Programarea Calculatoarelor și Limbaje de Programare

Seria CC

(2021 - 2022)

Deadline: 10.12.2021, ora 23:55

Tema 1

Responsabili Tema:
Andreea Nica,
Gabriel Mocanu,
Bogdan Nutu

Colaboratori Tema:
Alex Deonise,
Daria Florea

Cuprins

Istoric modificari:

- v1.0, 23.11.2021 - Publicare enunt

1 Let's get started

In cadrul acestei prime teme de la *Programarea Calculatoarelor si Limbaje de Programare* veti avea ocazia sa lucrati cu diferite concepte studiate pana acum la curs sau in cadrul laboratorului.

Pe de o parte, veti avea nevoie de functiile de citire / scriere din limbajul C si de instructiunile sale de baza. Pe de alta parte, vi se vor testa si cunostintele legate de tablouri, siruri de caractere si functii.

Tema este compusa din patru probleme independente una de cealalta, punctate individual.

2 Problema I - Anagrame

Andrei a descoperit un nou joc interesant, cel al anagramelor.

Fie A un sir de caractere format doar din litere ale alfabetului englez. O **anagrama** a lui A este orice sir care contine exact aceleasi caractere ca A, cu acelasi numar de aparitii, dar intr-o ordine diferita.

De exemplu, doua anagrame posibile pentru cuvantul ***kick*** ar fi ***kcik*** sau ***ckki***.

In continuare, vom defini o marime cu ajutorul careia vom putea masura cat de "puternica" este o anagrama. **Gradul** unei anagrame este un numar real cuprins in intervalul [0, 1], care se refera la procentul de caractere care difera intre cele doua cuvinte raportat la numarul total de caractere. In cazul in care doua cuvinte nu sunt anagrame, le vom atribui gradul -1. Vom oferi cateva exemple pentru a clarifica aceasta definitie:

- Perechea ***kick*** - ***cick*** are gradul -1, deoarece cele doua cuvinte nu sunt anagrame.
- Perechea ***kick*** - ***kick*** are gradul 0. Conform definitiei, cele doua cuvinte sunt anagrame, insa toate cele 4 caractere sunt identice in ambele cuvinte. Asadar, gradul acestei perechi va fi $0/4 = 0$.
- Perechea ***kick*** - ***ckki*** are gradul 1. Comparand caracterele din cele doua cuvinte, vom observa ca toate caracterele de pe acelasi indice difera unul de celalalt. Astfel, gradul acestei perechi va fi $4/4 = 1$.
- Perechea ***kick*** - ***kikc*** are gradul 0.5. In cazul acestui exemplu, primele doua caractere sunt identice, dar ultimele doua difera. Asadar, gradul acestei anagrame va fi $2/4 = 0.5$.

Acestea fiind spuse, sarcina voastra va fi sa cititi de la tastatura o serie de **N** perechi de cuvinte, care pot fi anagrame sau nu, si sa le ordonati descrescator dupa gradul lor. In cazul in care mai multe perechi au acelasi grad, le veti ordona lexicografic tot descrescator, dupa primul cuvant al perechii.

Date intrare:

- Pe prima linie se afla **N**, numarul de perechi de cuvinte care urmeaza sa fie citite.
- Apoi se citesc **N linii**, fiecare din ele continand o pereche de doua cuvinte, separate prin spatiu.

Date iesire:

- Se vor afisa **N linii**, reprezentand cele **N perechi de cuvinte** citite, ordonate descrescator dupa grad.

