- *Kreirajte* ***.doc*** *fajl sa vašim brojem indeksa (npr. IB160XXX.doc* ***BEZ IMENA I PREZIMENA****), te na kraju ispita u njega kopirajte rješenja vaših zadataka.*
- *Tokom izrade ispita nije dozvoljeno korištenje help-a*
- *Tokom izrade ispita mogu biti pokrenuta samo tri programa: PDF Reader (ispitni zadaci), MS Visual Studio, MS Word (u koji ćete kopirati vaša rješenja)*
- *Bez obzira na to da li su ispitni zadaci urađeni, svi studenti koji su pristupili ispitu moraju predati svoj rad*

1. ZADATAK

```
#include<iostream>
#include<string>
#include<vector>

/****************************************************************************
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.
****************************************************************************/

using namespace std;

enum Oblast { SoftverskiInzinjering, KomunikacijskiSistemi, SigurnostInformacijskihSistema,
InteligentniSistemi };
const char* oblast_txt[] = { "Softverski inzinjering", "Komunikacijski sistemi", "Sigurnost
informacijskih sistema", "Inteligentni sistemi" };

template<class T1, class T2, int max>
class Kolekcija {
	T1 _elementi1[max];
	T2 _elementi2[max];
	int _trenutno;
public:
	Kolekcija()
	{
		_trenutno = 0;
	}

	void AddElement(const T1& elem1, const T2& elem2)
	{
		if (_trenutno == max)
			throw exception("Dostigli ste maksimalan broj elemenata u kolekciji!");

		for (int i = 0; i < _trenutno; i++)
		{
			if (_elementi1[i] == elem1 || _elementi2[i] == elem2)
				throw exception("Nije moguce dodati duple elemente u kolekciju!");
		}

		_elementi1[_trenutno] = elem1;
		_elementi2[_trenutno++] = elem2;

	}

	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, max> & obj)
```

```
	{
		for (size_t i = 0; i < obj._trenutno; i++)
		{
			COUT << obj.getElement1(i) << " " << obj.getElement2(i) << endl;
		}
		return COUT;
	}
};

//Deklaracija klase Nastavnik omogucava njeno koristenje u klasi ZavrsniRad, a definicija je
data naknadno
class Nastavnik;

class ZavrsniRad
{
	char* _nazivTeme;
	Oblast* _oblastTeme;
	string _datumPrijave; //Format: dd.MM.gggg
	//U vector pohraniti samo adresu objekta tipa Nastavnik, dakle bez alokacije nove
memorije
	vector<Nastavnik*> _komisija;
	string _datumOdbrane; //Format: dd.MM.gggg (najranije 2 mjeseca od datuma prijave)
	int _ocjena;
public:
	ZavrsniRad() :_nazivTeme(nullptr), _oblastTeme(nullptr), _datumPrijave("NEMA
VRIJEDNOST"), _datumOdbrane("NEMA VRIJEDNOST"), _ocjena(5)
	{ }

	ZavrsniRad(string nazivTeme, Oblast oblastTeme, string datumPrijave) :
_datumPrijave(datumPrijave), _oblastTeme(new Oblast(oblastTeme))
	{
		_nazivTeme = new char[nazivTeme.size() + 1];
		strcpy_s(_nazivTeme, nazivTeme.size() + 1, nazivTeme.c_str());
	}

	~ZavrsniRad()
	{
		delete[] _nazivTeme; _nazivTeme = nullptr;
		delete _oblastTeme;	_oblastTeme = nullptr;
	}

	char* GetNazivTeme()const { return _nazivTeme; }
	Oblast GetOblastTeme()const { return *_oblastTeme; }
	string GetDatumOdbrane() { return _datumOdbrane; }
	int GetOcjena() { return _ocjena; }
	vector<Nastavnik*>& GetKomisija() { return _komisija; };

	friend ostream& operator<<(ostream& COUT, ZavrsniRad& r)
	{
		COUT << "Tema rada: " << r._nazivTeme << endl;
		COUT << "Oblast teme: " << oblast_txt[*r._oblastTeme] << endl;
		COUT << "Datum prijave rada: " << r._datumPrijave << endl;
	//Podatke o nastavnicima nije moguce ispisati ovdje jer klasa jos nije definisana
		return COUT;
	}
};

class Nastavnik
{
	string _imePrezime;
	Oblast _oblastIzboraUZvanje;
	//Parametar string predstavlja broj indeksa studenta koji prijavljuje zavrsni rad kod
odredjenog nastavnika
	Kolekcija<string, ZavrsniRad, 10> _teme;

public:
	Nastavnik(string imePrezime, Oblast oblastIzboraUZvanje) : _imePrezime(imePrezime),
_oblastIzboraUZvanje(oblastIzboraUZvanje)
	{}
```

```
	string GetImePrezime() { return _imePrezime; }
	Oblast GetOblast() { return _oblastIzboraUZvanje; }
	Kolekcija<string, ZavrsniRad, 10>& GetTeme() { return _teme; };
};
void main()
{
	const int max = 4;
	Nastavnik* nastavnici[max];

