1. ZADATAK

Izvršiti definiciju funkcija na način koji odgovara opisu(komentarima) datom neposredno uz pozive ili nazive funkcija.Možete dati komentar na bilo koju liniju code - a koju smatrate da bi trebalo unaprijediti ili da će eventualno uzrokovati grešku prilikom kompajliranja.Također, možete dodati dodatne funkcije koje će vam olakšati implementaciju programa.

```
#include <iostream>

using namespace std;


const char* PORUKA = "\n--------------------------------------------------------------------------------\n"

"0. PROVJERITE DA LI PREUZETI ZADACI PRIPADAJU VASOJ GRUPI (G1/G2)\n"

"1. SVE KLASE TREBAJU POSJEDOVATI ADEKVATAN DESTRUKTOR\n"

"2. NAMJERNO IZOSTAVLJANJE KOMPLETNIH I/ILI POJEDINIH DIJELOVA DESTRUKTORA CE BITI OZNACENO KAO TM\n"

"3. SPASAVAJTE PROJEKAT KAKO BI SE SPRIJECILO GUBLJENJE URADJENOG ZADATKA\n"

"4. ATRIBUTI, NAZIVI FUNKCIJA, TE BROJ I TIP PARAMETARA MORAJU BITI IDENTICNI ONIMA KOJI SU KORISTENI U TESTNOM CODE-U, "

"OSIM U SLUCAJU DA POSTOJI ADEKVATAN RAZLOG ZA NJIHOVU MODIFIKACIJU. OSTALE "

"POMOCNE FUNKCIJE MOZETE IMENOVATI I DODAVATI PO ZELJI.\n"

"5. IZUZETAK BACITE SAMO U FUNKCIJAMA U KOJIMA JE TO NAZNACENO.\n"

"6. FUNKCIJE KOJE NE IMPLEMENTIRATE TREBAJU BITI OBRISANE (KAKO POZIV TAKO I DEFINICIJA)!\n"

"7. NA KRAJU ISPITA SVOJE RJESENJE KOPIRATE U .DOCX FAJL (IMENOVAN BROJEM INDEKSA)!\n"

"8. RJESENJA ZADATKA POSTAVITE NA FTP SERVER U ODGOVARAJUCI FOLDER!\n"

"9. NEMOJTE POSTAVLJATI VISUAL STUDIO PROJEKTE, VEC SAMO .DOCX FAJL SA VASIM RJESENJEM!\n"

"--------------------------------------------------------------------------------\n";



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

enum eRazred { PRVI = 1, DRUGI, TRECI, CETVRTI };
```

```
char* GetNizKaraktera(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;
   bool _omoguciDupliranje;
public:
   Kolekcija(bool omoguciDupliranje = true) {
      _elementi1 = nullptr;
      _elementi2 = nullptr;
      _omoguciDupliranje = omoguciDupliranje;
   }
   ~Kolekcija() {
      delete[]_elementi1; _elementi1 = nullptr;
      delete[]_elementi2; _elementi2 = nullptr;
   }
   T1& getElement1(int lokacija)const { return _elementi1[lokacija]; }
   T2& getElement2(int lokacija)const { return _elementi2[lokacija]; }
   int getTrenutno() { return _trenutno; }
   friend ostream& operator<< (ostream& COUT, const Kolekcija& obj) {
```

```
        for (size_t i = 0; i < obj._trenutno; i++)
            COUT << obj.getElement1(i) << " " << obj.getElement2(i) << endl;
        return COUT;
    }
};
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, const Datum& obj) {
        COUT << *obj._dan << "." << *obj._mjesec << "." << *obj._godina;
        return COUT;
    }
};

class Predmet {
    char* _naziv;
    Kolekcija<Datum*, int>* _ocjene;
public:
    Predmet(const char* naziv = "", Datum datum = Datum(), int ocjena = 0) {
```

```
        _naziv = GetNizKaraktera(naziv);
        _ocjene = nullptr;
        if (ocjena > 0)
            AddOcjena(datum, ocjena);
    }
    ~Predmet() {
        delete[] _naziv; _naziv = nullptr;
    }
    char* GetNaziv() { return _naziv; }
    Kolekcija<Datum*, int> GetOcjene() { return _ocjene; }
};
class Uspjeh {
    eRazred _razred;
    //char* se odnosi na napomenu o polozenom predmetu
    Kolekcija<Predmet*, const char*> _polozeniPredmeti;
public:
    Uspjeh(eRazred razred = PRVI) {
        _razred = razred;
    }
    Kolekcija<Predmet*, const char*> GetPredmeti() { return _polozeniPredmeti; }
    eRazred GetERazred() { return _razred; }
    friend ostream& operator<< (ostream& COUT, const Uspjeh& obj) {
        COUT << obj._razred << " " << obj._polozeniPredmeti << endl;
        return COUT;
    }
};

class Kandidat {
    char* _imePrezime;
```

```
    string _emailAdresa;
    string _brojTelefona;
    vector<Uspjeh> _uspjeh;
public:
    Kandidat(const char* imePrezime, string emailAdresa, string brojTelefona) {
        _imePrezime = GetNizKaraktera(imePrezime);
        _emailAdresa = emailAdresa;
        _brojTelefona = brojTelefona;//izvrsiti validaciju broja telefona
    }
    ~Kandidat() {
        delete[] _imePrezime; _imePrezime = nullptr;
    }
    friend ostream& operator<< (ostream& COUT, Kandidat& obj) {
        COUT << obj._imePrezime << " " << obj._emailAdresa << " " << obj._brojTelefona << endl;
        for (size_t i = 0; i < obj._uspjeh.size(); i++)
            COUT << obj._uspjeh[i];
        return COUT;
    }
    vector<Uspjeh>& GetUspjeh() { return _uspjeh; }
    string GetEmail() { return _emailAdresa; }
    string GetBrojTelefona() { return _brojTelefona; }
    char* GetImePrezime() { return _imePrezime; }
};

