#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 ĆE BITI OZNACENO KAO "RE"

3. SPAŠAVAJTE PROJEKAT KAKO BI SE SPRIJEČILO GUBLJENJE URAĐENOG ZADATKA

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

5. IZUZETAK BACITE U FUNKCIJAMA U KOJIMA JE TO NAZNAČENO.

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

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

const char\* nedozvoljena\_operacija = "Nedozvoljena operacija";

const char\* not\_set = "NOT\_SET";

const int min\_polgavlja = 3;

const int min\_karaktera\_po\_poglavlju = 30;

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;

}

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; \_elementi1 = nullptr;

}

T1 \* getElementi1Pok() { return \_elementi1; }

T2 \* getElementi2Pok() { return \_elementi2; }

T1 & getElement1(int lokacija) { return \_elementi1[lokacija]; }

T2 & getElement2(int lokacija) { return \_elementi2[lokacija]; }

int getTrenutno() { return \_trenutno; }

friend ostream & operator<<(ostream & COUT, Kolekcija<T1, T2> & obj) {

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

COUT << obj.getElement1(i) << " " << obj.getElement2(i) << endl;

return COUT;

}

};

class Poglavlje {

char \* \_naslov;

char \* \_sadrzaj;

bool \_prihvaceno;

int \_ocjena;//da bi se poglavlje smatralo prihvacenim ocjena mora biti u opsegu od 6 - 10

public:

Poglavlje(const char \* naslov = nullptr, const char \* sadrzaj = nullptr)

:\_ocjena(0), \_prihvaceno(false) {

\_naslov = AlocirajNizKaraktera(naslov);

\_sadrzaj = AlocirajNizKaraktera(sadrzaj);

}

Poglavlje() {

delete[] \_naslov; \_naslov = nullptr;

delete[] \_sadrzaj; \_sadrzaj = nullptr;

}

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

if (obj.\_naslov == nullptr || obj.\_sadrzaj == nullptr)

return COUT;

COUT << endl << obj.\_naslov << endl << obj.\_sadrzaj << endl;

if (obj.\_prihvaceno)

COUT << "Ocjena: " << obj.\_ocjena << endl;;

return COUT;

}

char \* GetNaslov() { return \_naslov; }

char \* GetSadrzaj() { return \_sadrzaj; }

bool GetPrihvaceno() { return \_prihvaceno; }

int GetOcjena() { return \_ocjena; }

};

class ZavrsniRad {

char\* \_tema;

vector<Poglavlje> \_poglavljaRada;

string \_datumOdbrane;

float \_konacnaOcjena; //prosjek ocjena svih poglavlja u zavrsnom radu koja se izracunava u momentu zakazivanja odbrane

public:

ZavrsniRad(const char \* nazivTeme = nullptr) : \_konacnaOcjena(0), \_datumOdbrane(not\_set) {

\_tema = AlocirajNizKaraktera(nazivTeme);

}

ZavrsniRad(const ZavrsniRad& org) : \_poglavljaRada(org.\_poglavljaRada), \_konacnaOcjena(org.\_konacnaOcjena), \_datumOdbrane(org.\_datumOdbrane) {

\_tema = AlocirajNizKaraktera(org.\_tema);

}

~ZavrsniRad() {

delete[] \_tema; \_tema = nullptr;

}

char\* GetNazivTeme()const { return \_tema; }

vector<Poglavlje>& GetPoglavlja() { return \_poglavljaRada; };

string GetDatumOdbrane()const { return \_datumOdbrane; }

float GetOcjena() { return \_konacnaOcjena; }

void SetDatumOdbrane(string datumOdbrane) { \_datumOdbrane = datumOdbrane; }

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

COUT << "Tema rada: " << obj.\_tema << endl;

COUT << "Sadrzaj: " << endl;

for (size\_t i = 0; i < obj.\_poglavljaRada.size(); i++)

COUT << obj.\_poglavljaRada[i] << endl;

COUT << "Datum odbrane rada: " << obj.\_datumOdbrane << endl << " Ocjena: " << obj.\_konacnaOcjena << endl;

return COUT;

