#include <iostream>

#include <exception>

#include <functional>

using namespace std;

int Max(int a, int b) { return (a >= b) ? a : b; }

int Min(int a, int b) { return (a <= b) ? a : b; }

//Z0.1

bool CheckIfIsti(const char\* str1, const char\* str2);

//Z0.2

char\* Alokator(const char\* str);

//Z0.3

unique\_ptr<char> UniqueAlokator(const char\* str);

class Greska : public exception {

    int \_linija;

    const char\* \_funkcija;

public:

    \_declspec (property (get = GetPoruka)) const char\* Poruka;

    \_declspec (property (get = GetLinija, put = SetLinija)) int Linija;

    \_declspec (property (get = GetFunkcija, put = SetFunkcija)) const char\* Funkcija;

    //Z0.4

    Greska(const char\* poruka, int linija, const char\* funkcija);

    //Z0.5

    Greska(const Greska& obj);

    //Z0.6

    Greska(Greska&& obj);

    //Z0.7

    const char\* GetPoruka() const;

    int GetLinija() const;

    const char\* GetFunkcija() const;

    //Z0.8

    void SetLinija(int linija);

    void SetFunkcija(const char\* funkcija);

    //Z0.9

    Greska& operator =(const Greska& obj);

    //Z0.10

    ~Greska();

};

//Z0.11

ostream& operator <<(ostream& COUT, const Greska& obj);

template<class T>

class Vektor {

private:

    int \_trenutno;

    int \_maxElemenata;

    T\* \_elementi;

public:

    \_declspec(property (get = GetTrenutno)) int Length;

    //Z1.1

    Vektor(int maxElemenata = 10);

    //Z1.2

    Vektor(const Vektor<T>& obj);

    //Z1.3

    Vektor(Vektor<T>&& obj);

    //Z1.4

    Vektor<T>& operator = (const Vektor<T>& obj);

    //Z1.5

    T& operator [] (int index) const;

    int GetTrenutno() const;

    T\* GetElementi() const;

    //Z1.6

    void ProsiriVektor(int prosiriZa);

    //Z1.7 :: Dodavanje novog elementa

    //Ukoliko je \_elementi == nullptr, uraditi ponovnu alokaciju (\_maxElemenata = 10)

    //Ukoliko je \_trenutno == \_maxElemenata, prosiriti niz za 10

    void operator += (T noviElement);

    //Z1.8

    bool UkloniZadnjiElement();

    //Z1.9

    T\* CheckIfSadrzi(T element);

    //Z1.10

    bool Ukloni(T element);

    //Z1.11

    bool UkloniAt(int index);

    //Z1.12

    //Map je funkcija viseg reda (higher-order function) koja aplicira datu funkciju 'f' na svaki element niza.

    //Niz :[e1, e2, e3, e4, e5, ..., en]

    //Rezultat => [f(e1), f(e2), f(e3), f(e4), f(e5), ..., f(en)]

    void Map(function<void(T&)> f);

    //Z1.13

    //Filter je funkcija viseg reda (higher-order function) koja aplicira datu filter funkciju 'f' na svaki element niza.

    //Rezultat aplikacije filter funkcije je niz elemenata koji zadovoljavaju uslov (odredjen tom istom funkcijom)

    //Niz :[e1, e2, e3, e4, e5, ..., en]

    //Rezultat => [svaki element niza za kojeg vazi f(x) == true]

    Vektor<T> Filter(function<bool(T&)> f);

    //Z1.14

    //Reduce (accumulate) je funkcija viseg reda (higher-order function) zaduzena za agregaciju vrijednosti niza.

    //Funkcija reduce primjenjuje datu funkciju 'f' izmjedju susjednih elemenata da dobije medjurezultate.