Restrictii si precizari:

- $1 \leq N \leq 1000$.
- Cuvintele citite de la tastatura sunt formate doar din litere mici ale alfabetului englez.
- Lungimea oricarui cuvant citit nu va depasi 50 de caractere.
- Printre altele, trebuie sa regasim in programul vostru urmatoarele functii:
  - **int is_anagram(char *s1, char *s2)** - verifica daca doua siruri de caractere primite ca parametru sunt anagrame
  - **double compute_grade(char *s1, char *s2)** - calculeaza gradul a doua siruri de caractere primite ca parametru

Exemplu:

Input	Output	Explicatii
6		Gradele celor 6 perechi sunt, pe rand:
night thing	stressed desserts	1, 1, 0.66, 1, 0.5, 0.
stressed desserts	night thing	Avem astfel 3 perechi cu acelasi grad (1),
tar rat	arc car	pe care le vom ordona descrescator
arc car	tar rat	lexicografic dupa primul cuvant al perechii:
kick kick	kick kcik	(night, stressed, arc)
kick kcik	kick kick	Celelalte 3 perechi se vor afisa in ordinea gradului,
		tot descrescator.

3 Problema II - Sah

Dupa ce a invatat despre anagrame, Andrei s-a orientat catre un "sport" al mintii, sahul. El are o tabla speciala de sah, pe care exista doar nebuni, piese care se ataca reciproc daca impart aceeasi diagonala.

In continuare, vom considera ca tabla de sah a lui Andrei are dimensiune variabila, **MxM**. Pe aceasta se afla **N** nebuni, reprezentati ca perechi (rand, coloana).

In cadrul acestei probleme, veti avea de rezolvat urmatoarele doua cerinte:

1. Sa se scrie o functie care afiseaza numarul de perechi de nebuni care se ataca reciproc.
2. Daca tabla de sah contine o singura pereche de nebuni care se ataca, atunci sa se determine daca se poate muta unul dintre ei astfel incat sa nu mai existe nicio pereche de nebuni care sa se atace. Daca nu exista nicio posibilitate de mutare pentru niciunul dintre cei doi nebuni sau daca pe tabla nu exista doar o singura pereche de astfel de nebuni, se va aflsa "NU". Altfel, se va afisa "DA".

Date intrare:

- Pe prima linie se va afla **M**, dimensiunea tablei de sah.
- Pe a doua linie se afla **N**, numarul de nebuni de pe tabla.
- Apoi se citesc **N** linii de forma "x y", reprezentand coordonatele celor N nebuni de pe tabla de sah. Evident, $0 \leq$ xi, yi $<$ N.

Date iesire:

- Pe prima linie se va afisa numarul de perechi de nebuni care se ataca reciproc.
- Pe a doua linie se afla "DA" sau "NU", conform indicatiei de mai sus.

Exemplul 1:

Input	Output	Explicatii
M = 5 N = 4 0 0 1 2 2 2 4 0	2 NU	Tabla va arata in felul urmator: 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 Exista doua perechi de nebuni care se ataca: (1, 3) si (3, 4) Pentru cerinta 2, tabla nu indeplineste restrictia (o singura pereche care se ataca), deci se va afisa "NU".

Exemplul 2:

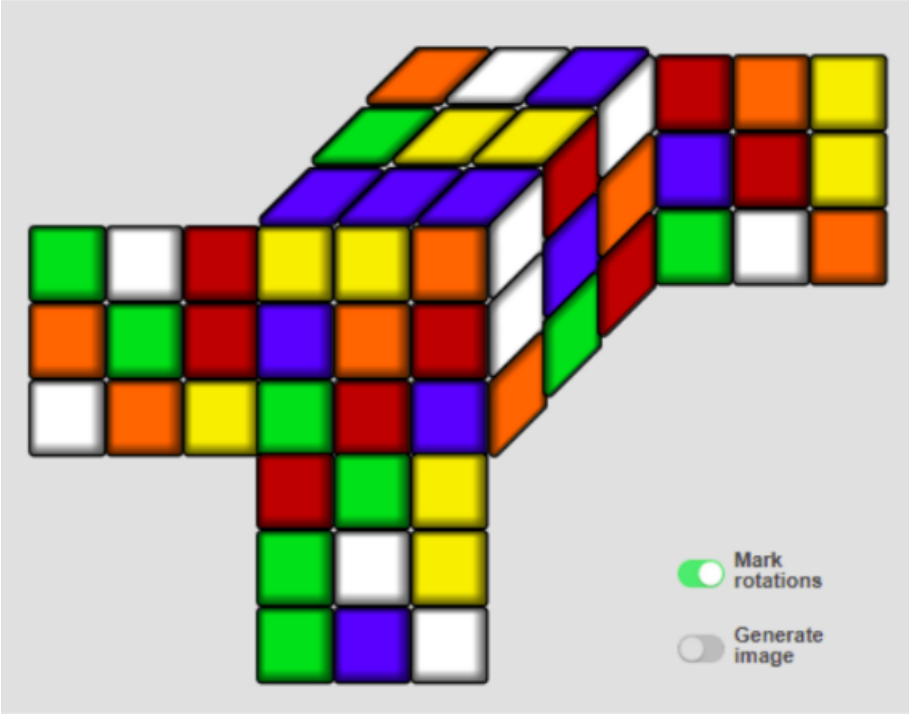
Input	Output	Explicatii
M = 5 N = 4 0 0 1 2 2 2 2 4	1 DA	Tabla va arata in felul urmator: 1 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 Exista o pereche de nebuni care se ataca: (1, 3) Pentru cerinta 2, o posibila solutie ar fi: 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 Am mutat practic nebunul 3 de pe pozitia (2, 2) pe pozitia (3, 1).

4 Problema III - Rubik's Cube

Andrei si-a gasit o noua pasiune, cubul Rubik, si va propune spre rezolvare o problema prin care el doreste sa observe daca mutarile pe care el le-a ales conduc la rezolvarea cubului. Pentru inceput, Andrei doreste sa va familiarizati cu formatul cubului sau Robik, de dimensiune 3x3x3. El va va transmite initial cele 6 culori care vor fi folosite pentru fetele cubului, urmate de 6 tablouri ce caracterizeaza cele 6 fete, ordinea lor fiind: front, back, up, down, left, right. Fiecare fata a cubului va fi formata din 3 linii, pe fiecare linie fiind cate 3 culori separate de caracterul '.'. De exemplu, pentru culorile rosu, galben, albastru, verde, portocaliu, alb (numite in continuare ros, gal, albs, vs, port si alb), cele 6 fete pot fi codificate astfel:

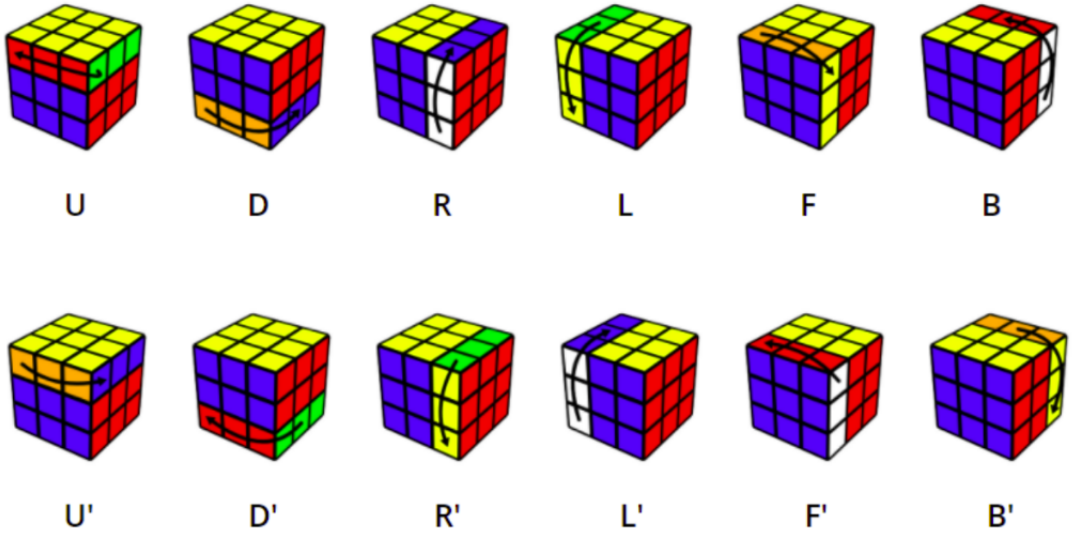
Fata 1 (Front)	Fata 2 (Back)	Fata 3 (Up)	Fata 4 (Down)	Fata 5 (Left)	Fata 6 (Right)
gal.gal.port	ros.port.gal	port.alb.albs	ros.vd.gal	vd.alb.ros	alb.ros.alb
albs.port.ros	albs.ros.gal	vd.gal.gal	vd.alb.gal	port.vd.ros	alb.albs.port
vd.ros.albs	vd.alb.port	albs.albs.albs	vd.albs.alb	alb.port.gal	port.vs.ros