}

};

class Osoba {

protected:

string \_imePrezime;

public:

Osoba(string imePrezime) : \_imePrezime(imePrezime) {}

string GetImePrezime() { return \_imePrezime; }

virtual void Info() = 0;

};

class Nastavnik {

//Parametar string predstavlja broj indeksa studenta koji prijavljuje zavrsni rad kod odredjenog nastavnika

Kolekcija<string, ZavrsniRad> \_teme;

public:

Kolekcija<string, ZavrsniRad>& GetTeme() { return \_teme; };

};

int main() {

cout << crt << "UPLOAD RADA OBAVEZNO IZVRSITI U ODGOVARAJUCI FOLDER NA FTP SERVERU" << endl;

cout << "U slucaju da je Upload folder prazan pritisnite tipku F5" << crt;

const int max = 4;

Nastavnik \* nastavnici[max];

nastavnici[0] = new Nastavnik("Denis Music");

nastavnici[1] = new Nastavnik("Zanin Vejzovic");

nastavnici[2] = new Nastavnik("Jasmin Azemovic");

nastavnici[3] = new Nastavnik("Emina Junuz");

//parametri: naziv zavrsnog rada

ZavrsniRad multimedijalni("Multimedijalni informacijski sistem za visoko - obrazovnu ustanovu");

ZavrsniRad podrsa\_operaterima("Sistem za podršku rada kablovskog operatera");

ZavrsniRad analiza\_sigurnosti("Prakticna analiza sigurnosti bežičnih računarskih mreža");

ZavrsniRad kriptografija("Primjena teorije informacija u procesu generisanja kriptografskih ključeva");

/\*u zavrsni rad dodaje novo poglavlje i njegov sadrzaj. ukoliko poglavlje vec postoji u zavrsnom radu, funkcija tom poglavlju treba dodati novi sadrzaj i pri tome zadrzi postojeci (izmedju postojeceg i novog sadrzaja se dodaje prazan prostor). u slucaju da poglavlje ne postoji, ono se dodaje zajedno sa sadrzaje\*/

//parametri: nazivPoglavlja, sadrzajPoglavlja

multimedijalni.DodajPoglavlje("Uvod", "U ovom poglavlju ce biti rijeci");

multimedijalni.DodajPoglavlje("Uvod", "o multimedijalnim sistemima koji se danas koriste");

multimedijalni.DodajPoglavlje("Uvod", "u savremenom poslovanju");

multimedijalni.DodajPoglavlje("Vrste multimedijalnih sistema", "Danas se moze govoriti o nekoliko vrsta multimedijalnih sistema, a neke od najznacajnijih su ...");

multimedijalni.DodajPoglavlje("Teorija multimedije", "Sadrzaj koji bi trebao stajati na pocetku treceg poglavlja zavrsnog rada o multimediji studenta IB130011");

multimedijalni.DodajPoglavlje("Zakljucak", "U ovom radu su predstavljeni osnovni koncepti i prakticna primjena...");

try {

/\*funkcija OcijeniPoglavlje, na osnovu naziva poglavlja, dodjeljuje ocjenu poglavlju te ukoliko je ocjena pozitivna (6 - 10) onda poglavlje oznacava prihvacenim. U slucaju da ocjena nije validna ili poglavlje ne postoji, funkcija baca izuzetak sa odgovarajucom porukom\*/

//parametri:nazivPoglavlja, ocjena

multimedijalni.OcijeniPoglavlje("Uvod", 8);

multimedijalni.OcijeniPoglavlje("Vrste multimedijalnih sistema", 8);

multimedijalni.OcijeniPoglavlje("Teorija multimedije", 9);

multimedijalni.OcijeniPoglavlje("Zakljucak", 7);

multimedijalni.OcijeniPoglavlje("Naziv poglavlja ne postoji", 8);

}

catch (exception & err) {

cout << "Greska -> "<< err.what() << endl;

}

/\*funkcija DodajZavrsniRad ima zadatak da odredjenom nastavniku dodijeli mentorstvo na zavrsnom radu. zavrsni rad se dodaje studentu sa brojem indeksa proslijedjenim kao prvi parametar.

