```
using namespace std;

const char *crt = "\n-------------------------------------------\n";
enum GodinaStudija { PRVA = 1, DRUGA, TRECA};

char * Alociraj(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 Dictionary {
    T1 * _elementi1;
    T2 * _elementi2;
    int * _trenutno;
public:
    Dictionary() {
        _elementi1 = nullptr;
        _elementi2 = nullptr;
        _trenutno = new int(0);
    }
    ~Dictionary() {
        delete[]_elementi1; _elementi1 = nullptr;
        delete[]_elementi2; _elementi2 = nullptr;
        delete _trenutno; _trenutno = 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 Dictionary &obj) {
        for (size_t i = 0; i < *obj._trenutno; i++)
            COUT << obj.getElement1(i) << " " << obj.getElement2(i) << endl;
        return COUT;
    }
};
class DatumVrijeme {
    int *_dan, *_mjesec, *_godina, *_sati, *_minuti;
public:
    DatumVrijeme(int dan = 1, int mjesec = 1, int godina = 2000, int sati =
0, int minuti = 0) {
        _dan = new int(dan);
        _mjesec = new int(mjesec);
        _godina = new int(godina);
        _sati = new int(sati);
        _minuti = new int(minuti);
    }
    ~DatumVrijeme() {
        delete _dan; _dan = nullptr;
        delete _mjesec; _mjesec = nullptr;
        delete _godina; _godina = nullptr;
        delete _sati; _sati = nullptr;
        delete _minuti; _minuti = nullptr;
    }
    friend ostream& operator<< (ostream &COUT, const DatumVrijeme &obj) {
```

```
        COUT << *obj._dan << "." << *obj._mjesec << "." << *obj._godina << "
" << *obj._sati << ":" << *obj._minuti << endl;
        return COUT;
    }
};

class Predmet {
    char * _naziv;
    int _ocjena;
    string _napomena;
public:
    Predmet(const char * naziv = "", int ocjena = 0, string napomena = "") {
        _naziv = Alociraj(naziv);
        _ocjena = ocjena;
        _napomena = napomena;
    }
    ~Predmet() {
        delete[] _naziv; _naziv = nullptr;
    }
    friend ostream& operator<< (ostream &COUT, Predmet &obj) {
        COUT << obj._naziv << " (" << obj._ocjena << ") " << obj._napomena <<
endl;
        return COUT;
    }
    string GetNapomena() { return _napomena; }
    char * GetNaziv() { return _naziv; }
    int GetOcjena() { return _ocjena; }

    void DodajNapomenu(string napomena) {
    _napomena += " " + napomena;
    }
};
class Uspjeh {
    GodinaStudija * _godina;
    //datumvrijeme se odnosi na vrijeme evidentiranja polozenog predmeta
    Dictionary<Predmet, DatumVrijeme> _predmeti;
public:
    Uspjeh(GodinaStudija godina) {
        _godina = new GodinaStudija(godina);
    }
    ~Uspjeh() { delete _godina; _godina = nullptr; }

    Dictionary<Predmet, DatumVrijeme> * GetPredmeti() { return &_predmeti; }
    GodinaStudija * GetGodinaStudija() { return _godina; }
    friend ostream& operator<< (ostream &COUT, const Uspjeh &obj) {
    COUT << *obj._godina << " " << obj._predmeti << endl;
    return COUT;
    }
};

class Student {
    char * _imePrezime;
    string _emailAdresa;
    string _brojTelefona;
    vector<Uspjeh> _uspjeh;
public:
```

```
    Student(const char * imePrezime, string emailAdresa, string brojTelefona)
{
        _imePrezime = Alociraj(imePrezime);
        _emailAdresa = emailAdresa;
        _brojTelefona = brojTelefona;
    }
    ~Student() {
        delete[] _imePrezime; _imePrezime = nullptr;
    }
    friend ostream& operator<< (ostream &COUT, Student &obj) {
        COUT << obj._imePrezime << " " << obj._emailAdresa << " " <<
obj._brojTelefona << endl;
        return COUT;
    }
    vector<Uspjeh> * GetUspjeh() { return &_uspjeh; }
    string GetEmail() { return _emailAdresa; }
    string GetBrojTelefona() { return _brojTelefona; }
    char * GetImePrezime() { return _imePrezime; }
};

void main() {

/*****************************************************************************
    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 URAĐENOG ZADATKA
    4. PROGRAMSKI CODE SE TAKOĐER NALAZI U FAJLU CODE.TXT
    5. 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.
    6. IZUZETAK BACITE U FUNKCIJAMA U KOJIMA JE TO NAZNAÈENO.
    ******************************************************************************
***/
    cout << "NA KRAJU ISPITA, RAD PREDAJTE U ODGOVARAJUCI FOLDER NA FTP
SERVERU (INTEGRALNI)!" << endl;
    DatumVrijeme temp,
    datum19062019_1015(19, 6, 2019, 10, 15),
    datum20062019_1115(20, 6, 2019, 11, 15),
    datum30062019_1215(30, 6, 2019, 12, 15),
    datum05072019_1231(5, 7, 2019, 12, 31);
    //funkcija ToCharArray vraca datum i vrijeme kao char *. konverziju
izvrsiti koristeci stringstream objekat.
    //voditi racuna o tome da se izmedju datuma i vremena nalazi samo jedan
razmak, te da su vrijednosti dana i mjeseca
    //iskazane kao dvije cifre
    cout << datum19062019_1015.ToCharArray() << endl;//treba ispisati:
19/06/2019 10:15
    temp = datum05072019_1231;
    cout << temp.ToCharArray() << endl;//treba ispisati: 05/07/2019 12:31

    const int DictionaryTestSize = 9;
    Dictionary<int, int> Dictionary1;
    for (size_t i = 0; i < DictionaryTestSize; i++)
    Dictionary1.AddElement(i + 1, i*i);
```

```
    try {
        //vraca elemente kolekcije koji se nalaze na lokacijama definisanim
vrijednostima parametara (npr. 2 - 7).
