PROGRAMOZÁS ALAPJAI VIZSGAZÁRTHELYI DOLGOZAT 2011. JANUÁR 19.

1. Függvények, függvényprototípus, függvényhívási mechanizmus. Rekurzió. Függvény pointer.
2. A struktúra adattípus, struktúrapointerek, struktúratömbök. Típusdefiníció.

Definiáljon függvényeket a következő feladatokra:

a. Visszatérő értéke 1, ha a paraméterként megadott A hosszú valós típusú, N elemű tömb elemei mértani sorozatot alkotnak.

b. Paramétere: az X hosszú egész típusú, M elemű tömb. Írja ki a standard hibacsatornára azon elemek értékét, melyek a tömbben többször is előfordulnak. Visszatérő értéke nincs.

c. Paramétere az SZ karaktertömb. Visszatérő értéke 1, ha az SZ-ben tárolt szöveg visszafelé olvasva is ugyanaz (tükörszó), különben 0.

d. Paramétere: az RE hosszú valós típusú, H elemszámú tömb, ami egy részvény napi árfolyamait tartalmazza. Visszatérő értéke annak a napnak a sorszáma, amikor a részvény árfolyama elkezdett csökkenni (feltételezheti, hogy csak ilyen nap volt).

e. Paramétere a T N*N-es egész típusú tömb. Visszatérő értéke a főátló alatti elemek összegének értéke.

f. Dinamikusan helyet foglal a memóriában egy M elemű, hosszú egész típusú tömb számára és a tömb elemeit nullára állítja. Visszatérő értéke a tömb címe, sikertelenség esetén NULL.

```
/*****************************************************************************
 *
 * Programozás alapjai 2011 január 19-ei vizsga kidolgozott tételei. Feladat leírása a
 * forrás végén
 *
 * Kidolgozta: Pozsgay Máté <matthew.linux@gmail.com>
 * Utolsó módosítás: 2011.01.21
 *
 * ***************************************************************************/

#include <stdio.h>      //ki-be menet kezeléséhez
#include <stdlib.h>     //memóriafoglaláshoz
#include <time.h>       //idő lekérdezéséhez (véletlen véletlenszámokhoz)
#include <string.h>     //szövegkezeléshez

int afeladat(double *a, long int n){
        int i=2; //ciklust kettőről indítjuk, mert az első kettő elemet levizsgáljujk előtte
```

```
	double div=a[1]/a[0];  // <- itt vizsgáljuk le
	while(i<n && a[i]/a[i-1] == div)	//addig megyünk, amíg nem futunk a töömbből ÉS a jelenlegi, illetve az előző elem hányadosa megegyezik az első kettő hányadosával
		i++;	// lépünk a tömben
	return !(i<n);	// visszatérünk az (i<n) ál-logikai érték negáltjával (c-ben a 0 hamis, egyéb igaz)
}

void bfelatdat(int *m, int x){
	int i,j;
	for(i=0; i<x-1; i++){	//minden elemre (kivéve az utolsót) megvizsgáljuk, hogy megvan-e benne mégegyszer
		j=i+1;	//a vizsgálást az aktuálist követő elemmel kezdjük
		while(j<x && m[i] != m[j])	//és addig vizsgálunk, amíg nem megyünk ki a tömbből ÉS nem találunk egyezést
			j++; //lépünk
		if(j<x)	//ha nem futottunk ki a tömbből, az azt jelenti, hogy találtunk egyező elemet
			fprintf(stderr, "%d, ", m[i]);	//kiírjuk a szabványos hibacsatornára a talált elemet (illetve itt azt, amelyikkel összehasonlítjuk, de mivel a kettő ugyan az, ezért teljesen mindegy)
	}
}

int cfeladat(char * sz){
	int i=0, szl=strlen(sz); //szl tárolja a szöveg hosszát (gyorsabb itt egyszer kiszámolni, mint minden egyes ciklusmag végrehajtása után egyszer-egyszer)
	while(i<szl/2 && sz[i]==sz[szl-1-i])	//addig vizsgálunk, amíg el nem érjük a szöveg felét (elég csak a felét vizsgálni, hiszen tükörszó) ÉS az aktuális karakter megegyezik a végéről visszaszámolt aktuális indexű elemmel
		i++;	//lépünk
	return !(i<szl/2);	//ugyan az az elv, mint az a feladatban
}

int dfeladat(double *re, int h){
	int i=1; //1-ről indulunk, nem csökkenhet már az első napon (0) az árfolyam
	while(i<h && re[i-1]<=re[i])	//addig megyünk, amíg nem futunk ki a tömbből ÉS az előző elem kisebb, mint a jelenlegi.
		i++;	//lépünk
	return i+1;	//visszatérünk azzal az elemmel ahol elromlott a ciklusfejlécben megadott feltétel (+1, mert 0-tól kezdődik az indexelés, de mi emberek 1-től kezdünk számolni :))
}

int efaladat(int **T, int n){
   int osszeg=0;
   int i,j;
	for(i=1;i<n;i++){	//az első sorban nincs mit összeadni
```

```
                for(j=0;j<i;j++){                   //minden sorban egyel kevesebb elemet kell
összeadni, mint ahányadik sorban éppen vagyunk
                        osszeg+=T[i][j];        //hozzáadjuk az összeghez az aktuális elemet
                }
        }
        return osszeg;
}

void *ffeladat(int *tomb, int m){
                int *tmp=malloc(m*sizeof(int));         //foglalunk darabszámszor típusméret
                int i;
                if(tmp != NULL)         //ha sikerült foglalni
                        for(i=0; i<m; i++)      //akkor az egész tömböt
                                tmp[i]=0;       //nullázuk
                return tmp;     //visszatérünk a címmel (Ha nem sikerült foglalni, akkor sincs
baj, mert pont NULL-at kell visszaadni)
}

int main(){
        static int elemszam=10;
        int i, j;
        srand(time(NULL));    //véletlenszámgenerátor inicializálása

        //-----------------Első feladat-----------------------------------------------
        double *rt = malloc(elemszam * sizeof(double));
        printf("\nElső feladat:\n\t Adott a következő sorozat:\n\t");
                for(i=0; i<elemszam; i++){
                rt[i]=rand()%10;
                printf("%f, ", rt[i]);
        }
        printf("\n\tA sorozat mértani sorozat, ha a következő szám egyes, különben nulla:
%d", afeladat(rt, elemszam));

        //--------------Második feladat-------------------------------------------
        int *it=malloc(elemszam * sizeof(int));
        printf("\n\nMásodik feladat:\n\tA tömb (%d) elemei: ", elemszam);
        for(i=0; i<elemszam; i++){
                it[i]=rand()%10;
                printf("%d, ", it[i]);
        }
        printf("\n\tIsmétlődő elemek (stderr-en):\n");
        bfelatdat(it, elemszam);

        //--------------Harmadik feladat-----------------------------------------
        char *karaktertomb = "indulagorogaludni";
        char *str=malloc(strlen(karaktertomb) * sizeof(char));
        strcpy(str, karaktertomb);
        //ha nem így csinálod, orbitális szopás lehet a következménye. Amíg nem akarod írni,
```

