

void setImg(int img){this->img = img;}

ComplexNo add(ComplexNo c){

ComplexNo temp;

temp.real = real + c.real;

temp.img = img + c.img;

return temp;

}

ComplexNo operator+(ComplexNo c){

ComplexNo temp;

temp.real = real + c.real;

temp.img = img + c.img;

return temp;

}

ComplexNo operator+(int val){

ComplexNo temp;

temp.real = real + val;

temp.img = img;

return temp;

}

//void operator++(){

ComplexNo operator++(){

real++; img++;

return \*this;

}

ComplexNo operator++(int x){

ComplexNo temp;

temp.real = real;

temp.img = img;

real++; img++;

return temp;

}

friend ComplexNo operator-(int, ComplexNo);

//friend void operator>>(istream&, ComplexNo&);

friend istream& operator>>(istream&, ComplexNo&);

};

ComplexNo operator+(int val, ComplexNo c){

ComplexNo temp;

temp.setReal( c.getReal() + val );

temp.setImg( c.getImg() + val );

return temp;

}

ComplexNo operator-(int val, ComplexNo c){

ComplexNo temp;

temp.real = c.real - val;

temp.img = c.img - val;

return temp;

}

//void operator>>(istream& in, ComplexNo& c){

istream& operator>>(istream& in, ComplexNo& c){

cout<<"Enter real value: "; in>>c.real;

cout<<"Enter imaginary value: "; in>>c.img;

return in;

}

//void operator<<(ostream& out, ComplexNo c){

ostream& operator<<(ostream& out, ComplexNo c){

out<<c.getReal()<<(c.getImg()>=0?"+":"")

<<c.getImg()<<"i";//<<endl;

return out;

}

int main(){

//int i=10, j=10, k;

//cout<<++i<<endl; //11

//cout<<j++<<endl; //10

//cout<<"i="<<i<<", j="<<j<<endl; //i=11, j=11

ComplexNo c1,c2, c3, c4;

cout<<"Enter real & img for c1: "<<endl;

cin>>c1;

cout<<"c1 = "<<c1<<endl;

cout<<"Enter real & img for c2: "<<endl;

cin>>c2;

cout<<"c1 = "<<c1<<endl;

cout<<"c2 = "<<c2<<endl;

//++c1;

c3 = ++c1;

c4 = c2++;

//cout<<"after ++C1, C1 = "<<c1<<endl;

cout<<"after c3 = ++c1, c1 = "<<c1<<endl;

cout<<"after c3 = ++c1, c3 = "<<c3<<endl;

cout<<"after c4 = c2++, c2 = "<<c2<<endl;

cout<<"after c4 = c2++, c4 = "<<c4<<endl;

return 0;

}

No.4

#include<iostream>

using namespace std;

class Author {

// fields: authorName (string), authorEmail(string),

private:

string autorName, authorEmail;

public:

// add necessary methods including setters and getters,

// AND/OR declare friends, so that the given main() works

friend istream& operator>>(istream& in, Author& a);

friend ostream& operator<<(ostream& out, Author& a);

};

class Book {

// fields:

//bookName (string), authorList(Author\*), noOfAuthors (int), price (float), isbn (string)

private:

string bookName, isbn;

int noOfAuthors;

float price;

Author\* authourList= new Author[10];

public:

// add necessary methods including setters and getters,

// AND/OR declare friends, so that the given main() works

friend istream& operator>>(istream& in, Book& b);

friend ostream& operator<<(ostream& out, Book& b);

};

class Course {

// fields: courseId (string), courseTitle (string), noOfCredits (int), textBook (Book)

private:

string courseId,courseTitle;

int noOfCredits;

Book textbook;

public:

// add necessary methods including setters and getters,

// AND/OR declare friends, so that the given main() works

friend istream& operator>>(istream& in, Course& c);

friend ostream& operator<<(ostream& out, Course& c);

int getCredit(){return noOfCredits;}

};

istream& operator>>(istream& in, Author& a)

{

cout<<"\nEnter author name: ";

in>>a.autorName;

cout<<"Enter author email: ";

in>>a.authorEmail;

return in;

}

istream& operator>>(istream& in, Book& b)

{

cout<<"\nEnter book name: ";

in>>b.bookName;

cout<<"Enter isbn: ";

in>>b.isbn;

cout<<"Enter price: ";

in>>b.price;

cout<<"Enter no. of authors: ";

in>>b.noOfAuthors;

for(int i=0;i<b.noOfAuthors;i++)