sprijeciti dodavanje zavrsnih radova sa istom temom\*/

//parametri: brojIndeksa, zavrsniRad

if (nastavnici[0]->DodajZavrsniRad("IB130011", multimedijalni))

cout << "Zavrsni rad uspjesno dodat!" << endl;

if (nastavnici[0]->DodajZavrsniRad("IB120051", podrsa\_operaterima))

cout << "Zavrsni rad uspjesno dodat!" << endl;

if (!nastavnici[0]->DodajZavrsniRad("IB120056", podrsa\_operaterima))//dupliranje rada

cout << "Zavrsni rad nije dodat!" << endl;

if (!nastavnici[0]->DodajZavrsniRad("IB120051", kriptografija)) //studentu vec dodijeljen rad

cout << "Zavrsni rad nije dodat!" << endl;

if (nastavnici[1]->DodajZavrsniRad("IB140102", analiza\_sigurnosti))

cout << "Zavrsni rad uspjesno dodat!" << endl;

if (nastavnici[2]->DodajZavrsniRad("IB140002", kriptografija))

cout << "Zavrsni rad uspjesno dodat!" << endl;

/\*funkcija ZakaziOdbranuRada ima zadatak da studentu sa proslijedjenim brojem indeksa zakaze odbranu zavrsnog rada sto podrazumijeva definisanje

datuma odbrane. odbrana rada se moze zakazati samo studentu koji je rad prethodno prijavio i pri tom su zadovoljeni sljedeci uslovi:

1. zavrsni rad ima broj poglavlja veci od minimalnog

2. svako poglavlje ima broj karaktera veci od minimalnog

3. svako poglavlje je prihvaceno/odobreno

ukoliko su zadovoljeni prethodni kriteriji, izracunava se konacna ocjena rada (prosjek ocjena svih poglavlja), postavlja datum odbrane rada i vraca pokazivac na rad kome je zakazan odbrana.

u slucaju da student sa primljenim brojem indeksa nije prijavio zavrsni rad ili neki od postavljenih kriterija nije zadovoljen, funkcija vraca nullptr.

\*/

//parametri: brojIndeksa, datumOdbrane

ZavrsniRad \* zr1 = nastavnici[0]->ZakaziOdbranuRada("IB130011", "25.09.2018");

if (zr1 != nullptr)

cout << \*zr1 << endl;

zr1 = nastavnici[0]->ZakaziOdbranuRada("IB130111", "25.09.2018");//student sa brojem indeksa IB130111 jos uvijek nije prijavio rad

if (zr1 != nullptr)

cout << \*zr1 << endl;

//ispisuje sve podatke o nastavniku i njegovim mentorstvima

nastavnici[0]->Info();

/\*Funkcija PosaljiPozivZaDodjeluNagrada ima zadatak da svim studentima koji su uspjesno okoncali svoj zavrsni rad kod odredjenog nastavnika/mentora

i tom prilikom ostvarili ocjenu vecu od proslijedjene, u zasebnom thread.u, posalje email poruku (mail adresa: brojIndeksa@edu.fit.ba) sa sadrzajem:

"Postovani {brojIndeksa}, uzimajuci u obzir cinjenicu da ste kod mentora {imePrezimeMentora} uspjesno odbranili rad sa ocjenom {ocjena} cast nam je pozvati vas na dodjelu nagrada za najbolje studente koja ce se odrzatu u na FIT-u 03.07.2019. godine." .

funkcija treba da vrati sadrzaj svih poslatih email poruka, a ukoliko niti jedan od nastavnika ne posjeduje evidentirano mentorstvo na zavrsnom radu, funkcija vraca not\_set\*/

cout << "Studenti za dodjelu nagrada: " << PosaljiPozivZaDodjeluNagrada(nastavnici, max, 9.5) << endl;

for (int i = 0; i < max; i++) {

delete nastavnici[i];

nastavnici[i] = nullptr;

}

system("pause>0");

return 0;

}