	nastavnici[0] = new Nastavnik("Denis Music", SoftverskiInzinjering);
	nastavnici[1] = new Nastavnik("Zanin Vejzovic", KomunikacijskiSistemi);
	nastavnici[2] = new Nastavnik("Jasmin Azemovic", SigurnostInformacijskihSistema);
	nastavnici[3] = new Nastavnik("Emina Junuz", SoftverskiInzinjering);

	try
	{
/*Funkcija DodajZavrsniRad ima zadatak da odredjenom nastavniku dodijeli mentorstvo na
zavrsnom radu. Sprijeciti dodavanje zavrsnih radova sa istom temom vise puta. Nastavnik moze
imati (mentorisati) samo radove iz oblasti za koju posjeduje izbor u zvanje.U slucaju da se
nastavniku pokusa dodati rad koji nije iz njegove oblasti funkcija treba da baci izuzetak sa
odgovarajucom porukom */

		//indeks, naslov, oblast, datum prijave
		nastavnici[0]->DodajZavrsniRad("IB130011", "Multimedijalni informacijski sistem
za visoko-obrazovnu ustanovu", SoftverskiInzinjering, "01.04.2017");
		nastavnici[0]->DodajZavrsniRad("IB120051", "Sistem za podršku rada kablovskog
operatera", SoftverskiInzinjering, "03.03.2017");

		nastavnici[1]->DodajZavrsniRad("IB140102", "Praktična analiza sigurnosti
bežičnih računarskih mreža", KomunikacijskiSistemi, "22.08.2017");

		nastavnici[2]->DodajZavrsniRad("IB140002", "Primjena teorije informacija u
procesu generisanja kriptografskih ključeva", SigurnostInformacijskihSistema, "10.09.2017");

		vector<Nastavnik*> komisija;//formira se komisija
		komisija.push_back(nastavnici[0]);
		komisija.push_back(nastavnici[2]);
		komisija.push_back(nastavnici[3]);

/*Funkcija ZakaziOdbranuRada ima zadatak da studentu sa proslijedjenim brojem indeksa zakaze
odbranu zavrsnog rada sto podrazumijeva definisanje datuma odbrane i liste clanova komisije
pred kojima ce student braniti zavrsni rad. Odbrana rada se moze zakazati samo studentu koji
je rad prethodno prijavio. Komisiju trebaju ciniti najmanje 2 nastavnika koji imaju izbor u
zvanje u oblasti kojoj pripada tema rada. Datum odbrane ne smije biti manji od datuma
prijave. U slucaju da bilo koji od navedenih uslova nije ispunjen funkcija treba da baci
izuzetak*/
		nastavnici[0]->ZakaziOdbranuRada("IB130011", "25.09.2017", komisija);
		nastavnici[0]->ZakaziOdbranuRada("IB130111", "25.09.2017", komisija);//student
sa brojem indeksa IB130111 jos uvijek nije prijavio rad

/*Studentu sa brojem indeksa IB130011 dodjeljuje ocjenu 8 na zavrsnom radu. Uslov za dodjelu
ocjene je da student posjeduje definisan datum odbrane i listu clanova komisije. U zavisnosti
od uspjesnosti izvrsenja, funkcija vraca true ili false*/

		if ((*nastavnici[0])("IB130011", 8))
			cout << "Uspjesno ste ocijenili zavrsni rad!" << endl;

/*Ispisuje sve podatke o nastavniku i njegovim mentorstvima. Za clanove komisije je dovoljno
ispisati samo ime i prezime.*/
		cout << *nastavnici[0] << endl;

/*Funkcija PronadjiNajStudenta ima zadatak da pronadje broj indeksa studenta koji je na
zavrsnom radu ostvario najvecu ocjenu kod nastavnika koji posjeduje najvecu prosjecnu ocjenu
na zavrsnim radovima. Ukoliko se kod nastavnika sa najvecom prosjecnom ocjenom pojavi vise
studenata sa istom ocjenom, onda funkcija vraca broj indeksa prvog pronadjenog studenta. Svim
studentima koji su odbranili rad kod nastavnika sa najvecom prosjecno ocjenom, u zasebnom
thread.u, poslati email poruku (mail adresa: brojIndeksa@edu.fit.ba) sa sadrzajem da su svoj
zavrsni rad uspjesno odbranili sa vecom ili manjom ocjenom od prosjecne. Ukoliko niti jedan
```

```
od nastavnika ne posjeduje evidentirano mentorstvo na zavrsnom radu, funkcija vraca tekst:
NIJE PRONADJEN*/

		cout << "Najsupjesniji student: " << PronadjiNajStudenta(nastavnici, max) <<
endl;

		//Baca izuzetak zbog neadekvatnog izbora u zvanje, odnosno oblasti
		nastavnici[2]->DodajZavrsniRad("IB150008", "Razvoj sistema autentifikacije na
osnovu biometrije glasa", InteligentniSistemi, "15.05.2017");
	}
	catch (exception& ex)
	{
		cout << "GRESKA -> " << ex.what() << endl;
	}

	for (int i = 0; i < max; i++)
	{
		delete nastavnici[i];
		nastavnici[i] = nullptr;
	}
}
```