Cand copiez codul din Codeblocks in Word, se strica foarte rau formatarea codului. Ca sa nu stau sa o refac si in document, o las asa si va invat sa o formatati odata ce o puneti in Codeblock. Un CTRL-A va selecta totul codul si apoi sus in tab-uil de Plugins exista Source Code Formatter ceva aranja codul foarte frumos :D.

Se citesc *n, m* şi apoi două mulţimi A şi B cu *n*, respectiv *m* numere întregi cuprinse între *[-x,x], x<=2000*. Să se afişeze numărul de elemente comune mulţimii. (*Indicaţie:* mulţimile A şi B nu se vor memora - se va crea un vector de frecvenţă).

Ideea vectorului de frecventa este simpla. Dat fiind faptul ca noi avem o plaja de valori fixa (-2000 >= x <= 2000), putem face un vector ce va avea o intrare corespunzatoare pentru fiecare posibilitate. Putem asigna valoarea unu pe pozitiile unde avem elementul in multime (tineti minte ca multimile nu au elemente duplicate). Cand citim a doua multime, daca gasim deja valoarea unu pe pozitia respectiva, putem incrementa un contor de elemente comune. Folosim indexul vectorul drept x+2000 pentru ca maparea valorilor este (pozitia 0=-2000), (pozitia 1=-1999)....(pozitia 2000=0)....(pozitia 3999=1999), (pozitia 4000=2000).
Am vazut ca multi dintre noi am afisat de fapt elementele comune la laborator. Este destul de mica diferenta de cod oricum (schimbam if-ul din al doilea for). Nu mai incrementam contorul, ci afisam direct vectorFrecventa[x+2000].

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

int vectorFrecventa[4001];

int main()
{
	int marimeN, marimeM;

	printf("Introduceti cardinalele pentru cele doua multimi:");
	scanf("%d %d", &marimeN, &marimeM);

	for(int i=0;i<marimeN;i++)
	{
	int x;
	scanf("%d", &x);
	vectorFrecventa[x+2000] = 1;
	}

	int contorElementeComune = 0;
	for(int i=0;i<marimeM;i++)
	{
	int x;
	scanf("%d", &x);
	if (vectorFrecventa[x+2000] == 1)
	contorElementeComune++;
	}

	printf("Avem %d elemente comune",contorElementeComune);
	return 0;
}
```

Se citesc: *n,* cele *n* elemente ale unui vector sortat crescator, apoi *x* şi *y* două elemente din vector. Să se afişeze toate elementele vectorului cuprinse între *x* şi *y*. *Optim:* folosiţi căutarea binară.

Observaţie: **https://research.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html**

Exercitiul asta l-am facut pe larg la tabla, dar il reiau rapid. Cautarea binara este mult mai eficienta despre un for pe toata lungimea vectorului cu un if inauntru ce intreaba daca numarul de pe pozitia i este mai mare decat X si mai mic decat Y.
Cautarea binara imparte vectorul sortat in doua si pe baza valorii din mijloc, se in jumatatea din stanga sau din dreapta (de aici si numele de "binara"). Astfel, gasim indicii celor doua elemente si putem sa afisam foarte usor elementele intermediare.

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

int cautareBinara(int vec[], int key, int marimeVec)
{
	int low = 0;
	int high = marimeVec - 1;

	while (low <= high)
	{
	int mid = (low + high) / 2;
	int midVal = vec[mid];

	if (midVal < key)
	low = mid + 1;
	else if (midVal > key)
	high = mid - 1;
	else
	return mid;
	}

	return -(low + 1);
}


int main()
{
	int marimeVec;

	printf("Introduceti marimea unui vector sortat: ");
	scanf("%d", &marimeVec);
	printf("\n");

	int vecSortat[marimeVec];
	for(int i=0;i<marimeVec;i++)
	{
	scanf("%d", &vecSortat[i]);
	}

	int x, y;
```

```
	printf("Introduceti doua elemente din vector: ");
	scanf("%d %d", &x, &y);

	int indexX = cautareBinara(vecSortat, x, marimeVec);
	int indexY = cautareBinara(vecSortat, y, marimeVec);

	for(int i=indexX;i<=indexY;i++)
	{
	printf("%d ", vecSortat[i]);
	}

	return 0;
}
```

Se citesc *x, y*, două numere mari (fiecare având peste 20 de cifre). Să se calculeze suma lor (folosind vectori). Numerele sunt naturale. Numerele sunt întregi.