    //----------------------------------------------

    //Primjer: Suma :: f(a,b) = a+b

    //Niz vrijednosti: [5, 1, 3, 4, 2]

    // f(5, 1), 3, 4, 2 => 6, 3, 4, 2

    // f(6, 3), 4, 2 => 9, 4, 2

    // f(9, 4), 2] => 13, 2

    // f(13, 2) => 15

    //----------------------------------------------

    //Primjer: Max :: f(a,b) = (a>=b)?a:b;

    //Niz vrijednosti: [5, 6, 3, 8, 2]

    // f(5, 6), 3, 8, 2 => 6, 3, 8, 2

    // f(6, 3), 8, 2 => 6, 8, 2

    // f(6, 8), 2 => 8, 2

    // f(8, 2) => 8

    //----------------------------------------------

    T Reduce(function<T(T, T)> f);

    //Z.15

    ~Vektor();

};

//Z1.16

template<class T>

ostream& operator << <>(ostream& COUT, const Vektor<T>& obj);

//Z1.17

template<class T>

void Ispis(const Vektor<T>& obj, const char\* delimiter = " ");

//Z1.18

template<class T>

bool operator == <>(const Vektor<T>& v1, const Vektor<T>& v2);

//Z1.19

template<class T>

bool operator != <>(const Vektor<T>& v1, const Vektor<T>& v2);

//Z1.20

template<class T>

bool operator == <>(const Vektor<const char\*>& v1, const Vektor<const char\*>& v2);

//Z1.21

template<class T>

bool operator != <>(const Vektor<const char\*>& v1, const Vektor<const char\*>& v2);

enum class Spol { Muski = 1, Zenski = 0 };

enum class Boja { Plava, Zelena, Zuta, Crvena, Crna, Bijela };

//Z1.22

ostream& operator << (ostream& COUT, const Boja& obj);

class Osoba {

protected:

    const char\* \_jmbg;

    pair<unique\_ptr<char>, unique\_ptr<char>> \_imePrezime;

    unique\_ptr<Spol> \_spol;

    float\* \_visina;

    Boja\* \_bojaOciju;

    Boja\* \_bojaKose;

public:

    \_declspec(property (get = GetJmbg, put = SetJmbg)) const char\* Jmbg;

    \_declspec(property (get = GetIme, put = SetIme)) const char\* Ime;

    \_declspec(property (get = GetPrezime, put = SetPrezime)) const char\* Prezime;

    \_declspec(property (get = GetSpol, put = SetSpol)) Spol Spol\_;

    \_declspec(property (get = GetVisina, put = SetVisina)) float Visina;

    \_declspec(property (get = GetBojaOciju, put = SetBojaOciju)) Boja BojaOciju;

    \_declspec(property (get = GetBojaKose, put = SetBojaKose)) Boja BojaKose;

    //Z3.1

    Osoba();

    //Z3.2

    Osoba(const char\* jmbg, const char\* ime, const char\* prezime, Spol spol, float visina, Boja bojaOciju, Boja bojaKose);

    //Z3.3

    Osoba(const Osoba& obj);

    //Z3.4

    Osoba(Osoba&& obj);

    //Z3.5

    const char\* GetJmbg() const;

    const char\* GetIme() const;

    const char\* GetPrezime() const;

    Spol GetSpol() const;

    float GetVisina() const;

    Boja GetBojaOciju() const;

    Boja GetBojaKose() const;

    //Z3.6

    void SetJmbg(const char\* jmbg);

    void SetIme(const char\* ime);

    void SetPrezime(const char\* prezime);

    void SetSpol(Spol spol);

    void SetVisina(float visina);

    void SetBojaOciju(Boja bojaOciju);

    void SetBojaKose(Boja bojaKose);

    //Z3.7

    Osoba& operator = (const Osoba& obj);

    //Z3.8

    ~Osoba();

};

//Z3.9

ostream& operator <<(ostream& COUT, const Osoba& osoba);

//Z3.10

bool operator == (const Osoba& o1, const Osoba& o2);

bool operator != (const Osoba& o1, const Osoba& o2);

class Teniser : public Osoba {

protected:

    Vektor<const char\*> \_osvojeniNaslovi;

    unique\_ptr<int> \_ranking;

public:

    \_declspec(property (get = GetOsvojeniNaslovi, put = SetOsvojeniNaslovi)) Vektor<const char\*> OsvojeniNaslovi;

    \_declspec(property (get = GetRanking, put = SetRanking)) int Ranking;

    //Z4.1

    Teniser();

    //Z4.2

    Teniser(const char\* jmbg, const char\* ime, const char\* prezime, Spol spol, float visina, Boja bojaOciju, Boja bojaKose,

        Vektor<const char\*> naslovi, int ranking);

