rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 2. ellenőrző dolgozat. 2016.03.21. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

Az alábbi osztályt valós számok tárolására készítettük:

```
class Tarolo {
    double *p;
    unsigned siz;
public:
    Tarolo(unsigned s = 0        ) :siz(s) { p = new double[siz]; }
    unsigned size() const                  { return siz; }
    double& operator[](unsigned i)         { return p[i]; }
    ~Tarolo() { delete[] p; }
    Tarolo(const Tarolo& t) {
        siz = t.siz;
        p = new double[siz];
        for (unsigned i = 0; i < siz; i++)
            p[i] = t.p[i];
    }
};
```

1. Magyarázza meg, hogy az alábbi függvény futtatásakor miért lép fel memóriahiba! (1p)

```
double osszeg(Tarolo& t) {
    double sum = 0;
    for (unsigned i = 0; i < t.size(); i++) sum += t[i];
    return sum;
}
```

Értékparaméter másoló konstruktorral másolódik be a verembe. Az alapértelmezett másoló konstruktor azonban nem jó, mivel az osztály dinamikus memóriaterületet foglal és tart nyilván.

Módosítsa a fenti függvényt, hogy ne jelentkezzen hiba! (1p)

2. Magyarázza meg, hogy az alábbi kódrészlet futtatásakor miért lép fel memóriahiba! (1p)

```
{ Tarolo t1(3), t2 = t1; }
```

Inicializáláskor másoló konstruktor hívódik. Az alapértelmezett másoló konstruktor azonban nem jó, mivel az osztály dinamikus memóriaterületet foglal és tart nyilván.

Módosítsa az osztályt, hogy a fenti kódrészlet futtatása ne okozzon hibát! Használja az üres helyeket az osztály deklarációjában! (3p)

3. Mi a hiba az alábbi kódrészletben? Adja meg a hiba jellegét is (fordítási vagy futási)! (2p)

```
{ Tarolo t1; Tarolo const t2;
  std::cout << t2.size();}
```

Az osztálynak nincs paraméter nélkül hívható konstruktora.
A size tagfüggvény nem konstans, így az konstans objektumra nem alkalmazható.
Ezek fordítási hibát okoznak.

Módosítsa az osztályt, hogy ne legyen hiba! Használja az üres helyeket az osztály deklarációjában! (2p)

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 2. ellenőrző dolgozat. 2016.03.21. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

Az alábbi osztályt egész számok tárolására készítettük:

```
class Tomb {
    int *p;
    unsigned siz;
public:
    Tomb(unsigned s = 0          ) :siz(s) { p = new int[siz]; }
    unsigned size()                        { return siz; }
    int&       operator[](unsigned i)      { return p[i]; }
    ~Tomb() { delete[] p; }
    Tomb& operator=(const Tomb& t) {
      if (this != &t) {
        delete[] p;
        siz = t.siz;
        p = new int[siz];
        for (unsigned i = 0; i < siz; i++) p[i] = t.p[i];
      }
      return *this;
    }


};
```

1. Magyarázza meg, hogy az alábbi kódrészlet futtatásakor miért lép fel memóriahiba! (1p)

```
{ Tomb t1(3), t2(3); t2 = t1 = t1; }
```

Az alapértelmezett értékadó operátor nem jó, mivel az osztály dinamikus memóriaterületet foglal és tart nyilván.

Módosítsa az osztályt, hogy a fenti kódrészlet futtatása ne okozzon hibát! Használja az üres helyeket az osztály deklarációjában! (5p)

2. Mi a hiba az alábbi kódrészletben? Adja meg a hiba jellegét is (fordítási vagy futási)! (2p)

```
{ Tomb t1[10]; Tomb t2(3); t2[1] = 2;}
```

Az osztálynak nincs paraméter nélkül hívható konstruktora.
Az operator[] nem referenciát ad vissza így bal oldalon nem használható.
Ezek fordítási hibát okoznak.

Módosítsa az osztályt, hogy ne legyen hiba! Használja az üres helyeket az osztály deklarációjában! (2p)