AAAAAAAAAAAAAAAAAAAAAA

Szoftver laboratórium II. 5. ellenőrző dolgozat. 2013.05.06. Kurz/**Terem**: L4/	**15 perc**
Név: **Neptun:**	Összpont:

1. feladat **4.5 pont**

A *Halak* osztály egy biológiai kísérlet adatait tárolja. Az osztály belső szerkezetét nem ismerjük, de minden tárolt adata lekérdezhető és beállítható *getter/setter* függvényekkel, melyek interfészét pontosan ismerjük:

Függvény	**Funkció**
`void Halak::setSejt(int);`	a vizsgált embrió sejtszáma
`int Halak::getSejt();`	sejtszám lekérdezése
`void Halak::setZ(double);`	vízben levő szennyezés mértéke
`double Halak::getZ();`	szennyezés mértékének lekérdezése

Az osztálynak számos egyéb tagfüggvénye is van, melyek pontos funkcióját/leírását a biológus kollégák ismerik, mi nem. Tudjuk, hogy az osztály nem perzisztens, csak automatikusan létrejövő konstruktorai vannak, és nincs read/write tagfüggvénye. **Hozzon** létre a *Halak* osztályból példányosított objektumok tárolására egy *HalakTar* osztályt, ami perzisztens és úgy viselkedik, mint egy *std::list tároló*! Legyenek a *HalakTar* osztálynak az *std::list* tárolónál megismert funkciójú konstruktorai (ld. megjegyzés)! **Deklarálja** és **valósítsa** meg a *HalakTar* osztályt úgy, hogy az alábbi programrészlet az elvárásoknak megfelelően működjön. Tételezze fel, hogy a fájlok írása/olvasása közben nem lép fel hiba!

```
Halak a1;
a1.setSejt(4); a1.setZ(1.3);
HalakTar t1, t2(5), t3(10, a1), t4(&a1, &a1+1); // KONSTRUKTOROK
...
std::ofstream ofs("Halak.dat"); t3.write(ofs);  // tároló kiírása fájlba
...
std::ifstream ifs("Halak.dat"); t2.read(ifs);   // tároló beolvasása fájlból
```

```
class HalakTar : public std::list<Halak> {
public:
    HalakTar(size_t s = 0, const Halak& v = Halak()) : std::list<Halak>(s, v) {}
    template <class Iter>
    HalakTar(Iter first, Iter last) : std::list<Halak>(first, last) {}
    void read(std::istream& is) {
        size_t siz;
        is >> siz;
        resize(0);
        for (size_t i = 0; i < siz; i++) {
            int l; double d; Halak hal;
            is >> l >> d;
            hal.setSejt(l);
            hal.setZ(d);
            push_back(hal);
        }
    }
    void write(std::ostream& os) {
        os << size() << std::endl;
        for (iterator it = begin(); it != end(); it++)
            os << it->getSejt() << ' ' << it->getZ() << std::endl;
    }
};
```

2. feladat **1.5 pont**

Tételezze fel, hogy a *HalakTar* osztály elkészült! **Írjon programrészletet**, amelyben létrehoz egy *HalakTar* objektumot 200 adat (*Halak*) tárolására, majd az *std::remove_if* algoritmussal távolítsa el az előző feladat *a1* adatával egyező tartalmú adatokat a tárolóból!

```
using namespace std;
HalakTar tar(100);
HalakTar::iterator it = remove_if(tar.begin(), tar.end(),
                                  bind2nd(equal_to<Halak>(), a1));
tar.erase(it, tar.end());
```