(mint jelen eseten) addig csinálhatod az egyszerű egyenlőség adással, de ha módosítani is akarod, akkor így kell, különben - bár nem mondod neki, hogy statikus legyen - string literallal egyenlővé téve védet memóriára fog mutatni, amit nem írhatsz, és kapod a segfaultot, ha megpróbálod.

```
	printf("\n\nHarmadik feladat:\n\tA megadott szöveg: \"%s\" tükörszó, ha a következő érték 1, nulla, ha nem az: %i", str, cfeladat(str));

	//--------------Negyedik feladat-------------------------------------------
	printf("\n\nNegyedik feladat:\n\tAz árfolymok:");
	for(i=0; i<elemszam; i++){
		rt[i]=rand()%10000/100.0; //normális számokat generáljunk
		printf("%f, ", rt[i]);
	}
	printf("\n\tAz első csökkenés a %d napon történt!",dfeladat(rt, elemszam));

	//--------------Ötödik feladat---------------------------------------------
	int **im=malloc(elemszam*sizeof(*im));	//definiáljuk a tömböt (mutatóra mutató mutató), mérete pontosan n darab * az önmaga típusának egyszeres mutatója
	for(i=0; i<elemszam; i++)//Memóriát foglalunk a mutatók számára
		im[i]=malloc(elemszam*sizeof(*im[i]));
	printf("\nÖtödik feladat:\n\tA mátrixunk:\n\t\t");
	for(i=0; i<elemszam; i++){	//töltsük fel a tömböt
		for(j=0; j<elemszam; j++){
			im[i][j]=rand()%10;
			printf("%d ",im[i][j]);
		}
		printf("\n\t\t");
	}
	printf("\n\tA főátló alatti elemek összege: %d",efaladat(im,elemszam));

	//--------------Hatodik feladat--------------------------------------------
	int *nt = ffeladat(nt, elemszam);
	printf("\n\nHatodik feladat:\n\tA foglalt memóriacím: %p", nt);

	//-----------------Felszabadítás-----------------------------------------
	free(rt);		//Visszaadjuk az operációs rendszernek a dinamikusan foglalt memóriát, ugyanis nem vagyunk tahó programozók ;)
	free(str);
	free(it);
	free(im);
	free(nt);
	return 0;
}

/*
 Definiáljon függvényeket a következő feladatokra:
	a. Visszatérő értéke 1, ha a paraméterként megadott A hosszú valós típusú, N elemű
```

tömb elemei mértani sorozatot alkotnak.

b. Paramétere: az X hosszú egész típusú, M elemű tömb. Írja ki a standard hibacsatornára azon elemek értékét, melyek a tömbben többször is előfordulnak. Visszatérő értéke nincs.

c. Paramétere az SZ karaktertömb. Visszatérő értéke 1, ha az SZ-ben tárolt szöveg visszafelé olvasva is ugyanaz (tükörszó), különben 0.

d. Paramétere: az RE hosszú valós típusú, H elemszámú tömb, ami egy részvény napi árfolyamait tartalmazza. Visszatérő értéke annak a napnak a sorszáma, amikor a részvény árfolyama elkezdett csökkenni (feltételezheti, hogy csak ilyen nap volt).

e. Paramétere a T N*N-es egész típusú tömb. Visszatérő értéke a főátló alatti elemek összegének értéke.

f. Dinamikusan helyet foglal a memóriában egy M elemű, hosszú egész típusú tömb számára és a tömb elemeit nullára állítja. Visszatérő értéke a tömb címe, sikertelenség esetén NULL.
*/