#include<iostream>

using namespace std;

class Artikal {

char \*sifra;

char \*naziv;

double cijena;

int kolicinaNaSkladistu;

public:

Artikal(const char \*sifra = "---", const char\*naziv = "---", double cijena = 0,int kolicina = 0) {

this->sifra = new char[strlen(sifra) + 1];

strcpy\_s(this->sifra, strlen(sifra) + 1, sifra);

this->naziv = new char[strlen(naziv) + 1];

strcpy\_s(this->naziv, strlen(naziv) + 1, naziv);

this->cijena = cijena;

kolicinaNaSkladistu = kolicina;

}

Artikal(const Artikal &original) {

this->sifra = new char[strlen(original.sifra) + 1];

strcpy\_s(this->sifra, strlen(original.sifra) + 1, original.sifra);

this->naziv = new char[strlen(original.naziv) + 1];

strcpy\_s(this->naziv, strlen(original.naziv) + 1, original.naziv);

this->cijena = original.cijena;

kolicinaNaSkladistu = original.kolicinaNaSkladistu;

}

~Artikal() {

delete[]sifra; sifra = nullptr;

delete[]naziv; naziv = nullptr;

}

Artikal& operator =(Artikal &original) {

this->sifra = new char[strlen(original.sifra) + 1];

strcpy\_s(this->sifra, strlen(original.sifra) + 1, original.sifra);

this->naziv = new char[strlen(original.naziv) + 1];

strcpy\_s(this->naziv, strlen(original.naziv) + 1, original.naziv);

this->cijena = original.cijena;

kolicinaNaSkladistu = original.kolicinaNaSkladistu;

return \*this;

}

Artikal operator-=(int kolicina) {

if (kolicina <= kolicinaNaSkladistu) {

this->kolicinaNaSkladistu -= kolicina;

}

return \*this;

}

Artikal operator+=(int kolicina) {

this->kolicinaNaSkladistu += kolicina;

return \*this;

}

//sufiksni

Artikal operator++(int v) {

Artikal povratni(sifra, naziv, kolicinaNaSkladistu);

cijena++;

return povratni;

}

//prefiksni

Artikal operator++() {

++cijena;

return \*this;

}

//sufiksni

Artikal operator--(int v) {

Artikal povratni(sifra, naziv, kolicinaNaSkladistu);

cijena--;

return povratni;

}

//prefiksni

Artikal operator--() {

--cijena;

return \*this;

}

bool operator ==(const Artikal &drugi) {

return (strcmp(this->sifra, drugi.sifra) == 0);

}

friend ostream & operator <<(ostream &COUT, Artikal &a) {

COUT << "Artikal " << a.naziv << " ( " << a.sifra << " ) "<<a.cijena<<" KM\n";

COUT << "Kolicina na skladistu: " << a.kolicinaNaSkladistu << endl;

return COUT;

}

friend istream & operator >> (istream &CIN, Artikal &a) {

char temp[100];

cout << "Unesite naziv artikla: ";

CIN.getline(temp, 100);

a.naziv = new char[strlen(temp) + 1];

strcpy\_s(a.naziv, strlen(temp) + 1, temp);

cout << "Unesite šifru artikla: ";

CIN.getline(temp, 100);

a.sifra = new char[strlen(temp) + 1];

strcpy\_s(a.sifra, strlen(temp) + 1, temp);

cout << "Unesite cijenu artikla: ";

CIN >> a.cijena;

cout << "Unesite količinu na skladištu: ";

CIN >> a.kolicinaNaSkladistu;

CIN.ignore();

return CIN;

}

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

//Operator -= na način da se stanje na skladištu artikla promijeni za proslijeđenu vrijednost

//Operator ++ na način da se cijena artikla uveća za 1

//Operator -- na način da se cijena artikla umanji za 1

//Operator == koji poredi artikle na osnovu sifre

//Preklopiti operatore za upis i ispis podataka o artiklu

int GetKolicinaNaSkladistu() { return kolicinaNaSkladistu; }

double GetCijena() { return cijena; }

char \*GetSifra() { return sifra; }

};

class StavkaRacuna {

Artikal artikal;

int \*kolicina;

public:

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

StavkaRacuna() {

kolicina = new int;

\*kolicina = 0;

}

StavkaRacuna(Artikal &artikal, int kolicina) {

this->artikal = artikal;//zbog ovoga moramo preklopiti operator = za artikal

this->kolicina = new int;

\*(this->kolicina) = kolicina;

}

StavkaRacuna(int kolicina) {

cin >> artikal;

this->kolicina = new int;

\*(this->kolicina) = 0;

}

StavkaRacuna(StavkaRacuna &original) {

this->artikal = original.artikal;

kolicina = new int;

\*kolicina = \*original.kolicina;

}

StavkaRacuna& operator = (StavkaRacuna &original) {

this->artikal = original.artikal;

delete kolicina;

kolicina = new int;

\*kolicina = \*original.kolicina;

return \*this;

}

StavkaRacuna& operator -= (int kolicina) {

this->artikal -= kolicina;

return \*this;

}

StavkaRacuna& operator += (int kolicina) {

this->artikal += kolicina;

return \*this;

}

bool operator ==(StavkaRacuna &obj) {

if (obj.artikal == artikal)

return true;

return false;

}

//sufiksni

StavkaRacuna operator++(int v) {

StavkaRacuna povratni(artikal,\*kolicina);

(\*kolicina)++;

return povratni;

}

//prefiksni

StavkaRacuna operator++() {

(\*kolicina)++;

return \*this;