    //Z4.3

    Teniser(const Teniser& obj);

    //Z4.4

    Teniser(Teniser&& obj);

    //Z4.5

    Vektor<const char\*> GetOsvojeniNaslovi() const;

    int GetRanking() const;

    //Z4.6

    void SetOsvojeniNaslovi(Vektor<const char\*> naslovi);

    void SetRanking(int ranking);

    //Z4.7

    Teniser& operator = (const Teniser& obj);

    //Z4.8

    void DodajOsvojeniNaslov(const char\* naslov);

    //Z4.9

    ~Teniser();

};

//Z4.10

ostream& operator <<(ostream& COUT, const Teniser& teniser);

//Z4.11

bool operator == (const Teniser& t1, const Teniser& t2);

//Z4.12

bool operator != (const Teniser& t1, const Teniser& t2);

class Mec {

protected:

    Teniser\* \_prvi;

    Teniser\* \_drugi;

    Vektor<pair<int, int>> \_setovi;

public:

    \_declspec(property (get = GetPrvi, put = SetPrvi)) Teniser Prvi;

    \_declspec(property (get = GetDrugi, put = SetDrugi)) Teniser Drugi;

    \_declspec(property (get = GetSetovi, put = SetSetovi)) Vektor<pair<int, int>> Setovi;

    //Z5.1

    Mec();

    //Z5.2

    Mec(Teniser prvi, Teniser drugi);

    //Z5.3

    Mec(const Mec& obj);

    //Z5.4

    Mec(Mec&& obj);

    //Z5.5

    Teniser GetPrvi() const;

    Teniser GetDrugi() const;

    Vektor<pair<int, int>> GetSetovi() const;

    //Z5.6

    void SetPrvi(Teniser teniser);

    void SetDrugi(Teniser teniser);

    void SetSetovi(Vektor<pair<int, int>> setovi);

    //Z5.7

    void DodajSet(int bodovi1, int bodovi2);

    //Z5.8

    Mec& operator = (const Mec& obj);

    //Z5.9

    ~Mec();

};

//Z5.10

ostream& operator <<(ostream& COUT, const Mec& teniskiMec);

//Z5.11

bool operator == (const Mec& m1, const Mec& m2);

bool operator != (const Mec& m1, const Mec& m2);

auto IspisiPoruku = [](const char\* tipFunkcije, bool success) {

    cout << tipFunkcije << " se izvrsio " << ((success) ? "uspjesno!" : "bezuspjesno!") << endl;

};

void Zadatak1() {

    //operacije [za mapiranje]

    auto pomnoziSa3 = [](int& broj) {broj \*= 3; };

    auto modulirajSa10 = [](int& broj) {broj %= 10; };

    //operacije [za primjenu filtera]

    auto veciOd5 = [](int& broj) {return broj > 5; };

    auto onlyEven = [](int& broj) {return broj % 2 == 0; };

    //operacije [za primjenu reduce]

    auto sumaBrojeva = [](int broj1, int broj2) {return broj1 + broj2; };

    Vektor<int> slucajniBrojevi;

    for (int i = 0; i < 30; i++)

        slucajniBrojevi += rand() % 40;

    cout << "Svi slucajni brojevi: " << endl;

    Ispis(slucajniBrojevi, " ");

    //Uklanjanje zadnjeg elementa

    cout << endl;

    slucajniBrojevi.UkloniZadnjiElement();

    cout << "Nakon uklanjanja zadnjeg ... " << endl;

    Ispis(slucajniBrojevi, " ");

    cout << endl;

    //Uklanjanje prvog elementa

    slucajniBrojevi.UkloniAt(0);

    cout << "Nakon uklanjanja prvog ... " << endl;

    Ispis(slucajniBrojevi, " ");

    cout << endl;

    //:::::::::::::::::: MAP ::::::::::::::::::

    slucajniBrojevi.Map(pomnoziSa3);

    cout << "Nakon mnozenja sa 3 ... " << endl;

    Ispis(slucajniBrojevi, " ");

    cout << endl;

    slucajniBrojevi.Map(modulirajSa10);

    cout << "Nakon moduliranja sa 10 ... " << endl;

    Ispis(slucajniBrojevi, " ");

    cout << endl;