Codificarea anterioara caracterizeaza cubul din imaginea urmatoare:



Andrei s-a apucat de treaba si a invatat care sunt mutarile specifice rezolvarii cubului Rubik 3x3x3. El stie urmatoarele 12 mutari:

- **F** - front (fata), rotatie in sensul acelor de ceasornic
- **F'** - front (fata), rotatie in sens trigonometric (invers acelor de ceasornic)
- **B** - back (spate), rotatie in sensul acelor de ceasornic
- **B'** - back (spate), rotatie in sens trigonometric (invers acelor de ceasornic)
- **U** - up (sus), rotatie in sensul acelor de ceasornic
- **U'** - up (sus), rotatie in sens trigonometric (invers acelor de ceasornic)
- **D** - down (jos), rotatie in sensul acelor de ceasornic
- **D'** - down (jos), rotatie in sens trigonometric (invers acelor de ceasornic)
- **R** - right (dreapta), rotatie in sensul acelor de ceasornic
- **R'** - right (dreapta), rotatie in sens trigonometric (invers acelor de ceasornic)
- **L** - left (stanga), rotatie in sensul acelor de ceasornic
- **L'** - left (stanga), rotatie in sens trigonometric (invers acelor de ceasornic)



Andrei incearca sa isi indeplineasca visul de a invata sa rezolve cubul Rubik. El va va transmite configuratia initiala a cubului (cele 6 culori, respectiv cele 6 fete), o variabila **m**, reprezentand numarul de mutari si m mutari (din cele 12 prezentate anterior), iar voi trebuie sa afisati configuratia cubului dupa cele m mutari efectuate.

Date intrare:

- O linie pe care se afla cele 6 culori folosite, separate prin spatiu.
- 18 linii, reprezentand configuratiile celor 6 fete, in ordinea descrisa anterior. Fiecare linie este reprezentata de un sir de caractere format din 3 dintre culorile de pe prima linie, separate de caracterul '.' (punct).
- m - numarul de mutari.
- Pe ultima linie se afla cele m mutari care se vor efectua, separate prin spatiu.

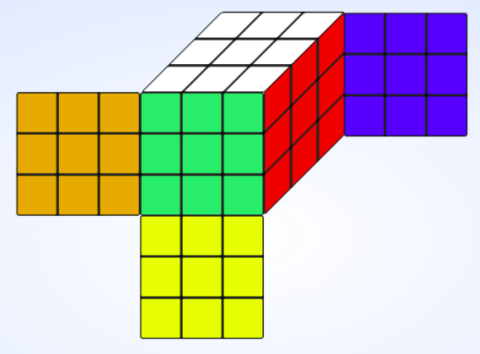
Date iesire: Configuratia finala a cubului, respectand formatul folosit si la citire.

Restrictii si precizari:

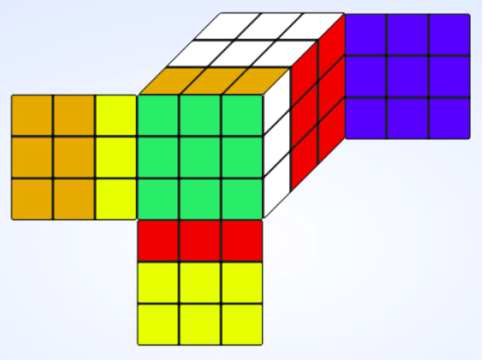
- Lungimea oricarei culori codificate nu va depasi 10 caractere.
- $1 \leq m \leq 100$.

