- *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

a) Pojasniti razliku između virtualnih klasa i virtualnih funkcija, te koja je uloga virtualizacije u kontekstu polimorfizma
b) Pojasniti prednosti i nedostatke višestrukog nasljeđivanja

2. ZADATAK

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

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.
6. BEZ OBZIRA NA TO DA LI SU ISPITNI ZADACI URAĐENI, SVI STUDENTI KOJI SU PRISTUPILI ISPITU
MORAJU PREDATI SVOJ RAD
7. ZA POTREBE TESTIRANJA, UNUTAR MAIN FUNKCIJE MOZETE DODAVATI NOVE TESTNE PODATKE
****************************************************************************/
//narednu liniju code-a ignorisite, osim u slucaju da vam bude predstavljala smetnje u radu
#pragma warning(disable:4996)

char *crt = "\n-------------------------------------------\n";
enum Predmet { Matematika, Biologija, Hemija, Fizika };

class Datum {
	int *_dan, *_mjesec, *_godina;
public:
	Datum(int dan = 1, int mjesec = 1, int godina = 2000) {
		_dan = new int(dan);
		_mjesec = new int(mjesec);
		_godina = new int(godina);
	}
	~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;
		return COUT;
	}
};

template<class T1, class T2, int max>
class FITKolekcija {
	T1 * _elementi1[max];
	T2 * _elementi2[max];
	int _trenutno;
```

```
public:
	FITKolekcija() {
		for (size_t i = 0; i < max; i++) {
			_elementi1[i] = nullptr;
			_elementi2[i] = nullptr;
		}
		_trenutno = 0;
	}
	~FITKolekcija() {
		for (size_t i = 0; i < max; i++) {
			delete _elementi1[i]; _elementi1[i] = nullptr;
			delete _elementi2[i]; _elementi2[i] = nullptr;
		}
	}
	T1 ** GetT1() { return _elementi1; }
	T2 ** GetT2() { 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, FITKolekcija &obj) {
		for (size_t i = 0; i < obj._trenutno; i++)
			COUT << *obj._elementi1[i] << " " << *obj._elementi2[i] << endl;
		return COUT;
	}

	bool AddElement(T1& el1, T2& el2) {
		if (_trenutno >= max)
			return false;
		_elementi1[_trenutno] = new T1(el1);
		_elementi2[_trenutno++] = new T2(el2);
		return true;
	}

};

class Ocjena {
	int _ocjena;//1-5
	Predmet _predmet;
	Datum _datum;
	bool _ostavljenDatum;//u slucaju da ucenik nije bio spreman odgovarati onda se
evidentira samo datum, a provjera znanja ostavlja za naredni put
public:
	Ocjena(Predmet predmet, Datum datum, bool zaNaredniPut, int ocjena = 0) :_datum(datum)
{
		_predmet = predmet;
		_ostavljenDatum = zaNaredniPut;
		_ocjena = ocjena;
	}
	void PostaviOcjenu(int ocjena) {
		if (ocjena > 0 && ocjena <= 5) {
			_ocjena = ocjena;
			_ostavljenDatum = false;
		}
	}
	int GetOcjena() { return _ocjena; }
	friend ostream & operator<<(ostream &COUT, Ocjena & obj) { COUT << obj._predmet << " "
<< obj._ocjena << " " << obj._ostavljenDatum << " " << obj._datum; }
};

class Ucenik {
	char _brojUDnevniku[15];
	char * _imePrezime;
	vector<Ocjena *> _ocjene;
public:
	Ucenik(char* imePrezime, char brojUDnevniku[]) {
		strncpy_s(_brojUDnevniku, 15, brojUDnevniku, _TRUNCATE);
		_imePrezime = new char[strlen(imePrezime) + 1];
		strcpy_s(_imePrezime, strlen(imePrezime) + 1, imePrezime);
```

```
	}
	~Ucenik() {
		delete[] _imePrezime; _imePrezime = nullptr;
	}
	char * GetImePrezime() { return _imePrezime; }
	char * GetBrojUDnevniku() { return _brojUDnevniku; }
	vector<Ocjena*> & GetOcjene() { return _ocjene; }

	friend ostream & operator<<(ostream &COUT, Ucenik & obj) {
		COUT << crt << obj._brojUDnevniku << " " << obj._imePrezime << crt;
		for (size_t i = 0; i < obj._ocjene.size(); i++)
			COUT << *obj._ocjene[i];
		COUT << crt;
	}
};
class Odjeljenje {
	char _oznaka[15];//primjer oznaka: a, b, I, II, Matematicari i sl.
				//koristeci vrijednost tipa bool oznacava da li ce odredjeni
ucenik ponavljati razred
	FITKolekcija<Ucenik *, bool, 30> _ucenici;//maksimalno 30 ucenika u odjeljenju
public:
	Odjeljenje(char oznaka[]) {
		strncpy_s(_oznaka, 15, oznaka, _TRUNCATE);
	}
	char * GetOznaka() { return _oznaka; }
	FITKolekcija<Ucenik *, bool, 30> & GetUcenici() { return _ucenici; }
	friend ostream & operator<<(ostream &COUT, Odjeljenje & obj) {
		cout << obj._oznaka << " " << obj._ucenici << endl;
	}
};