{

cin>>b.authourList[i];

}

return in;

}

istream& operator>>(istream& in, Course& c)

{

cout<< "\nEnter courseID: ";

in>>c.courseId;

cout<< "Enter course title: ";

in>>c.courseTitle;

cout<< "Enter course credit: ";

in>>c.noOfCredits;

cin>>c.textbook;

return in;

}

ostream& operator<<(ostream& out, Author& a)

{

cout<<"\nAuthor name: "<<a.autorName;

cout<<"\nAuthor email: "<<a.authorEmail;

return out;

}

ostream& operator<<(ostream& out, Book& b)

{

cout<<"\nBook name: "<<b.bookName;

cout<<"\nisbn: "<<b.isbn;

cout<<"\nPrice: "<<b.price;

cout<<"\nNo. of authors: "<<b.noOfAuthors;

for(int i=0;i<b.noOfAuthors;i++)

{

cout<<b.authourList[i];

}

return out;

}

ostream& operator<<(ostream& out, Course& c)

{

cout<< "\nCourseID: "<<c.courseId;

cout<< "\nCourse title: "<<c.courseTitle;

cout<< "\nCourse credit: "<<c.noOfCredits;

cout<<c.textbook;

return out;

}

void operator+=(int& a,Course c)

{

a=a+c.getCredit();

}

//Complete the following global function

void allocateMemory (Course CArr[],int n)

{

// complete the function

for(int i=0;i<n;i++)

{

//take input for each course.

cout<<"\nCourse "<<i+1<<":";

cin>>CArr[i];

}

}

int main(){

int n, i;

cout<<"How many courses? "; cin>>n;

Course\* courseArr = new Course[n];

allocateMemory(courseArr, n); //it is a global function

for(i=0;i<n;i++){cout<<"\n"<<courseArr[i];}//displaying values.

//for(i=0;i<n;i++)courseArr[i].populateCourse().displayCourse();

int totalCredits = 0;

for(i=0;i<n;i++) totalCredits += courseArr[i];

cout<<"\nTotal no of credits of these courses is: "<<totalCredits<<endl;

return 0;

}

No. 3.

#include<iostream>

using namespace std;

class Author {

// MUST have ID (int), name (string) and email (string) as private fields.

private:

int ID;

string name, email;

public:

// add necessary methods including setters and getters,

// AND/OR declare friends, so that the given main() works

string getName(){return name;}

friend istream& operator>>(istream& in, Author& a);

friend ostream& operator<<(ostream& out, Author& a);

};

class Book {

// MUST have title (string), ISBN\_No (string), price (float) and the following as private fields.

private:

float price;

string title, ISBN\_No;

int noOfAuthor;

Author\* ptr=new Author[10];

public:

// add necessary methods including setters and getters,

// AND/OR declare friends, so that the given main() works

Book()

{

price=0;

title="N/A";

ISBN\_No="N/A";

noOfAuthor=0;

}

Book(string a,string b,float c)

{

price=c;

title=a;

ISBN\_No=b;

noOfAuthor=0;

}

friend istream& operator>>(istream& in, Book& b);

friend ostream& operator<<(ostream& out, Book& b);

string getTitle(){return title;}

float getPrice(){return price;}

bool matchAuthor(string a)

{

for(int i=0;i<noOfAuthor;i++)

{

if(ptr[i].getName()==a){return true;}

}

return false;

}

};

istream& operator>>(istream& in, Author& a)

{

in>>a.name;

cout<<"Enter id: ";

in>>a.ID;

cout<<"Enter email: ";

in>>a.email;

}

istream& operator>>(istream& in, Book& b)

{

cout<<"\nEnter Title: ";

in>>b.title;

cout<<"Enter ISBN\_No: ";

in>>b.ISBN\_No;

cout<<"Enter price: ";

in>>b.price;

cout<<"Enter no. of authors: ";

in>>b.noOfAuthor;

for(int i=0;i<b.noOfAuthor;i++)

{

cout<<"/nEnter name "<<i+1<<": ";

cin>>b.ptr[i];

}

return in;

}

ostream& operator<<(ostream& out, Author& a)

{

cout<<"\nName: "<<a.name;

cout<<"\nid: "<<a.ID;

cout<<"\nEmail: "<<a.email;

