#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. SVE KLASE TREBAJU POSJEDOVATI ADEKVATAN DESTRUKTOR

2. NAMJERNO IZOSTAVLJANJE KOMPLETNIH I/ILI POJEDINIH DIJELOVA DESTRUKTORA KOJI UZROKUJU RUNTIME ERROR ÆE BITI OZNACENO KAO "RE"

3. SPAŠAVAJTE PROJEKAT KAKO BI SE SPRIJEÈILO GUBLJENJE URAÐENOG ZADATKA

4. NAZIVI FUNKCIJA, TE BROJ I TIP PARAMETARA MORAJU BITI IDENTIÈNI ONIMA KOJI SU KORIŠTENI U TESTNOM CODE-U, OSIM U SLUÈAJU DA POSTOJI ADEKVATAN RAZLOG ZA NJIHOVU MODIFIKACIJU. OSTALE, POMOÆNE FUNKCIJE MOŽETE IMENOVATI I DODAVATI PO ŽELJI.

5. BEZ OBZIRA NA TO DA LI SU ISPITNI ZADACI URAÐENI, SVI STUDENTI KOJI SU PRISTUPILI ISPITU MORAJU PREDATI SVOJ RAD

6. ZA POTREBE TESTIRANJA, UNUTAR MAIN FUNKCIJE MOZETE DODAVATI NOVE TESTNE PODATKE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char\* crt = "\n----------------------------------------------------\n";

class Datum {

int\* \_dan, \* \_mjesec, \* \_godina;

public:

Datum(int d, int m, int g) {

\_dan = new int(d);

\_mjesec = new int(m);

\_godina = new int(g);

}

~Datum() {

delete \_dan; \_dan = nullptr;

delete \_mjesec; \_mjesec = nullptr;

delete \_godina; \_godina = nullptr;

}

friend ostream& operator<<(ostream& COUT, Datum obj) {

COUT << \*obj.\_dan << "/" << \*obj.\_mjesec << "/" << \*obj.\_godina << endl;

return COUT;

}

};

template<class T, int velicina>

class FITArray {

T \_niz[velicina];

int \_trenutno;

bool \_dozvoliDupliciranje;

public:

FITArray(bool dozvoliDupliciranje = true) { \_dozvoliDupliciranje = dozvoliDupliciranje; \_trenutno = 0; }

int GetTrenutno() { return \_trenutno; }

T\* GetNiz() { return \_niz; }

};

class PolozeniPredmet {

char\* \_naziv;

int \_ocjena;

Datum \_datumPolaganja;

public:

~PolozeniPredmet() { delete[] \_naziv; \_naziv = nullptr; }

friend ostream& operator<<(ostream& COUT, const PolozeniPredmet& obj) {

COUT << obj.\_naziv << "(" << obj.\_ocjena << ")" << obj.\_datumPolaganja << endl;

return COUT;

}

};

class Student {

static int \_indeksBrojac; //sluzi za odredjivanje narednog broja indeksa koji ce biti dodijeljen studentu, pocevsi od 200000

const int \_indeks;

char\* \_imePrezime;

FITArray<PolozeniPredmet, 40> \_polozeniPredmeti;

public:

~Student() {

delete[] \_imePrezime;

\_imePrezime = nullptr;

}

friend ostream& operator<<(ostream& COUT, Student& obj) {

COUT << obj.\_indeks << " " << obj.\_imePrezime << endl << obj.\_polozeniPredmeti;

return COUT;

}

};

void main() {

const int max = 20;

Datum jucer(19, 4, 2021), danas(20, 4, 2021), sutra(21, 4, 2021);

cout << jucer << danas << sutra << endl;

jucer = danas;

cout << jucer << danas << sutra << endl;

//parametar tipa bool odredjuje da li je u nizu dozvoljeno dupliciranje elemenata

FITArray<int, max> nizIntegera(false);

for (size\_t i = 0; i < max - 1; i++)

nizIntegera += i;//dodaje novog clana niza

if (!(nizIntegera += 6))//pokusavamo dodati dupli clan niza

cout << "Element nije dodan u niz" << endl;

cout << crt << nizIntegera << crt; //ispisuje sve clanove niza

nizIntegera -= 9; //uklanja clan niza sa vrijednoscu 9

nizIntegera -= 17;

FITArray<int, max> noviNizIntegera(nizIntegera);

cout << crt << noviNizIntegera << crt;

///\*parametri odredjuju lokacije (indekse u nizu) elemenata OD - DO koje je potrebno vratiti. u slucaju da u nizu ne postoji trazeni broj elemenata funkcija treba da vrati sve element od lokacije OD pa do posljednje dostupnog elementa \*/

cout << "Clanovi niza od lokacije 2 do lokacije 5 su -> " << noviNizIntegera(2, 5) << crt;

Student adel("Adel Handzic"), jasmin("Jasmin Azemovic");

PolozeniPredmet prII(jucer, "PRII", 8), prIII(sutra, "PRIII", 7);

adel.DodajPredmet(prII);

adel.DodajPredmet(prIII);

cout << adel << endl;

////vraca prosjecnu ocjenu studenta

cout << "Prosjecna ocjena -> " << adel.GetProsjek() << crt;

jasmin = adel;

cout << jasmin << endl;

cout << "Prosjecna ocjena -> " << jasmin.GetProsjek() << crt;

}