        //funkcija baca izuzetak u slucaju da se zahtijeva lokacija koja ne
postoji ili je vrijednost posljednje lokacije manja od pocetne
        Dictionary<int, int> opseg = Dictionary1.getRange(2, 7);
        cout << opseg << endl;
        Dictionary1.getRange(7, 11);
    }
    catch (exception& err) {
        cout << err.what() << endl;
    }
    cout << Dictionary1 << endl;

    Dictionary<int, int> Dictionary2 = Dictionary1;
    cout << Dictionary2 << crt;

    Dictionary<int, int> Dictionary3;
    Dictionary3 = Dictionary1;
    cout << Dictionary3 << crt;

    //napomena se moze dodati i prilikom kreiranja objekta
    Predmet MAT("Matematika", 7, "Ucesce na takmicenju"),
    UIT("Uvod u informacijske tehnologije", 9),
    RM("Racunarske mreze", 8),
    EN("Engleski jezik", 6);
    UIT.DodajNapomenu("Pohvala za ostvareni uspjeh");
    cout << MAT << endl;

    /*
    email adresa mora biti u formatu: text-text@ nakon cega slijedi neka od
sljedecih domena: hotmail.com ili
    //outlook.com ili fit.ba. Pod text se podrazumijeva bilo koje slovo, malo
ili veliko.
    u slucaju da email adresa nije validna, postaviti je na defaultnu:
notSet@fit.ba
    za provjeru koristiti regex
    */
    Student jasmin("Jasmin Azemovic", "jasmin.azemovic@hotmail.com", "033 281
172");
    Student adel("Adel Handzic", "adel.handzic@fit.ba", "033 281 170");
    Student emailNotValid("Ime Prezime", "korisnik@lazna.ba", "033 281 170");

    /*
    uspjeh se dodaje za svaki predmet na nivou godine studija.
    tom prilikom onemoguciti:
    - dodavanje istoimenih predmeta na nivou jedne godine,
    - dodavanje vise predmeta u kratkom vremenskom periodu (na nivou jedne
godine, razmak izmedju dodavanja pojedinih predmeta mora biti najmanje 1
sat).
    godine (predmeti ili uspjeh) ne moraju biti dodavani sortiranim
redoslijedom (npr. prvo se moze dodati uspjeh za drugu godinu, pa onda za
prvu godinu i sl.).
    Funkcija vraca true ili false u zavisnosti od (ne)uspjesnost izvrsenja
    */
    if (jasmin.AddPredmet(UIT, DRUGA, datum20062019_1115))
    cout << "Predmet uspjesno dodan!" << crt;
```

```
    if (jasmin.AddPredmet(RM, DRUGA, datum30062019_1215))
    cout << "Predmet uspjesno dodan!" << crt;
    if (jasmin.AddPredmet(EN, PRVA, datum19062019_1015))
    cout << "Predmet uspjesno dodan!" << crt;
    if (jasmin.AddPredmet(MAT, PRVA, datum20062019_1115))
    cout << "Predmet uspjesno dodan!" << crt;
    //ne treba dodati MAT jer je vec dodana u prvoj godini
    if (jasmin.AddPredmet(MAT, PRVA, datum05072019_1231))
    cout << "Predmet uspjesno dodan!" << crt;
    //ne treba dodati UIT jer nije prosao 1 sat od dodavanja posljednjeg
predmeta
    if (jasmin.AddPredmet(UIT, PRVA, datum20062019_1115))
    cout << "Predmet uspjesno dodan!" << crt;
    /*nakon evidentiranja uspjeha na bilo kojem predmetu tj. prilikom
uspjesno izvrsene funkcije AddPredmet, Studentu se salje email sa sadrzajem:
    FROM:info@fit.ba
    TO: emailStudenta
    Postovani ime i prezime, evidentirali ste uspjeh za X godinu studija.
    Pozdrav.
    FIT Team.
    ukoliko je, nakon dodavanja predmeta, prosjek na nivou te godine veci od
8.0 Studentu se, pored email-a, salje i SMS sa sadrzajem: "Svaka cast za
uspjeh X.X       ostvaren u X godini studija".
    slanje poruka i emailova implemenitrati koristeci zasebne thread-ove.
    */
    cout << "USPJEH ISPISATI KORISTEÆI OSTREAM_ITERATOR" << endl;
    cout << jasmin << endl;
    //vraca broj ponavljanja odredjene rijeci u napomenama, koristiti
sregex_iterator
    cout << "Rijec takmicenje se pojavljuje " <<
jasmin.BrojPonavljanjaRijeci("takmicenju") << " puta." << endl;

    //vraca niz predmeta koji su evidentiranih u periodu izmedju vrijednosti
proslijedjenih parametara
    vector<Predmet> jasminUspjeh = jasmin(new DatumVrijeme(18, 06, 2019, 10,
15), new DatumVrijeme(21, 06, 2019, 10, 10));
    for (Predmet u : jasminUspjeh)
    cout << u << endl;

    Uspjeh * uspjeh_I_godina = jasmin["PRVA"];//vraca uspjeh Studenta
ostvaren u prvoj godini studija
    if (uspjeh_I_godina != nullptr)
    cout << *uspjeh_I_godina << endl;

    cin.get();
    system("pause>0");
}
```