4. labor kis ZH, 2023. november 15., szerda 10-12

NÉV (olvasható): NEPTUN kód:

Írj interpoláló C függvényt, amely egy valós számokat tartalmazó tömböt vesz át! A függvény szúrjon be a tömb minden olyan szomszédos eleme közé egy új értéket, a két szomszédos elem átlagát, ahol a szomszédos elemek távolsága >1! A függvény az eredményt egy éppen megfelelő méretű dinamikus tömbbe helyezze el, és ezt az új tömböt adja vissza! Ne feledd, hogy az új tömb méretét is vissza kell adni!
Pl. Be: -8.1, -3.2, 0.8, 1.1, 2.0, 4.4; Ki: -8.1, -5.65, -3.2, -1.2, 0.8, 1.1, 2.0, 3.2, 4.4;

Egészítsd ki teljes programmá! A program egész addig hívja újra az interpoláló függvényt az aktuális bővített tömbbel, amíg az tud beleszúrni új elemet! A végeredmény tömböt írd is ki! Ne legyen memóriaszivárgás!

max. 10p, megoldási idő: 20 perc

Program például:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double* interpol(const double* tbe, int nbe, int* pnki) { // fejléc + return: 2p
    int plusz = 0;
    for (int i = 1; i < nbe; i++) // 2p
        if (fabs(tbe[i - 1] - tbe[i]) > 1.0)
            plusz++;
    *pnki = nbe + plusz; // 1p
    double* tki = (double*)malloc(*pnki * sizeof(double)); // 2p
    if (tki == NULL) // nem muszáj
        return NULL;
    int j = 0;
    tki[j++] = tbe[0]; // átmásolás: 2p
    for (int i = 1; i < nbe; i++) {
        if (fabs(tbe[i - 1] - tbe[i]) > 1.0)
            tki[j++] = (tbe[i - 1] + tbe[i]) * 0.5;
        tki[j++] = tbe[i];
    }
    return tki;
}

int main() {
    double t[6] = { -8.1, -3.2, 0.8, 1.1, 2.0, 4.4 };
    double* elozo = NULL, * akt = NULL;
    int nElozo = 6, nAkt = 0;
    akt = interpol(t, nElozo, &nAkt); // 2p
    while (nElozo != nAkt) { // 2p
        free(elozo);
        elozo = akt;
        nElozo = nAkt;
        akt = interpol(elozo, nElozo, &nAkt);
    }

    for (int i = 0; i < nAkt; i++) // 1p
        printf("%g ", akt[i]);

    free(elozo); // 1p
    free(akt);

    return 0;
}
```