Am facut o functie ce converteste un char la int. Cifrele se afla intre 48 si 57 in codul ASCII. Daca iteram prin aceste valori si convertim valoarea numerica a lui i in caracter, putem gasi cifra reprezentata in variabila a si sa returnam valoarea acesteia cand fiind i-48.

Codul este deceptiv de simplu. Tinem cele doua numere in 2 vectori de caractere. Le inversam pozitiile inca de la inceput (ganditi-va la modul in care le citim). Le luam lungimile si apoi luam lungimea maxima pentru a itera pana la valoarea acesteia. Cat timp iteratorul i este mai mic decat ambele lungime, facem adunarile corespunzatoare, altfel doar copiem valoarea din numarul1 sau numarul2 in rezultat. La final pastram intr-o variabila carry excesul de la aceasta adunare, pentru a fi utilizat la urmatoarea iteratie.

Trebuie sa mai punem inca o data adunarea excesului dupa acest for apoi doar afisam vectorul rezultat, dar in mod invers pentru ca inca am inversat la inceput, inainte sa facem suma.

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

int charToInt(char a)
{
	int i;
	for (i = 48; i<=57; i++)
	{
		if (toascii(i) == a)
		{
			return i-48;
		}
	}
	return 0;
}



int main()
{
	char numar1[20];
```

```
char numar2[20];
int rezultat[20];
int lungimeNumar1, lungimeNumar2;

int i,j,carry, lungimeMax, sum;

printf("Primul numar: ");
scanf("%s", &numar1);
printf("\nAl doilea numar: ");
scanf("%s", &numar2);

lungimeNumar1 = strlen(numar1);
lungimeNumar2 = strlen(numar1);

strrev(numar1);
strrev(numar2);

lungimeMax = lungimeNumar1;
if(lungimeNumar1<lungimeNumar2)
{
	lungimeMax = lungimeNumar2;
}

carry=0;
for(i=0; i< lungimeMax; i++)
{
	if(lungimeNumar1==lungimeNumar2 || (i < lungimeNumar1 && i < lungimeNumar2))
	{
		sum = carry + charToInt(numar1[i])+charToInt(numar2[i]);
	}
	else if(i >= lungimeNumar1)
	{
		sum = carry + charToInt(numar2[i]);
	}
	else if(i >= lungimeNumar2)
	{
		sum = carry + charToInt(numar1[i]);
	}
	rezultat[i] = sum % 10;
	carry = sum / 10;
}

if(carry > 0)
{
	rezultat[i] = carry;
	i++;
}



printf("\nRezultat: ");
for(j=0; j < i; j++)
{
```

```
            printf("%d", rezultat[i-j-1]);
        }

        return 0;
}
```

Se citesc de la tastatură *m* şi *n* naturale nenule reprezentând dimensiunile unei matrice şi elementele matricei. Să se construiască şi să se afişeze matricea transpusă.

Codul este plin de afisari, pentru fiecare element citit si apoi pentru toata matricea. Ce vrem cu adevarat sa vedem este:
//////////////

```
    for(i=0; i<nrRanduri; ++i)
    {
            for(j=0; j<nrColoana; ++j)
            {
                transpusa[j][i] = matrice[i][j];
            }
    }
```

//////////////

Nu este mult de zis. Cand copiem datele din matricea citita in matricea transpusa, inversam indicii astfel incat in transpusa sa se scrie pe coloana si apoi pe linie.

Disclaimer: Stiu ca varianta mea este putin ineficienta pentru ca tinem matricea in memorie de 2 ori (pentru matrice mari, poate fi o problema).

```
#include <stdio.h>

int main()
{
    int nrRanduri, nrColoana, i, j;
    printf("Numarul de randuri si de coloane: ");
    scanf("%d %d", &nrRanduri, &nrColoana);

    int matrice[nrRanduri][nrColoana];
    int transpusa[nrRanduri][nrColoana];

    printf("\nElementele matricei:\n");
    for(i=0; i<nrRanduri; ++i)
    {
            for(j=0; j<nrColoana; ++j)
            {
                printf("Elementul %d %d: ",i+1, j+1);
                scanf("%d", &matrice[i][j]);
            }
    }



    printf("\nMatricea: \n");
    for(i=0; i<nrRanduri; ++i)
```

```
    {
            for(j=0; j<nrColoana; ++j)
            {
                printf("%d  ", matrice[i][j]);
                if (j == nrColoana-1)
                        printf("\n\n");
            }
    }

