5. labor kis ZH, 2024. november 27., szerda 10-12

NÉV (olvasható): NEPTUN kód:

A Krampuszt segítő program elkészítése a feladatod. Megoldásodban kerüld a kódismétlést!

Definiálj struktúrát, amely egy ember nevét (max. 40 betűből álló sztring) és jósági pontszámát (double típusú érték) tartalmazza! (2p)

Írj függvényt, amely egy ilyen tömböt vesz át, és helyben rendezi névsor szerint! (6p)

Írj függvényt, amely egy ilyen struktúrákból álló tömböt kap, és egy éppen megfelelő méretű dinamikus tömbben, névsorban visszaadja azokat az elemeket, amelyek jósági pontszáma negatív! A bemenő tömbben pozitív jósági pontszámmal rendelkező személyek is lehetnek, ők ne kerüljenek bele a visszaadott tömbbe! A bemenő tömböt a függvény nem változtathatja meg! Ne feledd a dinamikus tömb méretét is visszaadni! (6p)

Írj main függvényt, amelyben létrehozol és kezdeti értékekkel inicializálsz egy 5 elemű, ilyen struktúrákból álló nemdinamikus tömböt, majd kiírod névsor szerinti sorrendben azokat a személyeket, akiknek a jósági pontszáma negatív! Ne felejtsd el felszabadítani a dinamikusan foglalt memóriát! (2p)

max. 10p, megoldási idő: 25 perc

Program például:

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

typedef struct { // 2p (csak ha jó a méret és double)
    char nev[41];
    double josag;
} szemely;


void rendez(szemely t[], int n) { // fejléc: 1p
    for (int i = n - 1; i > 0; i--) // rendezés: 2p
        for (int j = 0; j < i; j++)
            if (strcmp(t[j].nev, t[j + 1].nev) > 0) { // összehasonlítás: 1p
                szemely temp = t[j]; // csere: 2p
                t[j] = t[j + 1];
                t[j + 1] = temp;
            }
}


szemely* rosszak(const szemely t[], int n, int* retn) { // fejléc: 1p
    int db = 0;
    for (int i = 0; i < n; i++) // számolás: 1p (ha > 0 van, nem jár pont)
        if (t[i].josag < 0)
            db++;
    *retn = db;
    szemely* rt = (szemely*)malloc(db * sizeof(szemely)); // foglalás: 1p
    if (rt == NULL)
        return NULL;
    db = 0;
    for (int i = 0; i < n; i++) // másolás: 1p
```

```
        if (t[i].josag < 0) {
            rt[db] = t[i];
            db++;
        }
    rendez(rt, db); // rendezés hívása az új tömbre: 1p, eredeti tömbre: -1p
    return rt; // tömb és mérete visszaadása: 1p
}


int main() {
    szemely t[5] = // 1p, ha az inicializálás is jó (csak kezdetiérték adással)
    { {"Bela", 2.9}, {"Denes", -0.1}, {"Aladar", 4.9}, {"Elemer", -5.3}, {"Cecil", -1.6} };

    int uj_n;
    szemely* uj_t = rosszak(t, 5, &uj_n); // fv. hívás: 1p

    for (int i = 0; i < uj_n; i++) // ha rossz: -1p
        printf("%s\n", uj_t[i].nev);

    free(uj_t); // ha nincs, -1p

    return 0;
}
```