Exemplu:

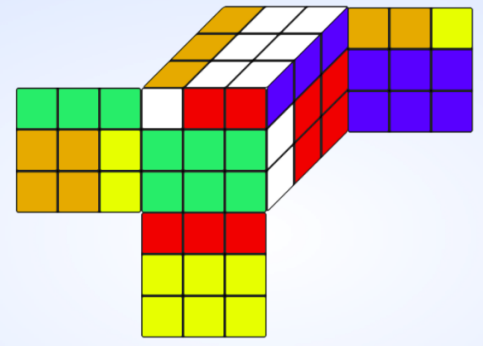
Input	Output	Explicatii
ros gal albs vd port alb vd.vd.vd vd.vd.vd vd.vd.vd albs.albs.albs albs.albs.albs albs.albs.albs alb.alb.alb alb.alb.alb alb.alb.alb gal.gal.gal gal.gal.gal gal.gal.gal port.port.port port.port.port port.port.port ros.ros.ros ros.ros.ros ros.ros.ros 3 F U L'	ros.ros.ros gal.vd.vd gal.vd.vd port.port.port albs.albs.port albs.albs.port alb.alb.alb vd.alb.alb vd.alb.alb albs.ros.ros albs.gal.gal gal.gal.gal vd.gal.gal vd.port.port vd.port.port albs.albs.albs alb.ros.ros alb.ros.ros	Codurile celor 6 culori sunt ros, gal, albs, vd, port, alb. In imaginile urmatoare este prezentat configuratia initiala a cubului, respectiv configuratiile sale dupa fiecare mutare efectuata.



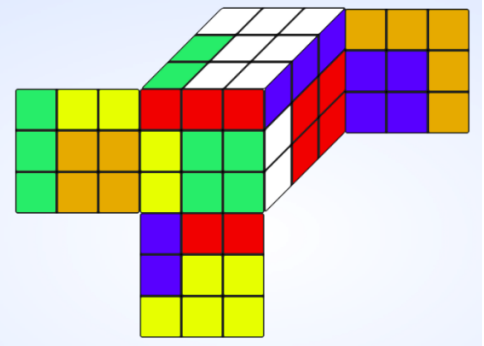
(a) Initial



(b) Dupa mutarea F



(c) Dupa mutarea U



(d) Dupa mutarea L'

5 Problema IV - X si 0

Andrei a invatat de curand logica booleana si operatiile cu matrice, asa ca se gandeste la o metoda prin care poate invata aceste notiuni matematice mai usor printr-un joc de X si 0 modificat.
Matricele pe care le va folosi Andrei in calcul sunt patratice, de dimensiune 8x8. Pentru a reprezanta aceste matrice Andrei se va folosi de anumite numere reprezentate pe 8 biti.
Spre exemplu, sa spunem ca Andrei vrea sa contruiasca o matrice de 8x8 cu ajutorul a 8 numere ce pot fi reprezentate pe 8 biti: 146, 66, 98, 30, 227, 130, 254, 136. El va converti fiecare numar in baza 2 si va umple fiecare linie din matrice succesiv.
In final ar trebui sa obtina:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Fiecare linie reprezinta, in ordine, cate un numar din cele 8 in binar:
146 -> 10010010
66 -> 01000010 etc.

Folosind aceasta matrice, el va genera alte doua pe care le va analiza ulterior pentru a putea observa care dintre ele are un scor mai mare in cadrul jocului X si 0. Notand cu **A** aceasta matrice, el va mai calcula $\boldsymbol{A * A^t}$ si $\boldsymbol{A^2}$ folosind operatiile cu matrice, dar folosind logica booleana.

