3. labor kis ZH, 2015. április 2., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Készíts dinamikus tömb osztályt, amely double értékeket tárol! Az osztály két adattagja a dinamikusan foglalt tömbre mutató pointer, valamit a tömb mérete.

- Készíts egyparaméteres konstruktort, amely a létrehozandó tömb méretét kapja paraméterként, melynek legyen alapértelmezett értéke 1! Ha a megadott méret 1-nél kisebb, a konstruktor dobjon std::range_error típusú hibát! A konstruktor hozza létre a megadott méretű tömböt, állítsa be az osztály tagváltozóit!
- Készíts destruktort, másoló konstruktort, értékadás operátort, index operátort! Az index operátor lehessen balérték, és érvénytelen index esetén dobjon range_error hibát!
- Készíts main függvényt, melyben bemutatod az osztály használatát, és elkapod az esetleg dobott hibát!

max. 10p, megoldási idő: 20 perc

Program például:

```
#include <iostream>
#include <stdexcept>

class Dintomb{
    double *t;
    int n;
public:
    Dintomb(int n = 1) :n(n){
        if (n < 1)
            throw std::range_error("Dintomb::Dintomb: n<1");
        t = new double[n];
    }
    ~Dintomb(){ delete[]t; }
    Dintomb(const Dintomb &m) :t(NULL){
        *this = m;
    }
    const Dintomb &operator = (const Dintomb &m){
        if (this == &m)
            return *this;
        delete[]t;
        n = m.n;
        t = new double[n];
        for (int i = 0; i < n; i++)
            t[i] = m.t[i];
        return *this;
    }
    double &operator[](int i){
```

```
        if (i < 0 || i >= n)
            throw std::range_error("Dintomb::operator[]: range error");
        return t[i];
    }
/*    const double &operator[](int i)const{ //most nem feladat, de gyakran
van rá szükség
        if (i<0 || i >= n)
            throw std::range_error("Dintomb::operator[]: range error")
            return t[i];
    }
*/
};

int main(){
    try{
        Dintomb t, t2(10), t3(t2);
        t2[2] = 5;
        t = t2;
        // t3[20]=8;
        std::cout << t[2] << std::endl;
    }
    catch (const std::range_error &e){
        std::cerr << e.what() << std::endl;
    }
    return 0;
}
```