}

ostream& operator<<(ostream& out, Book& b)

{

cout<<"\n\nTitle: "<<b.title;

cout<<"\nISBN\_No: "<<b.ISBN\_No;

cout<<"\nPrice: "<<b.price;

cout<<"\nNo. of authors: "<<b.noOfAuthor;

for(int i=0;i<b.noOfAuthor;i++)

{

cout<<b.ptr[i];

}

return out;

}

int main()

{

Book b1, b2("Data Structures", "978-0-7334-2609-4", 550);

//parameterized constructor also ask for no of authors and their details

cout << "Give input for book 1: " << endl;

cin >> b1;

//Should ask title, ISBN\_No, price.

//Then ask #ofAuthors & author info (id, name and email) for all authors.

cout << "Complete information of the books are: " << endl;

cout<< b1 << b2 << endl;

cout<<"For The book titled: "<< b1.getTitle() <<endl;

if(b1.getPrice() <= 1000 || b1.matchAuthor("Ataul Karim") == true)

{

cout<<"Ataul Karim is an author, or the price is <= 1000"<<endl;

}

else

{

cout<<"Ataul Karim is NOT an author & price is > 1000"<<endl;

}

return 0;

}

No.1

#include<iostream>

using namespace std;

class Address {

// MUST have houseNo, roadNo, street, thana, district, zipCode, etc. as private fields

private:

int houseNo,roadNo,zipCode;

string street, thana, district;

public:

Address()

{

houseNo=0,roadNo=0,zipCode=0;

street="N/A",thana="N/A",district="N/A";

}

// add necessary methods including setters and getters,

// AND/OR declare friends, so that the given main() works

friend istream& operator>>(istream& in,Address& a);

friend ostream& operator<<(ostream& out,Address& a);

int get\_zip(){return zipCode;}

};

class Employee{

// MUST have employeeID, name, department and the following as private fields

Address empAddress;

private:

int employeeID;

string name, department;

public:

// add necessary methods including setters and getters,

// AND/OR declare friends, so that the given main() works

Employee()

{

employeeID=0;

string name="N/A",department="N/A";

}

Employee(int a,string b,string c)

{

employeeID=a;

name=b;

department=c;

}

friend istream& operator>>(istream& in,Employee& e);

friend ostream& operator<<(ostream& out,Employee& e);

int get\_zip(){return empAddress.get\_zip();}

};

istream& operator>>(istream& in,Address& a)

{

cout<<"\nEnter houseNo: ";

in>>a.houseNo;

cout<<"Enter roadNo: ";

in>>a.roadNo;

cout<<"Enter zipCode: ";

in>>a.zipCode;

cout<<"Enter street: ";

in>>a.street;

cout<<"Enter thana: ";

in>>a.thana;

cout<<"Enter district: ";

in>>a.district;

return in;

}

istream& operator>>(istream& in,Employee& e)

{

cout<<"\nEnter employeeID: ";

in>>e.employeeID;

cout<<"Enter name: ";

in>>e.name;

cout<<"Enter department: ";

in>>e.department;

cin>>e.empAddress;

return in;

}

ostream& operator<<(ostream& out,Address& a)

{

cout<<"\nEnter houseNo: "<<a.houseNo;

cout<<"\nRoadNo: "<<a.roadNo;

cout<<"\nZipCode: "<<a.zipCode;

cout<<"\nStreet: "<<a.street;

cout<<"\nThana: "<<a.thana;

cout<<"\nDistrict: "<<a.district;

return out;

}

ostream& operator<<(ostream& out,Employee& e)

{

cout<<"\nEmployee ID: "<<e.employeeID;

cout<<"\nname: "<<e.name;

cout<<"\ndepartment: "<<e.department;

cout<<e.empAddress;

return out;

}

int main(){

Employee e1, e2(4161, "S K Dey", "CSE");

//parameterized constructor also ask for address details

cout << "Give input for employee 1: " << endl;

cin >> e1;

//Should ask employeeID, name, department.

//Then ask address related information.

cout << "Complete information of the employees are: " << endl;

cout<< e1 << e2 << endl;

cout<<"The employee e1 ";

if(e1.get\_zip() == 1229){ cout << "lives in Bashundhara R/A." << endl;}

else {cout << "does NOT lives in Bashundhara R/A." << endl;}

//comparing zipCode field

return 0;

}