rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 2. ellenőrző dolgozat. 2016.03.21. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

Az alábbi osztályt valós számok tárolására készítettük.

```
class RealVektor {
    double* p;
    const unsigned siz; // vektor mérete
public:
    RealVektor(unsigned int n) :siz(n) { p = new double[siz]; }
    unsigned int size() const { return siz; }
    ~RealVektor() { delete[] p; }
    RealVektor(const RealVektor& v) :siz(v.siz) {
        p = new double[siz];
        for (unsigned i = 0; i < siz; i++)
            p[i] = v.p[i];
    }
    double& operator[](unsigned i) {
        return p[i];
    }
    const double& operator[](unsigned i) const {
        return p[i];
    }
};

std::ostream& operator<<(std::ostream& os, const RealVektor& v) {
    for (unsigned i = 0; i < v.size(); i++)
        os << v[i] << ", ";
    return os;
}
```

1. Egészítse ki az osztályt úgy, hogy az alábbi kódrészlet hiba nélkül (ide értve a memóriaszivárgást is), az elvárásoknak megfelelően működjön! Ha szükséges, használja az üres helyeket az osztály deklarációjában és azon kívül! Az elvárások megértéséhez a megjegyzések segítenek! Ügyeljen az osztály konstans adattagjára! (10p)

```
{
  RealVektor v3(3);
  for (unsigned i = 0; i < v3.size(); i++) v3[i] = i + 1.1;
  const RealVektor cv = v3;
  std::cout << cv[1] << std::endl;              // kiír: 1.1
  std::cout << cv << std::endl;                 // kiír: 1.1, 2.1, 3.1,
  v3[0] += 10;
  std::cout << v3 << std::endl;                 // kiír: 11.1, 2.1, 3.1,
}
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 2. ellenőrző dolgozat. 2016.03.21. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

Az alábbi osztályt egész számok tárolására készítettük:

```
class Vektor {
    int* p;
    const unsigned siz;    // vektor mérete
public:
    Vektor(unsigned int n) :siz(n) { p = new int[siz]; }
    unsigned int size() const { return siz; }
    ~Vektor() { delete[] p; }
    Vektor(const Vektor& v) :siz(v.siz) {
        p = new int[siz];
        for (unsigned i = 0; i < siz; i++)
            p[i] = v.p[i];
    }
    int& operator[](unsigned i) {
        if (i >= siz) throw "index hiba";
        return p[i];
    }
};
```

1. Az alábbi kódrészlet futtatásakor memóriahiba keletkezik!

```
{ Vektor v5(5); }
```

   Módosítsa az osztályt, hogy a fenti kódrészlet futtatása ne okozzon hibát! Használja az üres helyeket az osztály deklarációjában! (2p)

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

```
{ Vektor v5(5), v = v5; }
```

v inicializálásakor 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! Ügyeljen arra, hogy az osztálynak van konstans adattagja! (3p)

3. Egészítse ki az osztályt olyan tagfüggvénnyel, amivel az osztály indexelhető! Indexelési hiba esetén dobjon **`const char*`** hibát! Működjön az elvárt módon az alábbi kódrészlet! Használja az üres helyeket az osztály deklarációjában! (2p)

```
{ Vektor v(6); v[1] = 2; }
```

4. Rövid kódrészlettel mutassa be a hibakezelést! (2p)

```
try {
  Vektor v(6);
  v[13] = 11;
} catch (const char *p) {
    std::cout << p << std::endl;
}
```