#include<iostream>

using namespace std;

template<class T1, class T2>

class Kolekcija

{

T1\* \_elementi1;

T2\* \_elementi2;

int \_trenutno;

public:

Kolekcija()

{

\_trenutno = 0;

\_elementi1 = nullptr;

\_elementi2 = nullptr;

}

~Kolekcija()

{

delete[] \_elementi1;

\_elementi1 = nullptr;

delete[] \_elementi2;

\_elementi2 = nullptr;

}

void DodajElemente(const T1& el1, const T2& el2)

{

T1\* temp1 = new T1[\_trenutno + 1];

T2\* temp2 = new T2[\_trenutno + 1];

for (int i = 0; i < \_trenutno; i++)

{

temp1[i] = \_elementi1[i];

temp2[i] = \_elementi2[i];

}

delete[] \_elementi1;

delete[] \_elementi2;

\_elementi1 = temp1;

\_elementi2 = temp2;

\_elementi1[\_trenutno] = el1;

\_elementi2[\_trenutno] = el2;

\_trenutno++;

}

int GetTrenutno() const { return \_trenutno; }

friend ostream& operator<<<>(ostream&, const Kolekcija&);

};

template<class T1, class T2>

ostream& operator<<(ostream& cout, const Kolekcija<T1, T2>& kol)

{

for (size\_t i = 0; i < kol.\_trenutno; i++)

{

cout << kol.\_elementi1[i] << "\t" << kol.\_elementi2[i] << endl;

}

return cout;

}

char \* AlocirajNizKaraktera(const char \* sadrzaj) {

if (sadrzaj == nullptr)

return nullptr;

int vel = strlen(sadrzaj) + 1;

char \* temp = new char[vel];

strcpy\_s(temp, vel, sadrzaj);

return temp;

}

//U programu implementirati nasljeđivanje između odgovarajućih klasa

//Klasu Osoba deklarisati kao apstraktnu.

class Osoba

{

char\* \_ime;

char\* \_prezime;

char\* \_adresa;

char \_telefon[15];

public:

Osoba(const char\* ime, const char\* prezime, const char\* adresa, const char\* telefon)

{

\_ime = AlocirajNizKaraktera(ime);

\_prezime = AlocirajNizKaraktera(prezime);

\_adresa = AlocirajNizKaraktera(adresa);

strncpy\_s(\_telefon, telefon, \_TRUNCATE);

}

Osoba(const Osoba& o)

{

\_ime = AlocirajNizKaraktera(o.\_ime);

\_prezime = AlocirajNizKaraktera(o.\_prezime);

\_adresa = AlocirajNizKaraktera(o.\_adresa);

strncpy\_s(\_telefon, o.\_telefon, \_TRUNCATE);

}

~Osoba()

{

delete[] \_ime;

\_ime = nullptr;

delete[] \_prezime;

\_prezime = nullptr;

delete[] \_adresa;

\_adresa = nullptr;

}

void Info()

{

cout << "Ime i prezime: " << \_ime << " " << \_prezime << endl;

cout << "Adresa: " << \_adresa << endl;

cout << "Telefon: " << \_telefon << endl;

}

};

class Proizvod

{

char\* \_naziv;

char\* \_opis;

double \_cijena;

public:

Proizvod(const char\* naziv, const char\* opis, double cijena)

{

\_naziv = AlocirajNizKaraktera(naziv);

\_opis = AlocirajNizKaraktera(opis);

\_cijena = cijena;

}

~Proizvod()

{

delete[] \_naziv;

\_naziv = nullptr;

delete[] \_opis;

\_opis = nullptr;

}

friend ostream& operator<<(ostream&, const Proizvod&);

};

ostream& operator<<(ostream& cout, const Proizvod& p)

{

cout << "Naziv: " << p.\_naziv << endl;

cout << "Opis: " << p.\_opis << endl;

cout << "Cijena: " << p.\_cijena << endl;

return cout;

}

class Narudzba {

char \_datum[12]; //Format: dd.MM.yyyy.

Kolekcija<Proizvod\*, int> \_proizvodi; //Podaci o naručenim proizvodima i količinama

public:

//Potrebne osnovne funkcije za rad sa klasom

//Funkciju GetIznosNarudzbe koja vraća ukupan iznos narudžbe na osnovu

//podataka o cijenama proizvoda i naručenim količinama. Funkcija prima neobavezni parametar popust

//koji inicijalno ima vrijednost 0, što znači da ne postoji popust na postojeće cijene proizvoda.

//Funkciju za ispis svih vrijednosti atributa klase.

};

class Klijent

{

int \_id;

char\* \_email;

Kolekcija<Narudzba, double> \_narudzbe; //Podaci o narudžbama sa ukupnim utrošenim iznosom u KM

public:

//Potrebne konstruktor i destruktor funkcije.

//Funkciju za evidenciju nove narudžbe.

//Funkciju koja vraća logičku vrijednost true ukoliko je klijent za odabrani mjesec u godini imao

//potrošnju veću od zadate. Prototip funkcije: bool IsPotrosnjaVeca(int mjesec, double potrosnja);

//Za pronalazak potrosnje po mjesecima koristiti podatak o datumu narudžbe.

//Funkciju za izmjenu e-mail adrese.

//Funkciju za ispis svih vrijednosti atributa klase.

};

class VIPKlijent

{

char\* \_parkingOznaka;

double \_popust;

public:

//Potrebne konstruktor i destruktor funkcije

//Funkciju za evidenciju nove narudžbe sa zaračunatim popustom na sve proizvode.

//Funkciju koja posljednjoj narudžbi klijenta dodaje gratis proizvod proslijeđen kao parametar.

//Funkciju za ispis svih vrijednosti atributa klase

};

//Funkcija treba da pronađe sve one klijente koji su za odabrani mjesec ostvarili potrošnju veću

//od zadate. Ispisati njihove podatke i vratiti ukupan broj pronađenih klijenata.

int PronadjiKlijentePoPotrosnji(Osoba\* niz[], int max);

//Pronaći sve VIP klijente, te im dodati gratis proizvod proslijeđen kao parametar.

void UkljuciGratisProizvodVIPKlijentima(Osoba\* niz[], int max, const Proizvod& p);

//Svim klijentima postaviti mail adresu u formatu: ime.prezime@fit.ba.

void GenerisiMailAdreseKlijentima(Osoba\* niz[], int max);

//Omogućiti pretragu klijenata po imenu ili prezimenu (ispisati informacije

//o svim klijentima koji sadrže uneseni tekst u dijelu imena ili prezimena.

void PronadjiKlijentePoImenuPrezimenu(Osoba\* niz[], int max);

int main()

{

//Testirati sve dostupne funkcionalnosti uz obaveznu provjeru globalnih funkcija.

//VIP klijentima sve narudžbe kreirati uz zaračunati popust

system("pause");

return 0;

}