#include <iostream>

using namespace std;

//Z0.1

char\* AlocirajIKopiraj(const char\* izvor);

//Z0.2 :: Funkcija vraca max. od dva elementa

template<typename T>

T Max(T el1, T el2);

//Z0.3 :: Funkcija vraca min. od dva elementa

template<typename T>

T Min(T el1, T el2);

class Datum

{

private:

    shared\_ptr<int> \_dan;

    shared\_ptr<int> \_mjesec;

    shared\_ptr<int> \_godina;

public:

    //Z1.1

    Datum();

    //Z1.2

    Datum(int dan, int mjesec, int godina);

    //Z1.3

    Datum(const Datum& obj);

    //Z1.4

    Datum(Datum&& obj);

    //Z1.5

    Datum& operator =(const Datum& obj);

    //Z1.6

    void SetDan(int dan);

    void SetMjesec(int mjesec);

    void SetGodina(int godina);

    int GetDan() const;

    int GetMjesec() const;

    int GetGodina() const;

    //Z1.7

    ~Datum();

    friend ostream& operator << (ostream& COUT, const Datum& datum);

    friend bool operator == (const Datum& d1, const Datum& d2);

    friend bool operator > (const Datum&, const Datum&);

};

//Z1.8

bool operator == (const Datum& d1, const Datum& d2);

//Z1.9

bool operator > (const Datum& d1, const Datum& d2);

bool operator >= (const Datum& d1, const Datum& d2);

bool operator < (const Datum& d1, const Datum& d2);

bool operator <= (const Datum& d1, const Datum& d2);

//Z1.10

ostream& operator << (ostream& COUT, const Datum& datum);

//Kolekcija elemenata (tip T) sa mogucnoscu dinamickog prosirivanja

template<class T>

class DinamickaKolekcija {

private:

    int \_trenutno; //oznacava velicinu niza

    int \_maxElemenata;

    T\* \_elementi;

public:

    //Z2.1

    DinamickaKolekcija(int maxElemenata = 100);

    //Z2.2

    DinamickaKolekcija(const DinamickaKolekcija<T>& kolekcija);

    //Z2.3

    DinamickaKolekcija(DinamickaKolekcija<T>&& kolekcija);

    //Z2.4

    DinamickaKolekcija<T>& operator = (const DinamickaKolekcija<T>& kolekcija);

    //Z2.5 :: Osigurati da je index unutar opsega [0,\_trenutno-1]. Koristiti genericke funkcije 'Min' i 'Max'

    T& operator [] (int index) const;

    //Z2.6

    void ProsiriKolekciju(int prosiriZa);

    //Z2.7 :: Dodavanje novog elementa (ukoliko je \_trenutno < \_maxElemenata, uraditi prosirivanje niza za 10)

    bool operator += (T noviElement);

    //Z2.8

    bool UkloniZadnjiElement();

    //Z2.9

    int GetTrenutno() const;

    int GetMaxElemenata() const;

    ~DinamickaKolekcija();

    friend ostream& operator << <> (ostream& COUT, const DinamickaKolekcija<T>& kolekcija);

};

//Z2.10 :: Ispisuje elemente kolekcije. Za pristup elementima koristiti preklopljeni operator []

template<class T>

ostream& operator << <>(ostream& COUT, const DinamickaKolekcija<T>& kolekcija);

template<class T, int max> //max oznacava velicinu niza

class StatickaKolekcija {

private:

    int \_trenutno; //trenutno elemenata

    T\* \_elementi[max] = { nullptr }; //staticki niz pokazivaca

public:

    //Z3.1

    StatickaKolekcija();

    //Z3.2

    StatickaKolekcija(const StatickaKolekcija<T, max>& kolekcija);

    //Z3.3

    StatickaKolekcija(StatickaKolekcija<T, max>&& kolekcija);

    //Z3.4

    StatickaKolekcija<T, max>& operator=(const StatickaKolekcija<T, max>& kolekcija);