Operatii in logica booleana:

Inmultire:
$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 1 = 1$
$1 \times 0 = 0$

Adunare:
$1 + 0 = 1$
$1 + 1 = 1$
$0 + 0 = 0$
$0 + 1 = 1$

Asadar folosind matricea **A** obtinuta anterior vom face calculul $\boldsymbol{A} * \boldsymbol{A^t}$ unde $\boldsymbol{A^t}$ este transpusa matricei **A** si vom obtine:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Iar pentru $\boldsymbol{A^2}$ vom obtine:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Acum pentru fiecare matrice in parte Andrei va calcula scorul X si 0 impartind matricea in 4 parti egale si aplicand aceleasi reguli ca la X si 0, fiecare linie completa sau fiecare coloana completa sau fiecare diagonala completa cu 1 in fiecare dintre cele 4 matrice se noteaza cu 1 punct.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

De exemplu, pentru matricea $\boldsymbol{A^2}$ calculata anterior vom avea 4 matrice impartite in mod egal asa cum este reprezentat in figura de mai sus.

Pentru matricea rosie vom avea:

- 4 linii
- 4 coloane
- 2 diagonale

Un total de 10 puncte.

Pentru matricea albastra vom avea:

- 1 linie
- 3 coloane
- 1 diagonala

Un total de 5 puncte.

Pentru matricea galbena vom avea:

- 4 linii
- 4 coloane
- 2 diagonale

Un total de 10 puncte.

Pentru matricea verde vom avea:

- 1 linie
- 1 coloana
- 1 diagonala

Un total de 3 puncte.

In total de 28 de puncte pentru aceasta varianta. Ulterior Andrei calculeaza si scorurile pentru matricea **A** pentru care va obtine scorul 5 si pentru matricea $\boldsymbol{A * A^t}$ pentru care va obtine scorul de 34 de puncte. Asadar varianta castigatoare este $\boldsymbol{A * A^t}$. In final Andrei trebuie sa afiseze pe cate o linie numerele in format zecimal de pe fiecare linie din matricea castigatoare, $\boldsymbol{A * A^t}$. In cazul in care mai multe variante sunt la egalitate castigatoare, se va afisa prima varianta in ordinea calcularii lor.(**A**, $\boldsymbol{A * A^t}$, $\boldsymbol{A^2}$).

Date de intrare:
Pe primele 8 linii se afla numerele ce urmeaza sa fie citite si transformate in matricea de biti.

Date de iesire:
Se vor afisa pe cate o linie separata, numerele de pe fiecare linie din matricea castigatore in format zecimal.

Restrictii si precizari:

- Trebuie sa regasim in programul vostru urmatoarele functii:
  - **void dec_to_bin (int n, char *s)** - transforma numarul n din baza 10 in baza 2 si il intoarce ca sir de caractere (s) de '0' sau '1'.
  - **int bin_to_dec(char *s)** - primeste un numar in baza 2 ca sir de caractere si intoarce valoarea in baza 10.
  - O functie pentru efectuarea inmultirii a doua matrice. Implementarea este la alegerea voastra.
  - O functie pentru efectuarea transpusei unei matrice. Implementarea este la alegerea voastra.
  - **int sum_X_0(int m[][8])** - functie pentru calculul scorului X si 0 pentru o matricea data.

Exemplu:

Input	Output
146	255
66	254
98	254
30	255
227	255
130	255
254	255
136	159

Explicatii
Tinand cont de faptul ca matricea $\boldsymbol{A * A^t}$ a avut cel mai mare scor atunci vom transforma numerele de pe fiecare linie din aceasta in format zecimal si le vom afisa pe aceeasi linie.

6 Restrictii si precizari

Separati logica programelor in mai multe functii, asa cum vi s-a cerut in enunturi! Nu se vor puncta sursele in care tot programul este scris in main!

