1. labor kis ZH, 2017. március 2., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Készíts függvényt, amely egy monoton növekvő értékű valós számokat tároló tömböt kap! A függvény egy új, éppen megfelelő méretű dinamikusan foglalt tömbbe tegye az eredményt, és adja vissza az új tömb címét és elemszámát is! A dinamikus memória kezeléséhez a new/delete operátorokat használja! Amelyik értéket paraméterként adja vissza, annál használjon referenciát! A függvény úgy másolja az új tömbbe a bemenő tömbben lévő értékeket, hogy ha két szomszédos tömbelem között 1-nél nagyobb a különbség, akkor az új tömbben a két érték közé kerüljön be az átlaguk is! (Ha nem nagyobb 1-nél, akkor csak az eredeti tömbelemeket másolja.) Pl. be: 1.2, 1.8, 3.0, 5.0, 6.0, 6.4, 7.6, 8.0. Ki: 1.2, 1.8, 2.4, 3.0, 4.0, 5.0, 6.0, 6.4, 7.0, 7.6, 8.0. (Itt a bemenő tömb mérete 8, a kimenő tömbé 11.)

Készíts main függvényt, amelyben kipróbálod a fenti függvényt: definiálsz egy tömböt a bemenő értékekkel, és meghívod a függvényt, majd kiírod az új tömb tartalmát. Használat után szabadítsd fel a lefoglalt dinamikus memóriát!

max. 10p, megoldási idő: 20 perc

```
double * bovito(double *be, int be_n, int & ki_n) {
    ki_n = be_n;
    for (int i = 1; i < be_n; i++)
        if (be[i] - be[i - 1]>1)
            ki_n++;
    double *ki = new double[ki_n];
    int j = 1;
    ki[0] = be[0];
    for (int i = 1; i < be_n; i++, j++) {
        if (be[i] - be[i - 1]>1)
            ki[j++] = (be[i] + be[i - 1])*0.5;
        ki[j] = be[i];
    }
    return ki;
}

#include <cstdio>

int main() {
    double be[8] = { 1.2, 1.8, 3.0, 5.0, 6.0, 6.4, 7.6, 8.0 };
    double *ki = NULL;
    int ki_n;
    ki = bovito(be, 8, ki_n);
    for (int i = 0; i < ki_n; i++)
        printf("ki[%2d] = %g\n", i, ki[i]);
    delete[]ki;
    return 0;
}
```