```cpp
#include <iostream>

using namespace std;

const char* crt = "\n-------------------------------------------\n";
enum Pojas { BIJELI, ZUTI, NARANDZASTI, ZELENI, PLAVI, SMEDJI, CRNI };
enum Dupliranje { BEZ_DUPLIKATA, SA_DUPLIKATIMA };

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, int max = 15>
class Kolekcija {
    T1 _elementi1[max];
    T2 _elementi2[max];
    int* _trenutno;
    Dupliranje _dupliranje;
public:
    Kolekcija(Dupliranje dupliranje = SA_DUPLIKATIMA) {
        _trenutno = new int(0);
        _dupliranje = dupliranje;
    }
    ~Kolekcija() {
        delete _trenutno; _trenutno = nullptr;
    }

    T1 getElement1(int lokacija)const { return _elementi1[lokacija]; }
    T2 getElement2(int lokacija)const { return _elementi2[lokacija]; }
    int getTrenutno()const { 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;
    }
```

```
    Datum(const Datum& obj) {
        _dan = new int(*obj._dan);
        _mjesec = new int(*obj._mjesec);
        _godina = new int(*obj._godina);
    }
};
class Tehnika {
    char* _naziv;
    //int se odnosi na ocjenu u opsegu od 1 - 5, a datum na momenat
postizanja ocjene
    Kolekcija<Datum, int>* _ocjene;
public:
    Tehnika(const char* naziv = "", Datum datum = Datum(), int ocjena = 0) {
        _naziv = GetNizKaraktera(naziv);
        _ocjene = nullptr;
        if (ocjena > 0)
            AddOcjena(&datum, ocjena);
    }
    ~Tehnika() {
        delete[] _naziv; _naziv = nullptr;
        delete _ocjene; _ocjene = nullptr;
    }
    char* GetNaziv()const { return _naziv; }
    Kolekcija<Datum, int> GetOcjene() const { return *_ocjene; }
};
class Polaganje {
    Pojas _pojas;
    //string se odnosi na napomenu o polozenoj tehnici
    Kolekcija<Tehnika*, string> _polozeneTehnike;
public:
    Polaganje(Pojas pojas = BIJELI) {
        _pojas = pojas;
    }
    Kolekcija<Tehnika*, string>& GetTehnike() { return _polozeneTehnike; }
    Pojas GetPojas() { return _pojas; }
    friend ostream& operator<< (ostream& COUT, const Polaganje& obj) {
        COUT << obj._pojas << " " << obj._polozeneTehnike << endl;
        return COUT;
    }
};
class KaratePolaznik {
    char* _imePrezime;
    string _emailAdresa;
    string _brojTelefona;
    vector<Polaganje> _polozeniPojasevi;
public:
    KaratePolaznik(const char* imePrezime, string emailAdresa, string
brojTelefona) {
        _imePrezime = GetNizKaraktera(imePrezime);
        _emailAdresa = emailAdresa;
        _brojTelefona = brojTelefona;
    }
    ~KaratePolaznik() {
        delete[] _imePrezime; _imePrezime = nullptr;
    }
    friend ostream& operator<< (ostream& COUT, KaratePolaznik& obj) {
```

```
        COUT << obj._imePrezime << " " << obj._emailAdresa << " " <<
obj._brojTelefona << endl;
        for (size_t i = 0; i < obj._polozeniPojasevi.size(); i++)
            COUT << obj._polozeniPojasevi[i];
        return COUT;
    }
    vector<Polaganje>& GetPolozeniPojasevi() { return _polozeniPojasevi; }
    string GetEmail() { return _emailAdresa; }
    string GetBrojTelefona() { return _brojTelefona; }
    char* GetImePrezime() { return _imePrezime; }
};
const char* GetOdgovorNaPrvoPitanje() {
    cout << "Pitanje -> Za sta se koriste modovi ios::ate i ios::trunc ?\n";
    return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";
}
const char* GetOdgovorNaDrugoPitanje() {
    cout << "Pitanje -> Pojasniti ulogu i način koristenja iteratora?\n";
    return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";
}
void main() {

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

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

    int kolekcijaTestSize = 9;
    Kolekcija<int, int> kolekcija1(BEZ_DUPLIKATA);
    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
            //takodjer, izuzetak se baca i u slucaju da se prekoraci
maksimalan broj elemenata
            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;
    }
    //svaka tehnika moze imati vise ocjena i polaze se u vise navrata (istog
ili drugog dana)
    //parametri: nazivTehnike, prva ocjena, datum polaganja
    Tehnika choku_zuki("choku_zuki", datum19062021, 5),
        gyaku_zuki("gyaku_zuki", datum20062021, 5),
        kizami_zuki("kizami_zuki", datum30062021, 2),
        oi_zuki("oi_zuki", datum05072021, 5);

    choku_zuki.AddOcjena(&datum05072021, 3);
    choku_zuki.AddOcjena(&datum05072021, 5);

    // ispisuje: naziv tehnike, ocjene (zajedno sa datumom polaganja) i
prosjecnu ocjenu za tu tehniku
    // ukoliko tehnika nema niti jednu ocjenu prosjecna treba biti 0
    cout << choku_zuki << endl;

    /*
    email adresa treba biti u sljedecem formatu 3_ime.prezime@karate.ba tj.