    //:::::::::::::::::: FILTER ::::::::::::::::::

    cout << "Filtriranje [veci od 5]... " << endl;

    Ispis(slucajniBrojevi.Filter(veciOd5), " ");

    cout << endl;

    cout << "Filtriranje [veci od 5 i parni]... " << endl;

    Ispis(slucajniBrojevi.Filter(veciOd5).Filter(onlyEven), " ");

    cout << endl;

    //:::::::::::::::::: REDUCE ::::::::::::::::::

    cout << "[veci od 5 i parni] --> Suma()" << endl;

    cout << "Suma je: " << slucajniBrojevi.Filter(veciOd5).Filter(onlyEven).Reduce(sumaBrojeva) << endl;

    cout << endl;

    cout << "[veci od 5 i parni] --> Max()" << endl;

    cout << "Max element je: " << slucajniBrojevi.Filter(veciOd5).Filter(onlyEven).Reduce(Max) << endl;

    cout << endl;

    cout << "[veci od 5 i parni] --> Min()" << endl;

    cout << "Min element je: " << slucajniBrojevi.Filter(veciOd5).Filter(onlyEven).Reduce(Min) << endl;

    cout << endl;

    //:::::::::::::::::: COPY CTOR | MOVE CTOR | OPERATOR '=' ::::::::::::::::::

    Vektor<int> v2(slucajniBrojevi);

    IspisiPoruku("Copy ctor", slucajniBrojevi == v2);

    Vektor<int> v3(move(v2));

    IspisiPoruku("Move ctor", slucajniBrojevi == v3);

    Vektor<int> v4;

    v4 = slucajniBrojevi;

    IspisiPoruku("Operator '='", slucajniBrojevi == v4);

    //:::::::::::::::::: Testiranje funkcije Vektor<T>::Ukloni ::::::::::::::::::

    cout << endl << "Testiranje funkcije Vektor<T>::Ukloni" << endl;

    Vektor<int> noviVektor;

    noviVektor += 11;

    noviVektor += 2;

    noviVektor += 5;

    noviVektor += 5;

    noviVektor += 6;

    noviVektor += 7;

    noviVektor += 6;

    noviVektor += 11;

    Ispis(noviVektor, " ");

    int zaUklanjanje = 6;

    noviVektor.Ukloni(zaUklanjanje);

    cout << "Nakon uklanjanja (jedne instance) broja " << zaUklanjanje << endl;

    Ispis(noviVektor, " ");

    zaUklanjanje = 11;

    noviVektor.Ukloni(zaUklanjanje);

    cout << "Nakon uklanjanja (jedne instance) broja " << zaUklanjanje << endl;

    Ispis(noviVektor, " ");

    cout << endl << "Dealokacija..." << endl;

}

void Zadatak2() {

    Vektor<const char\*> rijeci;

    const char\* kolekcijaRijeci[] = { "reverenda","nakasljati se","podrugljivo", "trgnuti", "zajapuriti",

        "dozlogrditi", "zamajavati", "upravljati", "doskociti", "kalota" };

    for (size\_t i = 0; i < size(kolekcijaRijeci); i++)

        rijeci += kolekcijaRijeci[i];

    Ispis(rijeci, ", ");

    //:::::::::::::::::: COPY CTOR | MOVE CTOR | OPERATOR '=' ::::::::::::::::::

    Vektor<const char\*> v2(rijeci);

    IspisiPoruku("Copy ctor", rijeci == v2);

    Vektor<const char\*> v3(move(v2));

    IspisiPoruku("Move ctor", rijeci == v3);

    Vektor<const char\*> v4;

    v4 = rijeci;

    IspisiPoruku("Operator '='", rijeci == v4);

    //:::::::::::::::::: FILTER ::::::::::::::::::

    auto pocinjeSaZ = [](const char\* rijec) { return rijec[0] == 'z'; };

    cout << "Rijeci koje pocinju sa 'Z': " << endl << rijeci.Filter(pocinjeSaZ) << endl;

    //:::::::::::::::::: REDUCE ::::::::::::::::::

    auto getDuzaRijec = [](const char\* rijec1, const char\* rijec2) { return strlen(rijec1) >= strlen(rijec2) ? rijec1 : rijec2; };

    cout << "Najduza rijec: " << endl << rijeci.Reduce(getDuzaRijec) << endl;

    cout << endl << "Dealokacija..." << endl;