class Razred {
	int _oznakaRazreda; // 1, 2, 3...
	Odjeljenje * _odjeljenja;
	int _trenutnoOdjeljenja;
public:
	Razred(int oznaka) {
		_oznakaRazreda = oznaka;
		_trenutnoOdjeljenja = 0;
		_odjeljenja = nullptr;
	}
	~Razred() {
		delete[] _odjeljenja; _odjeljenja = nullptr;
	}
	friend ostream & operator<<(ostream &COUT, Razred & obj) {
		cout << "Razred -> " << obj._oznakaRazreda << endl;
		for (size_t i = 0; i < obj._trenutnoOdjeljenja; i++)
			COUT << obj._odjeljenja[i];
	}
	int GetTrenutnoOdjeljenja() { return _trenutnoOdjeljenja; }
	Odjeljenje * GetOdjeljenja() { return _odjeljenja; }
};

void main() {
	Datum datum15092017(15, 9, 2017), datum02102017(2, 10, 2017), datum03102017(3, 10,
2017);

	Ocjena matematika_15092017_3(Matematika, datum15092017, false, 3),
		hemija_02102017_4(Hemija, datum02102017, true);//ostavljen datum

	Razred II(2);
	/*prilikom dodavanja ucenika broj u dnevniku se autmatski dodjeljuje na nacin da se
prvom uceniku u odjeljenju dodijeli oznaka u formatu
oznakaRazreda_oznakaOdjeljenja_redniBroj(npr. 1_a_1, 1_II_1, a svaki naredni ucenik treba
imati broj za 1 veci od prethodnog ucenika; 1_a_2, 1_II_2).
	ovu funkciju iskorisiti prilikom dodavanja novog ucenika, na nacin da za proslijedjenu
oznaku odjeljenja generise i vrati naredni redni broj u dnevniku. u slucaju da odredjeno
odjeljenje ne postoji funkcija vraca nullptr
	*/
```

```
	char * brojUDnevniku = II.GenerisiNaredniBrojUDnevniku("a");//a - oznaka odjeljenja
	if (brojUDnevniku != nullptr)
		cout << brojUDnevniku << endl;
	/*
	funkcija DodajUcenika ima zadatak da doda podatke o uceniku u okviru odredjenog odjeljenja (koji je poslan kao prvi parametar). u slucaju da dato odjeljenje ne postoji, funkcija najprije dodaje odjeljenje, te ucenika dodjeljuje tom odjeljenju.
	ucenik nece biti dodan u sluacaju da u razredu vec postoji ucenik sa identicnim imenom i prezimenom ili su sva mjesta u razredu popunjena.
	*/
	//odjeljenje, ime i prezime
	if (II.DodajUcenika("a", "Jasmin Azemovic"))
		cout << "UCENIK DODAN" << endl;
	if (II.DodajUcenika("a", "Adel Handzic"))
		cout << "UCENIK DODAN" << endl;
	if (II.DodajUcenika("a", "Emina Junuz"))
		cout << "UCENIK DODAN" << endl;
	if (II.DodajUcenika("b", "Zanin Vejzovic"))
		cout << "UCENIK DODAN" << endl;

	/*
	ucenik moze imati vise ocjena iz odredjenog predmeta.
	u slucaju da je iz datog predmeta uceniku ranije ostavljen datum, onda se vrsi samo modifikacija ocjene, a raniji datum i druge vrijednosti atributa ostaju nepromijenjene
	ucenik iz odredjenog predmeta moze imati najvise dva ostavljena datuma, te se u tom slucaju prva naredna ocjena (bila ona pozitivna ili ne) dodaje na prvi, a druga na drugi ostavljeni datum
	*/
	if (II.DodajOcjenu("Adel Handzic", matematika_15092017_3))
		cout << "OCJENA DODANA" << endl;

	if (II.DodajOcjenu("Adel Handzic", hemija_02102017_4))
		cout << "OCJENA DODANA" << endl;

	hemija_02102017_4.PostaviOcjenu(4);
	if (II.DodajOcjenu("Adel Handzic", hemija_02102017_4))
		cout << "OCJENA DODANA" << endl;

	/*
	funkcija (radi se o jednoj funkciji) ima zadatak da prikazuje uspjeh odredjenog (ukoliko se proslijedi ime i prezime) ili svih ucenika u razredu (ukoliko se ne proslijedi nikakva vrijednost) na nacin da pored svakog predmeta prikaze i zakljucnu ocjenu. ukoliko iz odredjenog predmeta ucenik ima dvije negativne ocjene (1), onda ce i uspjeh biti negativan (1). na kraju ispisa, pored zakljucnih ocjena iz pojedinih predmeta, prikazati i opsti uspjeh koji predstavlja prosjecnu ocjenu (ukoliko ucenik ima negativan uspjeh na nekom od predmeta, onda ce i opsti uspjeh biti negativan).
	*/
	II("Adel Handzic");
	II();

	//funkcija ima zadatak da oznaci (promijeni vrijednost atributa tipa bool) sve ucenike (i vrati njihov broj) koji ce ponavljati razred, a vrijednost parametra predstavlja broj predmeta za iz koliko ucenik treba ostvariti negativan uspjeh da bi ponavljao razred
	cout<<"Broj ponavljaca -> "<< II.OznaciPonavljace(2);

	/*u zasebnom thread.u poslati email poruke (mail adrese: brojUDnevniku@edu.fit.ba) svim ucenicima koji su ostvarili prosjecan uspjeh isti ili veci od onog koji je definisan parametrom npr. 4.2*/
	PosaljiPoruke(II, 4.2);
}
```