const char* GetOdgovorNaPrvoPitanje() {
    cout << "Pitanje -> Pojasnite ulogu i nacin koristenja generickog tipa future<>?\n";
    return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";
}
const char* GetOdgovorNaDrugoPitanje() {
```

```
   cout << "Pitanje -> Ukratko opisite na koji nacin se moze izbjeci pojavljivanje vise podobjekata bazne klase u slucaju visestrukog nasljedjivanja?\n";
   return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";
}

void main() {

   cout << PORUKA;
   cin.get();

   cout << GetOdgovorNaPrvoPitanje() << endl;
   cin.get();
   cout << GetOdgovorNaDrugoPitanje() << endl;
   cin.get();

   Datum
      datum19062021(19, 6, 2021),
      datum20062021(20, 6, 2021),
      datum30062021(30, 6, 2021),
      datum05072021(5, 7, 2021);

   int kolekcijaTestSize = 9;
   Kolekcija<int, int> kolekcija1(false);
   for (int i = 0; i <= kolekcijaTestSize; i++)
      kolekcija1.AddElement(i, i);

   try {
      //ukoliko nije dozvoljeno dupliranje elemenata (provjeravaju se T1 i T2), metoda AddElement baca izuzetak
```

```
    kolekcija1.AddElement(3, 3);
  }
  catch (exception& err) {
    cout << err.what() << crt;
  }
  cout << kolekcija1 << crt;

  /*objekat kolekcija2 ce biti inicijalizovan elementima koji se u objektu kolekcija1 nalaze na lokacijama
1 - 4
  ukljucujuci i te lokacije. u konkretnom primjeru to ce biti parovi sa vrijednostima: 1 1 2 2 3 3 4 4*/
  Kolekcija<int, int> kolekcija2 = kolekcija1(1, 4);
  cout << kolekcija2 << crt;
  try {
    //primjeri u kojima opseg nije validan, te bi funkcija trebala baciti izuzetak
    Kolekcija<int, int> temp1 = kolekcija1(1, 14);//imamo 10 elemenata
    Kolekcija<int, int> temp2 = kolekcija1(-1, 8);//lokacija -1 ne postoji
  }
  catch (exception& err) {
    cout << err.what() << crt;
  }

  //parametri: nazivPredmeta, datum, prva ocjena
  Predmet Matematika("Matematika", datum19062021, 5),
    Fizika("Fizika", datum20062021, 5),
    Hemija("Hemija", datum30062021, 2),
    Engleski("Engleski", datum05072021, 5);

  Matematika.AddOcjena(datum05072021, 3);
  Matematika.AddOcjena(datum05072021, 3);
```

```
// ispisuje: naziv predmeta, ocjene (zajedno sa datumom polaganja) i prosjecnu ocjenu na predmetu
// ukoliko predmet nema niti jednu ocjenu prosjecna treba biti jednaka 0
cout << Matematika << endl;



/*
broj telefona mora biti u formatu
- znak +
- pozivni broj drzave (2 ili 3 cifre)
- pozivni broj operatera (2 cifre) npr. 063, 061, 065 pri cemu je broj 0 opcionalan
- prvi dio broja (2 ili 3 cifre)
- razmak ili crtica, oboje opcionalno tj. broj telefona ne mora sadrzavati niti jedno niti drugo
 - drugi dio broja (2 ili 3 cifre)
ukoliko broj telefona nije u validnom formatu, njegova vrijednost se postavlja na NOT VALID
*/

if (ValidirajBrojTelefona("+38761222333"))
   cout << "Broj telefona validan" << crt;
if (ValidirajBrojTelefona("+38761222-333"))
   cout << "Broj telefona validan" << crt;
if (ValidirajBrojTelefona("+38761222 333"))
   cout << "Broj telefona validan" << crt;
if (ValidirajBrojTelefona("+387061222 333"))
   cout << "Broj telefona validan" << crt;
if (!ValidirajBrojTelefona("+38761 222 333"))
   cout << "Broj NIJE telefona validan" << crt;
if (!ValidirajBrojTelefona("+387 61222 333"))
   cout << "Broj NIJE telefona validan" << crt;
```

```
Kandidat jasmin("Jasmin Azemovic", "jasmin@fit.ba", "+38761222333");
Kandidat adel("Adel Handzic", "adel@edu.fit.ba", "+387061222 333");
Kandidat brojTelefonaNotValid("Ime Prezime", "korisnik@klix.ba", "+387 61222 333");

/*
uspjeh (tokom srednjoskolskog obrazovanja) se dodaje za svaki predmet na nivou razreda.
tom prilikom onemoguciti:
- dodavanje istih (moraju biti identicne vrijednosti svih clanova tipa Predmet) predmeta na nivou jednog razreda,
- dodavanje predmeta kod kojih je prosjecna ocjena manja od 2.5
- dodavanje vise od 5 predmeta na nivou jednog razreda
razredi (predmeti ili uspjeh) ne moraju biti dodavani sortiranim redoslijedom (npr. prvo se moze dodati uspjeh za II razred, pa onda za I razred i sl.).
Metoda vraca true ili false u zavisnosti od (ne)uspjesnost izvrsenja
*/
if (jasmin.AddPredmet(DRUGI, Fizika, "Napomena 1"))
   cout << "Predmet uspjesno dodan!" << crt;
if (jasmin.AddPredmet(DRUGI, Hemija, "Napomena 2"))
   cout << "Predmet uspjesno dodan!" << crt;
if (jasmin.AddPredmet(PRVI, Engleski, "Napomena 3"))
   cout << "Predmet uspjesno dodan!" << crt;
if (jasmin.AddPredmet(PRVI, Matematika, "Napomena 4"))
   cout << "Predmet uspjesno dodan!" << crt;
//Matematiku je vec dodana u prvom razredu
if (jasmin.AddPredmet(PRVI, Matematika, "Napomena 5"))
   cout << "Predmet uspjesno dodan!" << crt;
```