    for(i=0; i<nrRanduri; ++i)
    {
            for(j=0; j<nrColoana; ++j)
            {
                transpusa[j][i] = matrice[i][j];
            }
    }

    printf("\nMatricea transpusa:\n");
    for(i=0; i<nrColoana; ++i)
    {
            for(j=0; j<nrRanduri; ++j)
            {
                printf("%d  ",transpusa[i][j]);
                if(j==nrRanduri-1)
                {
                        printf("\n\n");
                }
            }
    }

    return 0;
}
```

Să se parcurgă o matrice în spirală.

Exemplu: **Pentru 1 2 3**
4 5 6
7 8 9
10 11 12
se va afişa 1, 2, 3, 6, 9, 12, 11, 10, 7, 4, 5, 8.

Este destul de greu sa explic aceasta problema folosind doar text (problema este destul de vizuala). Recomand sa vizionati un video precum acesta:

https://www.youtube.com/watch?v=siKFOI8PNKM&feature=youtu.be

(macar partea in care discuta ideea in sine a algoritmului). *Am luat exemplul de aici si l-am rescris intr-un mod putin clar. In mare, problema trebuie impartita in 4 etape (mereu cand avem o problema mai complexa, vrem o spargem in bucati mai mici). In principal noi vom parcuge cele 4 laturi ale unei matrice, deci vom merge stanga->dreapta, sus->jos, dreapta->stanga, jos->sus. La fiecare tur in jurul matricei vom da la o parte un nivel din ea (adica un rand si coloana de pe fiecare parte.)*

Ne ocupam de acest lucru cu 4 indici: iRandStart iColStart iRandFinal iColFinal. For-urile

din program sunt destul de clare odata ce le vizualizati (ele fac pe rand, cele 4 parcugeri de care am vorbit). Dupa ce am parcurs stanga-> dreapta, trebuie sa incrementam indexul ce tine randul de start, dupa ce am parcurs sus->jos decrementam de coloane final, dreapta->stanga decrementam randul de final, jos->sus incrementam index de coloana de start.

Din nou, este destul de dificil de explicat in text. Va propun sa va luati o matrice 3x3 si sa scrieti pe hartie variabilele si afisarile la fiecare iteratie, dupa cum am facut la laborator cu cautarea binara.

```
#include <stdio.h>
#define RMAX 100
#define CMAX 100

void spiralPrint(int iRandFinal, int iColFinal, int a[RMAX][CMAX])
{
    int i, iRandStart = 0, iColStart = 0;

    while (iRandStart < iRandFinal && iColStart < iColFinal)
    {
    /* stanga-> dreapta */
    for (i = iColStart; i < iColFinal; ++i)
    {
        printf("%d ", a[iRandStart][i]);
    }
    iRandStart++;

    /* sus->jos */
    for (i = iRandStart; i < iRandFinal; ++i)
    {
        printf("%d ", a[i][iColFinal-1]);
    }
    iColFinal--;

    /* dreapta->stanga */
    if (iRandStart < iRandFinal)
    {
        for (i = iColFinal-1; i >= iColStart; --i)
        {
                printf("%d ", a[iRandFinal-1][i]);
        }
        iRandFinal--;
    }

    /* jos->sus */
    if (iColStart < iColFinal)
    {
        for (i = iRandFinal-1; i >= iRandStart; --i)
        {
                printf("%d ", a[i][iColStart]);
        }
        iColStart++;
    }
    }
}
```

```
int main()
{
    int matrice[RMAX][CMAX];
    int nrRanduri, nrColoana, i, j;
    printf("Numarul de randuri si de coloane: ");
    scanf("%d %d", &nrRanduri, &nrColoana);

    printf("\nElementele matricei:\n");
    for(i=0; i<nrRanduri; ++i)
    {
            for(j=0; j<nrColoana; ++j)
            {
                printf("Elementul %d %d: ",i+1, j+1);
                scanf("%d", &matrice[i][j]);
            }
    }

    printf("\nMatricea: \n");
    for(i=0; i<nrRanduri; ++i)
    {
            for(j=0; j<nrColoana; ++j)
            {
                printf("%d  ", matrice[i][j]);
                if (j == nrColoana-1)
                        printf("\n\n");
            }
    }

    spiralPrint(nrRanduri, nrColoana, matrice);
    return 0;
}
```

Nu le-am mai scris si pe 6,7,8. Sunt toate relativ mari si greoi de explicat in text. Daca le vrea cineva in mod explicit, le putem explica la inceputul sau finalul unui laborator. 9 si 10 sunt deja cam mult :D.