Solutiile laboratorului sa salvam in solutii_lab_3.c (in care vom avea si functia main())
Pentru definirea structurilor si constantelor, vom folosi un fisier header solutii_lab_3.h, pe care il vom include in solutii_lab_3.c

1. In Codul ASCII (standard pentru reprezentarea caracterelor in calculatoare), literele sunt reprezentate consecutiv (mai intai cele uppercase si apoi lowercase):
   'A' -> 65, 'B' -> 66, ... , 'Z' -> 90, … 'a' -> 97, 'b' -> 98, …, 'z' -> 122,
   deci un char este uppercase daca este cuprins intre 'A' si 'Z' (inclusiv)

2. Pentru citire putem folosi scanf("%d %c %d", …)

3. a. Atentie ca in C trebuie sa folosim:
   typedef struct {
   ….
   } Student;
   pentru a ne putea referi la struct-ul Student in mod direct. (Student x;)
   Altfel, daca definim ca in C++, adica:
   struct Student {
   ….
   };
   va trebui ca de fiecare data cand definim un Student (sau il pasam ca parametru) sa scriem '*struct Student*'
   b. , c. Vezi exemple in lab
   d. Pentru simplitate putem presupune ca numele este format dintr-un singur cuvant.
   Solutie posibila (ineficienta): Dupa fiecare citire a unui student nou, sortam vectorul.
   In C exista functia qsort() care poate fi folosita pentru a sorta un array de structs (vezi exemplu) cu ajutorul unei functii comparator. In cazul nostru, comparatorul va folosi strcmp() pentru a compara numele a doi studenti.

   Exemplu: **studenti.in**
   234 Nicolae 5.6 8.9 7.8
   890 Avram 9.1 7.0 5.6
   654 Popescu 4.3 3.0 5.4
   111 One-scu 8 8 8
   1024 Binarescu 10 10 10
   Output posibil:

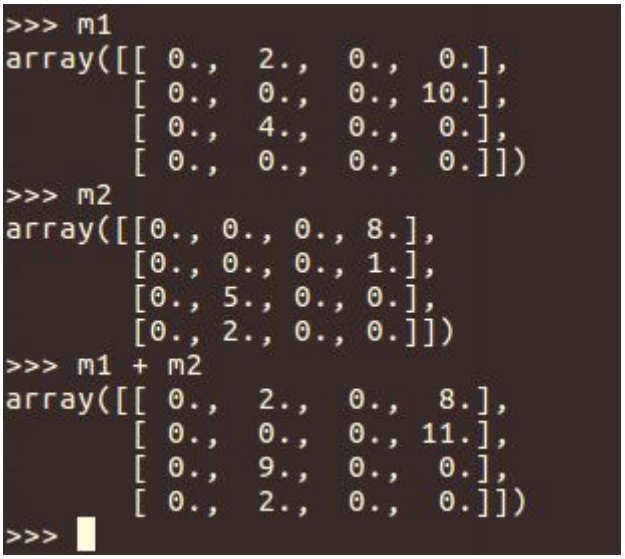
```
andreea@HP:lab03$ cat in
234 Nicolae 5.6 8.9 7.8
890 Avram 9.1 7.0 5.6
654 Popescu 4.3 3.0 5.4
111 One-scu 8 8 8
1024 Binarescu 10 10 10andreea@HP:lab03$
andreea@HP:lab03$ g++ solutii_lab_3.c
andreea@HP:lab03$ ./a.out < in
Legitimatie: 890, Nume: Avram, Nota mate: 9.10, Nota info 7.00, Nota bac: 5.60
Legitimatie: 1024, Nume: Binarescu, Nota mate: 10.00, Nota info 10.00, Nota bac: 10.00
Legitimatie: 234, Nume: Nicolae, Nota mate: 5.60, Nota info 8.90, Nota bac: 7.80
Legitimatie: 111, Nume: One-scu, Nota mate: 8.00, Nota info 8.00, Nota bac: 8.00
Legitimatie: 654, Nume: Popescu, Nota mate: 4.30, Nota info 3.00, Nota bac: 5.40
andreea@HP:lab03$
```

e. Putem crea un comparator nou pentru a ordona studentii in functie de media lor. Calculam cati studenti inseamna 75% din totalul lor, iar apoi parcurgem array-ul sortat (descrescator) si populam primele 75% dintre entry-uri cu buget = 1, iar celelalte cu buget = 0.

4. In loc sa tinem toata matricea mat[50000][50000], este suficient sa retinem un array cu informatii despre elementele nenule (Q: Ce informatii ? A: Ne intereseaza randul, coloana si valoarea elementelor nenule), deci este suficient sa declaram:
int mat[MAX_ELEMS][3];
cu seminificatia:
mat[i][0] = randul la care se gaseste al i-lea element nenul
mat[i][1] = coloana la care se gaseste al i-lea element nenul
mat[i][2] = valoarea celui de-al i-lea element nenul
In plus, avem nevoie de un contor care sa retina numarul curent de elemente nenule din matricea noastra.

Pentru adunare: Putem initializa o matrice rezultat cu una dintre matricele initiale, iar apoi iteram prin elementele celeilalte matrice si daca exista un entry pe pozitia respectiva in matricea rezultat, atunci adunam valoarea curenta la aceasta; altfel un nou entry in matricea rezultat si incrementam numarul de elemente.

Exmplu input:
3 4 // dimensiunea matricelor
3 // numar elemente nenule in prima matrice
0 1 2 // 3 x (linie, coloana, valoare) unde se afla elementele nenule
1 3 10
2 1 4
4 // numar elemente nenule in a doua matrice
0 3 8 // 4 x (linia, coloana, valoare) unde se agla elementele nenule
1 3 1
2 1 5
3 1 2



Cum ar trebui sa arate rezultatul ? (In formatul struct-ului ales de noi)

Pentru **inmultire**:
result[i][j] = suma_k(m1[i][k] * m2[k][j])

Pentru explicatii si tutorial (+ cod, dar nu in C, ci in Java):
https://www.geeksforgeeks.org/operations-sparse-matrices/