}

void Zadatak3() {

    Osoba putin;

    putin.Jmbg = "0710952215003";

    putin.Ime = "Vladimir";

    putin.Prezime = "Putin";

    putin.Spol\_ = Spol::Muski;

    putin.BojaKose = Boja::Plava;

    putin.BojaOciju = Boja::Plava;

    putin.Visina = 1.68f;

    cout << putin << endl;

    //:::::::::::::::::: COPY CTOR | MOVE CTOR | OPERATOR '=' ::::::::::::::::::

    try

    {

        Osoba o2(putin);

        IspisiPoruku("Copy ctor", putin == o2);

        Osoba o3(move(o2));

        IspisiPoruku("Move ctor", putin == o3);

        Osoba o4;

        o4 = putin;

        IspisiPoruku("Operator '='", putin == o4);

        cout << endl << "Dealokacija..." << endl;

    }

    catch (const Greska& greska)

    {

        cout << greska << endl;

    }

}

void Zadatak4() {

    Vektor<const char\*> titule;

    titule += "Australian Open 2020";

    titule += "Italian Open 2020";

    titule += "Cincinnati Masters 2020";

    Teniser djole;

    djole.Jmbg = "2205986345113";

    djole.Ime = "Novak";

    djole.Prezime = "Djokovic";

    djole.Spol\_ = Spol::Muski;

    djole.Visina = 1.88f;

    djole.BojaKose = Boja::Crna;

    djole.BojaOciju = Boja::Zelena;

    djole.OsvojeniNaslovi = titule;

    djole.Ranking = 11063;

    cout << djole << endl;

    //:::::::::::::::::: COPY CTOR | MOVE CTOR | OPERATOR '=' ::::::::::::::::::

    try {

        Teniser t2(djole);

        IspisiPoruku("Copy ctor", djole == t2);

        Teniser t3(move(t2));

        IspisiPoruku("Move ctor", djole == t3);

        Teniser t4;

        t4 = djole;

        IspisiPoruku("Operator '='", djole == t4);

        cout << endl << "Dealokacija..." << endl;

    }

    catch (const Greska& greska)

    {

        cout << greska << endl;

    }

}

void Zadatak5() {

    Teniser djole;

    djole.Jmbg = "2205986345113";

    djole.Ime = "Novak";

    djole.Prezime = "Djokovic";

    djole.Spol\_ = Spol::Muski;

    djole.Visina = 1.88f;

    djole.BojaKose = Boja::Crna;

    djole.BojaOciju = Boja::Zelena;

    djole.Ranking = 11063;

    Teniser nadal;

    nadal.Jmbg = "2304984344414";

    nadal.Ime = "Rafael";

    nadal.Prezime = "Nadal";

    nadal.Spol\_ = Spol::Muski;

    nadal.Visina = 1.83f;

    nadal.BojaKose = Boja::Crna;

    nadal.BojaOciju = Boja::Crna;

    nadal.Ranking = 9500;

    Mec finale;

    finale.Prvi = djole;

    finale.Drugi = nadal;

    finale.DodajSet(6, 1);

    finale.DodajSet(5, 7);

    finale.DodajSet(6, 4);

    cout << finale;

    //:::::::::::::::::: COPY CTOR | MOVE CTOR | OPERATOR '=' ::::::::::::::::::

    try

    {

        Mec m2(finale);

        IspisiPoruku("Copy ctor", finale == m2);

        Mec m3(move(m2));

        IspisiPoruku("Move ctor", finale == m3);

        Mec m4;

        m4 = finale;

        IspisiPoruku("Operator '='", finale == m4);

        cout << endl << "Dealokacija..." << endl;

    }

    catch (const Greska& greska)

    {

        cout << greska << endl;

    }

}

int main() {

    int nastaviDalje = 1;

    while (nastaviDalje) {

        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 broj za testiranje zadatka: -->: ";

            cin >> izbor;

            cout << endl;

        } while (izbor <= 0 || izbor > 5);

        switch (izbor) {

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

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

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

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

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

        default:

            break;

        }

        do {

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

            cin >> nastaviDalje;

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

    }

    return 0;

}