PROGRAMAREA CALCULATOARELOR ȘI LIMBAJE DE PROGRAMARE 1

- TEMA 1 -

Profesor coordonator

Carmen ODUBĂȘTEANU

Responsabili

Costin-Alexandru DEONISE

Farhad Ali Irinel GUL

George-Alexandru TUDOR

Eduard-Andrei RADU

Termen de predare

Vineri, 09.12.2022, ora 23:59 (hard)

Ultima actualizare

Miercuri, 30.11.2022, ora 22:53

2022 - 2023

CUPRINS

1. PROBLEMA CUTIILOR

1.1. Enunț

În cadrul unei închisori sunt P prizonieri. Aceasta este pe cale să se închidă, așa că directorul închisorii le propune prizonierilor un joc.

Într-o cameră vor fi plasate, într-o ordine aleatorie, P cutii numerotate de la 1 la P. Fiecare cutie conține un alt număr de la 1 la P, posibil diferit față de numărul de pe cutie. Prizonierii vor intra, pe rând, în cameră și vor trebui să găsească acea cutie care conține (în interiorul ei) numărul lor propriu. Fiecare deținut va avea voie să deschidă doar P/2 cutii. Dacă toți prizonierii își vor găsi numărul corespunzător, vor fi eliberați. Este suficient ca doar unul să nu-și găsească numărul pentru ca toți să piarda.

Întrucât ideea alegerii la întâmplare a celor P/2 cutii de către fiecare deținut nu este una rentabilă pentru un număr mare de prizonieri, unul dintre ei a venit cu o soluție care le oferă tuturor o șansă mai mare de reușită.

Strategia lor este următoarea: fiecare prizonier deschide prima dată cutia corespunzătoare numărului acestuia. Adică, deținutul Y deschide cutia Y. În această cutie se află un număr Z. El continuă să deschidă cutia Z care, la rândul ei, conține un alt număr T și o va deschide și pe aceasta, repetând pașii până când își găsește numărul sau până la deschiderea numărului maxim de cutii, P/2. Astfel, posibilitatea de a ajunge la cutia care conține numărul său este mult mai mare. Fiecare prizonier își va crea o listă cu toate cutiile pe care le-a deschis până a ajuns la final (fie a găsit numărul său de ordine, fie a deschis P/2 cutii). Prizonierii vor intra pe rând, începând cu numărul 1, urmat de 2, până la P.

Rolul vostru este cel de director și trebuie să fiți atenți la desfășurarea jocului pentru ca la final să afișați:

- „Da” dacă prizonierii câștigă jocul sau „Nu” dacă nu câștigă jocul;
- Toate ciclurile formate de cutii, pe câte o linie nouă. Cutiile formează un ciclu dacă, plecând de la o cutie, se ajunge la aceeași cutie. Aceste liste de cutii vor fi afișate în ordine crescătoare: prima oară ciclul care începe din cutia 1, urmat de ciclul care începe din cutia imediat următoare (care nu face parte din ciclurile anterioare) și așa mai departe. Atenție! Vorbim despre numărul cutiei, nu despre conținutul acesteia.

1.2. Date de intrare

- Pe prima linie va fi numărul P de prizonieri
- P numere întregi, separate printr-un spațiu. Primul număr reprezintă conținutul cutiei 1, al doilea număr este conținutul cutiei 2 și așa mai departe.

1.3. Date de ieșire

- Pentru prima cerință se va afișa pe prima linie doar **Da** sau **Nu.**
- Pentru cerința numărul 2, toate ciclurile.

1.4. Restricții și precizări

- $2 \leq P \leq 500$;
- O implementare realizata integral în funcția main() nu va fi punctată! Implementati functii auxiliare!
- Nu se folosesc varaibile globale!

1.5. Testare și punctare

- Punctajul maxim este de 20 puncte.
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **cutii.c**

1.6. Exemple

1.6.1. Exemplul 1

Exemplul 1		
in	out	Explicație
6 3 6 2 1 4 5	Nu 1 3 2 6 5 4	Prizonierul 1 deschide cutia de pe poziția 1. Aceasta conține numărul 3, deci următoarea cutie deschisă de prizonierul 1 va fi cutia de pe poziția 3, care conține numărul 2. Va deschide și cutia de pe a doua poziție, care conține numărul 6. Cum prizonierul și-a epuizat numărul de cutii pe care are voie să le deschidă și nu și-a găsit numărul, deținuții vor pierde jocul. De asemenea, ciclul nu s-a încheiat (nu s-a ajuns de la cutia de la care a plecat) și, urmând aceeași regulă, se descoperă că toate cutiile fac parte din același ciclu.