}

friend ostream & operator <<(ostream &COUT, StavkaRacuna &stavkaRacuna) {

COUT << stavkaRacuna.artikal;

COUT << "Kolicina: " << \*stavkaRacuna.kolicina << endl;

return COUT;

}

friend istream & operator >> (istream &CIN, StavkaRacuna &stavkaRacuna) {

cout << "Unesite artikal:\n";

CIN >> stavkaRacuna.artikal;

cout << "Unesite količinu: ";

CIN >> (\*stavkaRacuna.kolicina);

CIN.ignore();

return CIN;

}

double GetCijena() { return artikal.GetCijena()\*(\*kolicina); }

double GetKolicina() { return (\*kolicina); }

};

int pom = 1000;

class Racun {

int brojRacuna;

StavkaRacuna stavke[100];

int brojStavki;

double iznosBezPDV;

double iznosSaPDV;

bool naplaceno;

public:

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

Racun() {

brojRacuna = pom++;

brojStavki = 0;

iznosBezPDV = 0;

iznosSaPDV = 0;

naplaceno = false;

}

Racun(Racun &original) {

brojRacuna = original.brojRacuna;

brojStavki = original.brojStavki;

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

{

stavke[i] = original.stavke[i];//da nemamo preklopljen operator = za stavke ovo ne bi radilo

}

iznosSaPDV = original.iznosSaPDV;

iznosBezPDV = original.iznosBezPDV;

naplaceno = original.naplaceno;

}

Racun operator = (Racun &original) {

brojRacuna = original.brojRacuna;

brojStavki = original.brojStavki;

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

{

stavke[i] = original.stavke[i];//da nemamo preklopljen operator = za stavke ovo ne bi radilo

}

iznosSaPDV = original.iznosSaPDV;

iznosBezPDV = original.iznosBezPDV;

naplaceno = original.naplaceno;

return \*this;

}

//Operator() - dodaje nove artikle u stavke racuna (ulazni parametri su artikal i kolicina), kolicina nije obavezni parametar

//Voditi računa o tome da li je račun već naplaćen

//Ukoliko nema dovoljno artikala na stanju, ispisati odgovarajuću poruku

//Ukoliko artikal već postoji, potrebno je samo uvećati količinu za odgovarajuću stavku

//Prilikom svakog dodavanja artikla, azurirati vrijednosti iznosBezPDV i iznosSaPDV

Racun& operator()(Artikal &a, int kolicina) {

if (naplaceno) {

cout << "Racun je vec naplacen, nemoguce dodavati artikle!\n";

return \*this;

}

if (a.GetKolicinaNaSkladistu() < kolicina) {

cout << "Nema dovoljno artikla na stanju. Nemoguće dodati artikal.\n";

return \*this;

}

StavkaRacuna novaStavka(a, kolicina);

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

{

if (stavke[i] == novaStavka) {//ako ima taj artikal već, samo uvećaj količinu

++stavke[i];

iznosBezPDV += a.GetCijena()\*kolicina;

iznosSaPDV = iznosBezPDV + iznosBezPDV \* 0.17;

//azuriraj stanje na skladistu

stavke[i] -= kolicina;

return \*this;

}

}

stavke[brojStavki] = novaStavka;

//azuriraj stanje na skladistu

stavke[brojStavki] -= kolicina;

brojStavki++;

iznosBezPDV += a.GetCijena()\*kolicina;

iznosSaPDV = iznosBezPDV + iznosBezPDV \* 0.17;

return \*this;

}

//Operator -= uklanja artikal iz stavki na osnovu njegove sifre

//azurirati vrijednosti iznosBezPDV i iznosSaPDV

Racun &operator -=(char \*sifra) {

Artikal zaUkloniti(sifra);

StavkaRacuna stavka(zaUkloniti, 0);

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

{

if (stavke[i] == stavka) {

//atzuriranje stanja na skladistu

stavke[i] += stavke[i].GetKolicina();

iznosBezPDV -= stavke[i].GetCijena();

iznosSaPDV = iznosBezPDV + iznosBezPDV \* 0.17;

//uklanjanje stavke iz liste

//uklanjanje stavke iz liste

for (int j = i; j < brojStavki-1; j++)

{

stavke[j] = stavke[j+1];

} brojStavki--;

return \*this;

}

}

cout << "Artikal nije pronađen!\n";

return \*this;

}

double PrintRacun() {

//Ispisuje sve stavke računa zajedno sa iznosom za plaćanje

//Omogućiti da se unese iznos koji je korisnik uplatio, te se

//vraća iznos kusura

naplaceno = true;

cout << "======================== Racun br. " << brojRacuna<<" ======================== " << endl;

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

{

cout << stavke[i];

cout << "==============================================\n";

}

cout << "\nIznos sa PDV-om: " << iznosSaPDV << " KM\n";

cout << "\nIznos bez PDV-a: " << iznosBezPDV << " KM\n\n";

double uplata;

do {

cout << "Unesite iznos koji uplaćujete: ";

cin >> uplata;

} while (uplata < iznosSaPDV);

cin.ignore();

return uplata-iznosSaPDV;

}

};

void main() {

Racun racun;

Artikal artikli[3];

for (int i = 0; i < 3; i++)

{

cin >> artikli[i];

racun(artikli[i], 2);

}

//pokusaj dodavanja istog artikla na racun

racun(artikli[0], 2);

char sifra[20];

cout << "Unesite sifru artikla kojeg želite ukloniti: ";

cin.getline(sifra, 20);

racun -= sifra;

racun(artikli[1], 1);

cout<<"Kusur: "<<racun.PrintRacun()<<" KM"<<endl;

system("pause>0");

}