3. labor kis ZH, 2026. április 16., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Készíts float értékeket tároló Verem osztályt! A verem egy olyan adattároló, amelybe a push függvénnyel tudunk új értéket betenni, a pop függvénnyel pedig kivenni. Mindig az utoljára betett értéket tudjuk kivenni, azaz az egymást követ pop hívások fordított sorrendben adják az értékeket, mint ahogy a push-sal betettük.

Legyen:

- alapértelmezett konstruktor
- destruktor
- másoló konstruktor
- = operátor
- pop() függvény, amely kiveszi és visszaadja a legutoljára betett értéket, és egyúttal el is távolítja ezt a tárolóból, lecsökkentve ennek méretét. Üres veremből történő kivétel kivétel dobását váltsa ki!
- a push() függvényt nem kell megírnod, de feltételezheted, hogy létezik és jól működik, ha használni szeretnéd.
- Legyen main() függvény kivételkezeléssel, ahol kipróbálod a verem minden funkcióját.

max. 10p, megoldási idő: 20 perc

```
#include <cstdio>
#include <cstdlib>
#include <stdexcept>

class Verem {
    float* t; // osztály + adatok: 1p
    int n;
public:
    // tömb átméretezés: 2p (kell a másoló ctor-ban, =-ben és a push-ban, pop-ban, de csak
egyszer jár érte a pont)
    Verem() :t{ nullptr }, n{ 0 } {} // 0,5p
    ~Verem() { delete[]t; } // 0,5p
    Verem(const Verem& v) :t{ nullptr } { // 1p
        *this = v;
    }
    Verem& operator=(const Verem& v) { // 1p
        if (this == &v)
            return *this;
        delete[]t;
        n = v.n;
        t = new float[n];
        for (int i = 0; i < n; i++)
            t[i] = v.t[i];
        return *this;
    }
    void push(float uj) { // nem kellett
        float* temp = new float[n + 1];
        for (int i = 0; i < n; i++)
            temp[i] = t[i];
        temp[n] = uj;
        n++;
        delete[]t;
        t = temp;
    }
    float pop() { // 1p
        if (n == 0)
            throw std::domain_error("Ures verembol probaltal kivenni"); // 1p
        float* temp = new float[n - 1];
```

```
        for (int i = 0; i < n - 1; i++)
            temp[i] = t[i];
        n--;
        float ki = t[n];
        delete[]t;
        t = temp;
        return ki;
    }
};

int main() {
    try { // try+catch 1p
        Verem v; // kipróbálás: 2p

        v.push(2); // nem kellett push kipróbálás
        v.push(9);
        v.push(-3.1f);

        // Verem w{ v };
        Verem w = v, z;

        v.push(3.14f);
        v = v;
        z = (w = v);

        for (int i = 0; i < 3; i++)
            printf("%g\n", v.pop());
    }
    catch (const std::domain_error& h) {
        printf("%s\n", h.what());
    }
    return 0;
}
```