    //Z3.5 :: Osigurati da je index unutar opsega [0,\_trenutno-1]. Koristiti genericke funkcije 'Min' i 'Max'

    T& operator [] (int index) const;

    //Z3.6 :: Dodati (alocirati) novi element

    bool operator +=(T noviElement);

    //Z3.7 :: Izbrisati (dealocirati) zadnji element i umanjiti brojac

    bool UkloniZadnjiElement();

    //Z3.8

    int GetTrenutno() const;

    int GetMaxElemenata() const;

    ~StatickaKolekcija();

    friend ostream& operator << <>(ostream& COUT, const StatickaKolekcija<T, max>& kolekcija);

};

//Z3.9

template<class T, int max>

ostream& operator << <>(ostream& COUT, const StatickaKolekcija<T, max>& kolekcija);

class Sahista {

private:

    const char\* \_imePrezime;

    Datum \_datumRodjenja;

    const char\* \_drzava;

    shared\_ptr<bool> \_spol;

public:

    //Z4.1

    Sahista();

    //Z4.2

    Sahista(const char\* imePrezime, Datum datumRodjenja, const char\* drzava, bool spol);

    //Z4.3

    Sahista(const Sahista& obj);

    //Z4.4

    Sahista(Sahista&& obj);

    //Z4.5

    Sahista& operator = (const Sahista& obj);

    //Z4.6

    const char\* GetImePrezime() const;

    const char\* GetDrzava() const;

    Datum GetDatumRodjenja() const;

    bool GetSpol() const;

    //Z4.7

    void SetImePrezime(const char\* imePrezime);

    void SetDatumRodjenja(Datum datumRodjenja);

    void SetDrzava(const char\* drzava);

    void SetSpol(bool spol);

    //Z4.8

    ~Sahista();

    friend ostream& operator << (ostream& COUT, const Sahista& s);

    friend bool operator >(const Sahista& obj1, const Sahista& obj2);

    friend bool operator <(const Sahista& obj1, const Sahista& obj2);

    friend bool operator == (const Sahista& obj1, const Sahista& obj2);

};

//Z4.9

ostream& operator << (ostream& COUT, const Sahista& s);

//Z4.10 :: Odgovara na pitanje: "Da li je prvi sahista mladji od drugog?"

bool operator >(const Sahista& obj1, const Sahista& obj2);

//Z4.11 :: Odgovara na pitanje: "Da li je drugi sahista mladji od prvog?"

bool operator <(const Sahista& obj1, const Sahista& obj2);

//Vraca true ako sahisti imaju isto ime i prezime

bool operator == (const Sahista& obj1, const Sahista& obj2);

bool operator != (const Sahista& obj1, const Sahista& obj2);

//Z4.12 :: Koristiti genericku funkciju Max

Sahista\* VratiNajmladjegSahistu(DinamickaKolekcija<Sahista>& sahisti);

template<class T1, class T2>

class Par {

    T1 \_lijevi;

    T2 \_desni;

public:

    //Z5.1

    Par();

    Par(T1 el1, T2 el2);

    //Z5.2

    Par(const Par<T1, T2>& obj);

    Par(Par<T1, T2>&& obj);

    //Z5.3

    Par<T1, T2>& operator = (const Par<T1, T2>& obj);

    //Z5.4

    T1 GetLijevi() const;

    T2 GetDesni() const;

};

//Z5.5

template<class T1, class T2>

ostream& operator << (ostream& COUT, const Par<T1, T2>& par);

class SahovskiTurnir {

private:

    char\* \_naziv;

    char\* \_mjestoOdrzavanja;

    StatickaKolekcija<Par<Sahista, Sahista>, 20> \_dueli;

public:

    //Z5.6

    SahovskiTurnir();

    //Z5.7

    SahovskiTurnir(const char\* naziv, const char\* mjestoOdrzavanja);

    //Z5.8

    bool DodajDuel(Sahista& s1, Sahista& s2);

