AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Szoftver laboratórium 2. 3. ellenőrző dolgozat. 2012.03.27. Kurz/**Terem**: L3/	**15 perc**
Név: **Neptun:**	Összpont:

1.feladat 4.5+1.5 pont

Valósítson meg C++ nyelven egy olyan osztályt (***Pontok***), ami tetszőleges számú *Pont* típusú objektumot képes tárolni! Tételezze fel, hogy a *Pont* osztály létezik, és síkbeli pontok valós koordinátáit tárolja! A *Pontok* osztályt úgy valósítsa meg, hogy másolható legyen és értékadás bal és jobb oldalán is szerepelhessen. Legyen összehasonlító operátora, ami logikai **igaz** értékkel tér vissza, ha a két objektum azonos számú pontot tartalmaz, és minden tárolt pont a tárolás sorrendjében megegyezik.

Megadtuk a ***Pontok*** osztály deklarációját, és a tagfüggvények definícióját, de ez több helyen hiányos. A **pontozott** részekre írva **egészítse ki** az alábbi **kódrészletet** úgy, hogy a megadott főprogram az elvárásoknak megfelelően működjön, és ne lépjen fel memóriakezelési hiba! Nem kell minden pontozott részre írnia! Feltételezheti, hogy a **másoló konstruktor,** amely külön állományban van megvalósítva, **helyesen működik,** ezért azt **nem kell** megírnia!

A feladatlap hátoldalán **adja meg**, hogy **megoldása** minimálisan **milyen elvárásokat támaszt** a *Pont* osztállyal szemben! (pl: van olyan konstruktora, ami 2 valós számot kap paraméterként; van … operátora; stb.) (1.5p)

```
class Pontok {
    Pont* pPnt;      // dinamikus adatterület kezdőcíme, ahol tárolunk
    int db;            // tárolt Pontok száma (pontosan ennyi pont van)
public:
    // Konstruktor: n darab valós számpárt (2*n darab számot) kap, amiből
    // n darab Pont objektumot készít és letárolja
    Pontok(const double pv[], int n = 0) ........ :db(n) ...........{
        pPnt = new Pont[db];
        for (int i = 0; i < 2*db; i += 2)
            pPnt[i/2] = Pont(pv[i], pv[i+1]);
    }
    Pontok(const Pontok&);
    Pontok& operator=(const Pontok&) ..............;
    bool operator==(const Pontok&) const;
    ........~Pontok() { delete [] pPnt;  }
};
Pontok& Pontok::operator=(const Pontok& p) ................. {
    if (this != &p) {
        delete[] pPnt;
        db = p.db;
        pPnt = new Pont[db];
        for (int i = 0; i < db; i++)
            pPnt[i] = p.pPnt[i];
    }
    return *this;
}
bool Pontok::operator==(const Pontok& p) const {
    if (db != p.db)
        return false;
    for (int i = 0; i < db; i++)
        if (pPnt[i] != p.pPnt[i])
            return false;
    return true;
}
int main() {
    double dv[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    Pontok p1(dv, 4); // négy darab valós számpár (8 darab szám)
    Pontok p2(dv, 3);
    cout << (p1 == p2); // ezek nem azonosak
    p2 = p1 = p1;
    cout << (p1 == p2); // ezek azonosak
}
```

Pont osztállyal szemben támasztott elvárások:

- legyen létrehozható 2 db valós számból
- legyen default konstruktora
- legyen értékadás (operator=) operátora
- legyen egyenlőtlenség (operator!=) operátora, ami hamis értékkel tér vissza, ha a tárolt koordináták azonosak.