Toate citirile se fac de la tastatura. Nu folositi fisiere. Datele de intrare sunt descrise pentru fiecare cerinta individual si vor fi citite de la tastatura. Pentru a nu fi nevoiti sa scrieti de fiecare data inputul, puteti redirecta un fisier in care sa scrieti datele pe care de obicei le introduceati de la tastatura, si sa rulati folosind formatul urmator:

./nume_executabil < numele_fisierului_de_intrare

Este interzisă folosirea variabilelor globale.

Soluția temei va fi scrisă în C. Nu folosiți sintaxă sau instrucțiuni specifice limbajului C++.

Fiecare problema va fi scrisa intr-un fisier sursa separat, numit problemaX.c, X = {1,2,3,4}.

Fișierul Makefile va conține doua reguli:**build** (implicit) și **clean**. Regula build va genera cele 4 executabile, iar clean va șterge toate fișierele create de compilare și execuție.

In **README** precizați cât timp v-a luat implementarea cerintelor și explicați, pe scurt, implementarea temei (comentariile din cod vor documenta mai amănunțit rezolvarea). Este recomandat ca liniile de cod și cele din fișierul README să nu depășească 80 de caractere

Folosiți un coding style astfel încât codul să fie ușor de citit și înțeles. De exemplu:

- Dați nume corespunzătoare variabilelor și funcțiilor.

- Nu adăugați prea multe linii libere sau alte spații goale unde nu este necesar (exemplu: nu terminați liniile în spații libere, trailing whitespaces; nu adăugați prea multe linii libere între instrucțiuni sau la sfârșitul fișierului). Principalul scop al spațiilor este identarea.

- Fiți consecvenți in coding style-ul ales. Va recomandam sa parcurgeti aceasta resursa: `https://ocw.cs.pub.ro/courses/programare-cc/coding_style`

- Există programe sau extensii pentru editoare text care vă pot formata codul. Deși vă pot ajuta destul de mult, ar fi ideal să încercați să respectați coding style-ul pe măsură ce scrieți codul.

Temele sunt strict individuale. Copierea temelor va fi sancționată. Persoanele cu porțiuni de cod identice nu vor primi niciun punctaj pe temă. Temele trimise după deadline nu vor fi luate în considerare.

Nu copiati cod de pe Internet! Se poate ajunge la situatia in care doi studenti sa aiba acelasi cod, preluat de pe Internet, caz in care ambele teme vor fi depunctate, desi studentii nu au colaborat direct intre ei.

Veţi trimite o arhivă **ZIP** cu numele de tipul **GRUPA_Nume_Prenume_Tema1.zip**. De exemplu: **311CC_Popescu_Maria_Tema1.zip**, care va conţine fișierele necesare, Makefile și README. Fisierele trebuie să fie în rădăcina arhivei, nu în alte subdirectoare.

Pentru intrebari despre tema se va folosi in mod exclusiv forumul temei, pe care va recomandam sa il vizitati chiar si daca nu aveti intrebari, intrucat este posibil sa aflati informatii noi din intrebarile puse de colegii vostri, respectiv din raspunsurile date de noi.

Compilarea nu ar trebui să producă avertizări (verificați prin adăugarea flagului -Wall la gcc).

In cazul in care ati alocat dinamic memorie (desi nu va impunem asta), trebuie sa o eliberati. Folosiți comanda de mai jos pentru a verifica dacă memoria este eliberată corect. Compilați programul cu -g pentru a vedea liniile unde aveti variabilele ce nu au fost eliberate. *valgrind –tool=memcheck –leak-check=full ./problemaX*

Temele trebuie să fie încărcate pe **vmchecker**. **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.

Temele vor fi prezentate asistentilor in cadrul primelor doua laboratoare de dupa deadline.

7 Notare

Total - 200p

Problema 1 - 40p

Problema 2 - 40p

Problema 3 - 50p

Problema 4 - 50p

Coding Style, Makefile, README - 20p

- 15p - Coding Style
  - Comentarii
  - Logică clară
  - Spațiere corectă
  - Stil consecvent
  - Variabile și funcții denumite adecvat
- 5p - Completarea fișierului README