1.6.2. Exemplul 2

Exemplul 2		
in	out	Explicație
10 4 5 2 7 1 8 3 6 10 9	Nu 1 4 7 3 2 5 6 8 9 10	Au voie să deschidă maxim 5 cutii. Prizonierul 1: 1→ 4→ 7→ 3→ 2 (nu și-a găsit numărul, deci vor pierde jocul). Ciclurile sunt: 1→ 4→ 7→ 3→ 2→ 5, din 5 se ajunge la cutia 1 6→ 8, din 8 se ajunge la 6 9→ 10, din 10 se ajunge la 9

1.6.3. Exemplul 3

Exemplul 3		
in	out	Explicație
10 4 5 1 7 8 9 3 2 10 6	Da 1 4 7 3 2 5 8 6 9 10	Au voie să deschidă maxim 5 cutii. Prizonierul 1: 1→ 4→ 7→ 3→ 1 (cutia 3 conține numărul de ordine al prizonierului 1). Prizonierul 2: 2→ 5→ 8 → 2 (cutia 8 conține numărul de ordine al prizonierului 2). ... Prizonierul 10: 10→ 6→ 9→ 10. Astfel, toți prizonierii își găsesc numărul. Ciclurile sunt: 1→ 4→ 7→ 3, din 3 se ajunge la cutia 1 2→ 5→ 8, din 8 se ajunge la 2 6 → 9→ 10, din 10 se ajunge la 6

2. NUMERE COMPLEXE LUNGI

2.1. Enunț

Să se scrie un program în care se implementează funcționalitatea unui calculator pentru numere complexe lungi (care pot fi mai mari decât suportă long long).

Operațiile care trebuie realizate sunt ***adunarea și scăderea.***

Se va citi de la început N, care reprezintă dimensiunea numerelor pentru toate calculele din sesiunea curentă. Numărul N va fi întotdeauna impar.

Se va ține cont de următoarele precizări privind citirea numărului complex lung, care va fi stocat intr-un șir de N caractere:

- numărul este împărțit în 2 părți: partea reală și cea imaginară;
- bitul de semn pentru partea reală va fi bitul 0, aka a[0];
- numărul corespunzător părții reale va fi cel format prin concatenarea tuturor caracterelor (care reprezinta cifre) începând cu bitul 1 până la bitul $\frac{n-1}{2} - 1$;
- bitul de semn pentru partea imaginară va fi bitul $\frac{n-1}{2}$;
- numărul corespunzător părții imaginare va fi cel format prin concatenarea tuturor caracterelor (care reprezinta cifre) începând cu bitul $\frac{n-1}{2} + 1$ până la bitul n – 1;

Folosind pe tot parcursul programului doi vectori a și b, puteți să suprascrieți rezultatul operației dintre a și b în vectorul a.

2.2. Date de intrare

- Pe prima linie se va citi numărul N.
- Pe a doua linie se va citi numărul a.
- Pe următoarele linii se citesc în ordine tipul operației și noul număr b, pana la întâlnirea ‘0’.

2.3. Date de ieșire

- Se va afișa rezultatul după toate operațiile.

2.4. Restricții și precizări

- Pentru fiecare număr se va folosi un vector de maxim 1001 de char-uri.
- Funcțiile de adunare, scădere și afișare vor primi ca parametri doar vectori și lungimea acestora.
- O implementare realizata integral în funcția main() nu va fi punctată! Implementati functii auxiliare! Nu se folosesc varaibile globale!
- În cazul în care rezultatul trece peste limita stocării, rezultatul va fi trunchiat la cele mai puțin semnificative cifre. Exemplu: 002098 + 003005 = 005003 (se va ignora 1 din 103, păstrându-se doar 03).
- Exemplu de rulare:
    1. Se citește n, dimensiunea vectorilor care va fi folosită pe tot parcursul programului.
    2. Se introduce primul număr.
    3. Se introduce tipul operației care va fi efectuată între primul număr și cel care urmează să fie citit („+” pentru adunare și „-” pentru scădere).
    4. Se introduce următorul număr.
    5. Se afișează rezultatul.
    6. Se repetă procedeul de la pasul 3, cu precizarea că programul se va opri atunci când în loc de operație, se va citi cifra „0”.