    //Z5.9

    ~SahovskiTurnir();

    friend ostream& operator <<(ostream& COUT, SahovskiTurnir& st);

};

//Z5.10 :: Ispisati naziv, mjesto, te sve duele

ostream& operator <<(ostream& COUT, SahovskiTurnir& st);

void Zadatak1() {

    Datum aprilFoolsDay;

    Datum laborDay(1, 5, 2021);

    Datum newYearsEve(31, 12, 2021);

    aprilFoolsDay.SetDan(1);

    aprilFoolsDay.SetMjesec(4);

    aprilFoolsDay.SetGodina(2021);

    Datum d1(aprilFoolsDay);

    Datum d2(move(laborDay));

    Datum d3;

    d3 = newYearsEve;

    cout << d1 << endl;

    cout << d2 << endl;

    cout << d3 << endl;

    cout << "Dealokacija..." << endl;

}

void Zadatak2() {

    cout << "Kolekcija datuma..." << endl;

    DinamickaKolekcija<Datum> datumi;

    cout << "Dodajemo 5 elemenata: " << endl;

    datumi += Datum(1, 5, 2020);

    datumi += Datum(7, 7, 2020);

    datumi += Datum(1, 3, 2020);

    datumi += Datum(7, 10, 2020);

    datumi += Datum(8, 12, 2020);

    cout << "Ispis elemenata..." << endl;

    cout << datumi << endl;

    cout << "Uklanjamo zadnji element...." << endl;

    datumi.UkloniZadnjiElement();

    cout << datumi << endl;

    cout << "Pravimo kopiju kolekcije: " << endl;

    DinamickaKolekcija<Datum> kopijaDatuma(datumi);

    cout << "Dodajemo jedan element (11.3.2019)" << endl;

    kopijaDatuma += Datum(11, 3, 2019);

    cout << kopijaDatuma << endl;

    cout << "Zatim premjestamo elemente (iz kopije) na novu lokaciju..." << endl;

    DinamickaKolekcija<Datum> noviDatumi(move(kopijaDatuma));

    cout << "Ispis nakon premjestanja elemenata..." << endl;

    cout << noviDatumi << endl;

    cout << "Kreiramo novu kolekciju i vrsimo naknadno kopiranje.. " << endl;

    DinamickaKolekcija<Datum> nova;

    nova = noviDatumi;

    cout << nova << endl;

    cout << "Dealokacija..." << endl;

}

void Zadatak3() {

    cout << "Kreiramo 'tudjeOcjene'" << endl;

    cout << "Dodajemo nove elemente u 'tudjeOcjene'..." << endl;

    StatickaKolekcija<int, 100> tudjeOcjene;

    tudjeOcjene += 9;

    tudjeOcjene += 10;

    tudjeOcjene += 10;

    tudjeOcjene += 10;

    tudjeOcjene += 9;

    tudjeOcjene += 10;

    tudjeOcjene += 10;

    cout << tudjeOcjene << endl;

    cout << "Kreiramo 'mojeOcjene' na osnovu 'tudjeOcjene'" << endl;

    StatickaKolekcija<int, 100> mojeOcjene(tudjeOcjene);

    cout << mojeOcjene << endl;

    cout << "Brisemo sve elemente iz 'mojeOcjene'" << endl;

    for (size\_t i = 0; i < 10; i++)

        mojeOcjene.UkloniZadnjiElement();

    cout << "Dodajemo nove (losije) ocjene u 'mojeOcjene'  .... " << endl;

    mojeOcjene += 6;

    mojeOcjene += 7;

    mojeOcjene += 6;

    mojeOcjene += 6;

    mojeOcjene += 7;

    mojeOcjene += 8;

    cout << mojeOcjene << endl;

    cout << "Kreiramo 'josGoreOcjene' na osnovu 'mojeOcjene' " << endl;