/*nakon evidentiranja uspjeha na bilo kojem predmetu kandidatu se salje email sa porukom:

FROM:info@fit.ba

TO: emailKorisnika

Postovani ime i prezime, evidentirali ste uspjeh za X razred. Dosadasnji uspjeh (prosjek)

na nivou X razreda iznosi Y, a ukupni uspjeh u toku skolovanja iznosi Z.

Pozdrav.

FIT Team.

ukoliko je prosjek na nivou tog razreda veci od 4.5 kandidatu se salje SMS sa porukom: "Svaka cast za uspjeh 4.X u X razredu". Slanje SMS-a izvrsiti samo u slucaju da je broj telefona validna, u protivnom ispisati poruku „BROJ TELEFONA NIJE VALIDAN“

slanje poruka i emailova implemenitrati koristeci zasebne thread-ove.

*/

cout << jasmin << crt;

//vraca kolekciju predmeta koji sadrze najmanje jednu ocjenu evidentiranu u periodu izmedju proslijedjenih datuma

//float se odnosi na prosjecan broj dana izmedju ostvarenih ocjena na predmetu

Kolekcija<Predmet, float> jasminUspjeh = jasmin(new Datum(18, 06, 2021), new Datum(21, 06, 2021));

cout << jasminUspjeh << crt;

cin.get();

system("pause>0");

}