5. labor kis ZH, 2019. december 4., szerda 10-12

NÉV (olvasható): NEPTUN kód:

Definiálj struktúrát, amely egy ember nevét (max. 40 betűből álló sztring) és egyedi azonosítószámát (egész érték) tartalmazza! (2p)

Írj függvényt, amely egy ilyen struktúrákból álló tömböt kap, és egy éppen megfelelő méretű dinamikus tömbben visszaadja azokat az elemeket, amelyek azonosítószáma 1000 és 2000 között van! Ne feledd a dinamikus tömb méretét is visszaadni! (8p)

Írj függvényt, amely egy ilyen struktúrákból álló tömböt kap, és rendezi azt saját helyén, azonosítószám szerint növekvő sorrendben! (7p)

Írj függvényt, amely egy ilyen struktúrákból álló tömböt kap, és kiírja az „emberek.xyz” fájlba! Szabadon dönthetsz, hogy bináris vagy szöveges fájlba írod-e ki. (6p)

Írj main függvényt, amelyben létrehozol és kezdeti értékekkel inicializálsz egy 3 elemű, ilyen struktúrákból álló nemdinamikus tömböt, majd bemutatod a három függvény használatát! Ne felejtsd el felszabadítani a dinamikusan foglalt memóriát! (7p)

max. 10p, megoldási idő: 25 perc

Program például:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    char nev[41];
    int azon;
} ember;

ember * kozott(const ember * be, int be_meret, int * pki_meret) {
    int db = 0;
    for (int i = 0; i < be_meret; i++)
        if (be[i].azon>1000 && be[i].azon < 2000)
            db++;
    ember * ki = (ember*)malloc(db*sizeof(ember));
    int j = 0;
    for (int i = 0; i < be_meret; i++)
        if (be[i].azon>1000 && be[i].azon < 2000)
            ki[j++] = be[i];
    *pki_meret = db;
    return ki;
}

void rendez(ember * t, int n) {
    for (int i = 0; i < n; i++) {
        int min = i;
        for (int j = i + 1; j < n; j++)
            if (t[j].azon < t[min].azon)
                min = j;
        if (min != i) {
            ember temp = t[i];
            t[i] = t[min];
            t[min] = temp;
        }
```

```
    }
}

void kiir_binbe(const ember * t, int n) {
    FILE * fp = fopen("ember.xyz", "wb");
    fwrite(t, sizeof(ember), n, fp);
    fclose(fp);
}

void kiir_textbe(const ember * t, int n) {
    FILE * fp = fopen("ember.xyz", "wt");
    for (int i = 0; i < n; i++)
        fprintf(fp, "%d %s\n", t[i].azon, t[i].nev);
    fclose(fp);
}

int main() {
    ember tomb[3] = { {"Arany János", 1817}, {"Szent István", 975}, {"Teller Ede", 1908} };
    rendez(tomb, 3);
    int n;
    ember * kozt = kozott(tomb, 3, &n);
    kiir_binbe(kozt, n); // vagy
    kiir_textbe(kozt, n); // vagy
    free(kozt);
    return 0;
}
```

Struktúra: 1p,
41 elemű tomb: 1p

Között:
Bemenő paraméterek: 1p
Kimenő paraméterek: 1p
Szükséges méret meghatározása: 2p
Memóriafoglalás: 2p
Másolás: 2p

Rendez:
Fejléc: 1p
Rendező algoritmus: 4p
Struktúra helyes kezelése: 2p

Kiír:
Fejléc: 1p
Fájlnyitás: 2p
Kiírás: 2p
Fájlzárás: 1p

Main
Tömbdefiníció és inicializálás: 2p
Rendez hívása: 1p
Között hívása: 2p
Kiírás hívása: 1p
Free: 1p

Össz: 2 + 8 + 7 + 6 + 7 = 30