    StatickaKolekcija<int, 100> josGoreOcjene(move(mojeOcjene));

    cout << josGoreOcjene << endl;

    cout << "Brisemo sve elemente iz 'josGoreOcjene'..." << endl;

    for (size\_t i = 0; i < 10; i++)

        josGoreOcjene.UkloniZadnjiElement();

    cout << "Dodajemo nove (losije) ocjene u 'josGoreOcjene' .... " << endl;

    for (size\_t i = 0; i < 5; i++)

        josGoreOcjene += 6;

    cout << josGoreOcjene << endl;

    cout << "Kreiramo 'josGoreOcjene2' na osnovu 'josGoreOcjene' " << endl;

    StatickaKolekcija<int, 100> josGoreOcjene2;

    josGoreOcjene2 = josGoreOcjene;

    cout << josGoreOcjene << endl;

    cout << "Dealokacija..." << endl;

}

void Zadatak4() {

    Sahista s1("Gary Kasparov", Datum(2, 2, 1963), "Rusija", 1);

    Sahista s2(s1);

    Sahista s3(move(s2));

    Sahista garyKasparov;

    garyKasparov = s3;

    Sahista bobbyFischer("Bobby Fischer", Datum(4, 4, 1943), "SAD", 1);

    Sahista carlsen("Magnus Carlsen", Datum(3, 3, 1990), "Norveska", 1);

    Sahista judithPolgar("Judith Polgar", Datum(5, 5, 1976), "Madjarska", 0);

    DinamickaKolekcija<Sahista> sahisti;

    sahisti += garyKasparov;

    sahisti += bobbyFischer;

    sahisti += carlsen;

    sahisti += judithPolgar;

    cout << sahisti << endl;

    cout << "Najmladji sahista: " << endl << (VratiNajmladjegSahistu(sahisti)->GetImePrezime()) << endl;

    cout << "Dealokacija..." << endl;

}

void Zadatak5() {

    Sahista kingBobby("Bobby Fischek", Datum(4, 4, 1943), "SAD", 1);

    Sahista sahmatov("Viktor Sahmatov", Datum(4, 4, 1966), "Rusija", 1);

    Sahista chuckNorris("Chuck Mate", Datum(4, 4, 1966), "SAD", 1);

    SahovskiTurnir bugojno2021("MVST Bugojno (2021)", "Bugojno, BiH");

    bugojno2021.DodajDuel(kingBobby, sahmatov);

    bugojno2021.DodajDuel(sahmatov, chuckNorris);

    bugojno2021.DodajDuel(chuckNorris, kingBobby);

    cout << bugojno2021 << endl;

    cout << "Dealokacija..." << endl;

}

void Menu() {

    int nastaviDalje = 1;

    while (nastaviDalje == 1) {

        int izbor = 0;

        do {

            system("cls");

            cout << "::Zadaci::" << endl;

            cout << "(1) Zadatak 1" << endl;

            cout << "(2) Zadatak 2" << endl;

            cout << "(3) Zadatak 3" << endl;

            cout << "(4) Zadatak 4" << endl;

            cout << "(5) Zadatak 5" << endl;

            cout << "Unesite odgovarajuci broj zadatka za testiranje: -->: ";

            cin >> izbor;

            cout << endl;

        } while (izbor < 1 || izbor > 5);

        switch (izbor) {

        case 1: Zadatak1(); cout << "Zadatak 1. Done." << endl; break;

        case 2: Zadatak2(); cout << "Zadatak 2. Done." << endl; break;

        case 3: Zadatak3(); cout << "Zadatak 3. Done." << endl; break;

        case 4: Zadatak4(); cout << "Zadatak 4. Done." << endl; break;

        case 5: Zadatak5(); cout << "Zadatak 5. Done." << endl; break;

        default:break;

        }

        do {

            cout << "DA LI ZELITE NASTAVITI DALJE? (1/0): ";

            cin >> nastaviDalje;

        } while (nastaviDalje != 0 && nastaviDalje != 1);

    }

}

int main() {

    Menu();

    return 0;

}