zadovoljavati sljedeća
pravila:
    - poceti sa jednim brojem nakon cega slijedi donja crtica
    - u imenu posjedovati najmanje 3 karaktera
    - izmedju imena i prezimena moze biti tacka ili donja crtica ili nista od
navedenog
    - u prezimenu posjedovati najmanje 3 karaktera
    - znak @
    - domenu karate.ba ili edu.karate.ba. Pored drzavne(.ba), dozvoljene su
oznake .com i .org.
    za provjeru validnosti email adrese koristiti globalnu funkciju
ValidirajEmail, a unutar nje regex
metode.
    validacija email adrese ce se vrsiti unutar konstruktora klase
KaratePolaznik, a u slucaju da nije validna
    postaviti je na defaultnu adresu: notSet@edu.karate.ba
    */

    if (ValidirajEmail("2_ime.prezime@edu.karate.ba"))
        cout << "Email validan" << crt;
    if (ValidirajEmail("3_ime_prezime@karate.ba"))
        cout << "Email validan" << crt;
    if (ValidirajEmail("4_imeprezime@karate.com"))
        cout << "Email validan" << crt;
    if (ValidirajEmail("8_imeprezime@karate.org"))
        cout << "Email validan" << crt;
    if (!ValidirajEmail("2ime.prezime@edu.karate.org"))
        cout << "Email NIJE validan" << crt;
    if (!ValidirajEmail("5_ime prezime@edu.karate.org"))
```

```
        cout << "Email NIJE validan" << crt;


    KaratePolaznik* jasmin = new KaratePolaznik("Jasmin Azemovic",
"1_jasmin.azemovic@karate.ba",
        "033 281 172");
    KaratePolaznik* adel = new KaratePolaznik("Adel Handzic",
"2_adel_handzic@edu.karate.ba", "033 281 170");
    KaratePolaznik * emailNotValid = new KaratePolaznik("Ime Prezime",
"korisnik@karate.ru", "033 281 170");

    /*
    svi kandidati podrazumijevano imaju BIJELI pojas (za njega se ne dodaju
tehnike)
    uspjeh se dodaje za svaku tehniku na nivou pojasa (ZUTI, ZELENI ... ).
    tom prilikom onemoguciti:
    - dodavanje istih (moraju biti identicne vrijednosti svih atributa)
tehnika na nivou jednog pojasa,
    - dodavanje tehnika za vise pojaseve ako ne postoji najmanje jedna
tehnika za nizi pojas (ne mozemo
dodati tehniku za NARANDZASTI ako ne postoji niti jedna tehnika za ZUTI
pojas)
    - dodavanje tehnika kod kojih je prosjecna ocjena manja od 3.5
    funkcija vraca true ili false u zavisnosti od (ne)uspjesnost izvrsenja
    */

    //ne treba dodati kizami_zuki jer ne postoji niti jedna tehnika za ZUTI
pojas
    if (jasmin->AddTehniku(NARANDZASTI, kizami_zuki, "Napomena 0"))
        cout << "Tehnika uspjesno dodan!" << crt;
    if (jasmin->AddTehniku(ZUTI, gyaku_zuki, "Napomena 1"))
        cout << "Tehnika uspjesno dodan!" << crt;
    if (jasmin->AddTehniku(ZUTI, kizami_zuki, "Napomena 2"))
        cout << "Tehnika uspjesno dodan!" << crt;
    if (jasmin->AddTehniku(ZUTI, oi_zuki, "Napomena 3"))
        cout << "Tehnika uspjesno dodan!" << crt;
    if (jasmin->AddTehniku(ZUTI, choku_zuki, "Napomena 4"))
        cout << "Tehnika uspjesno dodan!" << crt;
    //ne treba dodati choku_zuki jer je vec dodana za zuti pojas
    if (!jasmin->AddTehniku(ZUTI, choku_zuki, "Napomena 5"))
        cout << "Tehnika NIJE uspjesno dodana!" << crt;

    /*nakon evidentiranja tehnike na bilo kojem pojasu kandidatu se salje
email sa porukom:
    FROM:info@karate.ba
    TO: emailKorisnika
    Postovani ime i prezime, evidentirana vam je thenika X za Y pojas.
Dosadasnji uspjeh (prosjek ocjena)
    na pojasu Y iznosi F, a ukupni uspjeh (prosjek ocjena) na svim pojasevima
iznosi Z.
    Pozdrav.
    KARATE Team.
    ukoliko je prosjek na nivou tog pojasa veci od 4.5 kandidatu se salje SMS
sa porukom: "Svaka cast za
uspjeh 4.D na X pojasu".
    slanje poruka i emailova implemenitrati koristeci zasebne thread-ove.
    */
```

```
    cout << *jasmin << crt;

    //vraca kolekciju tehnika koji sadrze najmanje jednu ocjenu evidentiranu
u periodu izmedju
    // proslijedjenih datuma
        //float se odnosi na prosjecan broj dana izmedju ostvarenih ocjena na
tehnici
        Kolekcija<Tehnika*, float> jasminUspjeh = (*jasmin)(Datum(18, 06,
2021), Datum(21, 06, 2021));
    for (size_t i = 0; i < jasminUspjeh.getTrenutno(); i++)
        cout << *jasminUspjeh.getElement1(i) << " " <<
jasminUspjeh.getElement2(i) << crt;

    delete jasmin;
    delete adel;
    delete emailNotValid;

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