4. labor kis ZH, 2017. november 22., szerda 10-12

NÉV (olvasható): NEPTUN kód:

A feladatok összpontszáma 14, de >=10 pont esetén 10 pont kerül beírásra. A feladatok egy teljes program részei, ne külön feladatként tekints az a)-c) részekre!

a) Készíts függvényt, amely egy valós számokból álló tömböt vesz át, és innen egy éppen megfelelő méretű dinamikus tömbbe másolja azokat az értékeket, amelyek a [-1, 1] zárt intervallumban találhatók! (6p)

b) Készíts függvényt, amely egy sztringet vesz át bemenetként, és egy olyan karaktertömböt, ahová az eredményt teszi! Ha a bemenő sztring legalább három szót tartalmaz, akkor a függvény a középső szót másolja az eredménytömbbe sztringként! Ha a bemenő sztring nem tartalmaz legalább három szót, akkor az eredmény sztring "-" legyen! Minden szót egy szóköz választ el egymástól. (Azaz egy mínuszjelet tartalmazó sztring.) (4p)

c) Egészítsd ki teljes programmá! (4p)

- A main függvényben hozz létre és inicializálj kezdőértékekkel egy valós értékeket tartalmazó tömböt, ezzel hívd meg az a) függvényt, majd írd ki az eredménytömb tartalmát!
- Hozz létre két karaktertömböt is, és az elsőbe a felhasználótól kérj be egy mondatnyi szöveget! (Ügyelj rá, hogy a beolvasás alkalmas legyen több szóból álló mondatok beolvasására!) Hívd meg ezzel a szöveggel a b) függvényt, majd írd ki a mondat második szavát!
- Ne legyen memóriaszivárgás!

max. 10p, megoldási idő: 25 perc

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

double * intervallum(double *be, int n, int *ki_n) {
    int db = 0, i, j;
    for (i = 0; i < n; i++)
        if (be[i] >= -1 && be[i] <= 1)
            db++;
    double *ki = (double*)malloc(db*sizeof(double));
    *ki_n = db;
    for (i = j = 0; i < n; i++)
        if (be[i] >= -1 && be[i] <= 1)
            ki[j++] = be[i];
    return ki;
}

void kozepso(char be[], char ki[]) {
    char *sp = strchr(be, ' ');
    if (sp != NULL)
        sp = strchr(sp+1, ' ');
    if (sp != NULL)
        sscanf(be, "%*s%s", ki);
    else
        strcpy(ki, "-");
}

int main(){
    double t[6] = { 0.3, 6, -1.1, -0.2, 0, 5 };
    int db, i;
    double *ki = intervallum(t, 6, &db);
    for (i = 0; i < db; i++)
        printf("%g ", ki[i]);
    free(ki);
    char s1[100], s2[100];
    printf("\n");
    fgets(s1, 100, stdin);
    kozepso(s1, s2);
    printf("%s\n", s2);
    return 0;
}
```

Intervallum: 6p
- tömb+méret, vissza tömb pointer, vissza méret pointer: 1p
- minden elemre ellenőrzi, hogy benne van-e az intervallumban: 1p
- db számítása: 1p
- din tömb foglalás: 1p
- átmásolás a kimeneti tömbbe: 1p
- pointer + db visszaadása: 1p

Középső: 4p
- be+ki tömb, méret nincs: 1p
- van benne 2 szóköz: 2 p
- középső szó másolása: 1p
- van/nincs 3 szó: ha hiányzik, -1p

Main: 4p
- double tömb + inicializálás: 1p
- intervallum hívása + visszatérési értékének tárolása: 1p
- kiírás helyes mérettel: 1p
- free nincs: -1p
- 2 karaktertömb hibás: -1p
- fgets: 1p
- kozepso hívása hibás: -1p
- sztring kiírása nincs: -0,5p