2.5. Testare și punctare

- Punctajul maxim este de 30 puncte.
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **lungi.c**

2.6. Exemple

2.6.1. Exemplul 1

Exemplul 1		
in	out	Explicație
7 001045 + 014034 - 012113 0	015079 003092	7 dimensiunea vectorilor 001045 primul număr (1 + 45i) + 014034 al doilea număr (14 + 34i) 015079 rezultatul sumei, număr care va fi luat în calcul pentru următoarele operații (și va fi afișat) - 012113 următorul număr (12 – 13i) 003092 rezultatul (va fi afișat și acesta) 0 finalizarea programului Colorat cu roșu = biții de semn Colorat cu alb = numerele

2.6.2. Exemplul 2

Exemplul 2		
in	out	Explicație
9 14571567 + 05040232 0	00471335	9 dimensiunea vectorilor 14571567 primul număr (-457 - 567i) + 05040232 al doilea număr (504 + 232i) 00471335 rezultatul sumei (47 – 335i) 0 finalizarea programului Colorat cu roșu = biții de semn Colorat cu alb = numerele

3. TRANSMISIE BRUIATĂ

3.1. Enunț

Ana și Bob se află în orașe diferite. Amândoi își schimbă poziția constant, neajungând niciodată în același oraș simultan, astfel că doresc să comunice prin mesaje. De la un oraș la altul există o antenă care transmite acest mesaj. Problema este că toate antenele din această rețea criptează mesajul transmis. Dacă Ana trimite mesajul, Bob va primi o formă alterată a acestuia în funcție de distanța dintre cele doua orașe. Fiecare antena are un mod diferit de a codifica. Există doar 2 moduri de codificare, iar acestea permit decodificarea mesajului inițial, deoarece tot mesajul poate fi reprodus prin operația inversă. Operatiile sunt:

- Codificarea A - Columnar Transposition Chipher

  Aceasta codificare necesită o cheie pe lângă textul de intrare. Folosim o matrice cu un număr egal de coloane cu lungimea cheii și cate rânduri este nevoie pentru a acoperi tot textul de intrare. Textul va fi scris linie cu linie in matrice. Se vor citi coloanele succesiv in ordinea valorilor ASCII ale caracterelor din cheie.

- Codificarea B

  Textul se va muta la dreapta cu un număr de caractere date.

  "Ana are mere", 5 -> " mereAna are"

  De asemenea, caracterele mutate vor fi modificate prin adăugarea numărului de caractere la codul ASCII al fiecăruia.

  " mereAna are" -> " rjwjAna are"

  Creșterea/Scăderea codului ASCII se va face doar în raza de coduri ASCII a caracterelor alfabetului. Dacă se ajunge la caracterul z și mai sunt 2 caractere de crescut, rezultatul va fi B (de la z la A și de la A la B). Dacă se ajunge la caracterul Z și mai sunt 2 caractere de crescut, rezultatul va fi b. Dacă trebuie scăzut de la A cu 2 caractere, rezultatul va fi y (de la A la z și de la z la y). Dacă trebuie scăzut de la a 2 caractere, rezultatul va fi Y.

  Prin rotire, spatiile ramân tot spații! Vezi exemplele 3, 4, 5.

- Decodificarea A

  Presupune inversarea pasilor de la codificarea A.

- Decodificarea B

  Presupune inversarea pasilor de la codificarea B.

3.2. Date de intrare

- Pe prima linie se va citi textul inițial.
- Pe următoarele linii se vor citi succesiv numele operațiilor pe care vom aplica (CodificareA, CodificareB, DecodificareA, DecodificareB), urmată de un spațiu și argumentul codificării (char* pentru A și int pentru B).
- Citirea se va opri cand se va tasta STOP.

3.3. Date de ieșire

Programul va afișa după fiecare operație rezultatul operației (codificării / decodificării).

3.4. Restricții și precizări

- Textul ce se va cripta are lungime maximă de 500 caractere
- Cheia cu care se realizeaza criptarea A are lungime maximă de 50 caractere
- Cheia cu care se realizeaza criptarea B are valoarea maxima egala cu lungimea textului
- Fiecare codificare si decodificare se va implementa intr-o functie separata
- O implementare realizata integral în funcția main() nu va fi punctată! Implementati functii auxiliare! Nu se folosesc varaibile globale!
- Codificarile si decodificarile se vor apela din functia main conform modului de executie
- Ordonarea cheii se va face prin algoritmul de sequential sort

3.5. Testare și notare

- Punctajul maxim este de 40 puncte.
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **codificari.c**

3.6. Exemple

Pentru o mai bună înțelegere, output-ul va fi subliniat (pentru a pune în evidență unde vor fi spațiile). Nu trebuie să subliniați în output-ul vostru!

3.6.1. Exemplul 1

Exemplul 1		
in	out	Explicație
Eu pup poala popii popa pupa poala mea CodificareB 7 STOP	hsh tlhEu pup poala ppii popa pupa po	Vezi exemplul 4

3.6.2. Exemplul 2

<table>
<tr><th colspan="3">Exemplul 2</th></tr>
<tr><td>in</td><td>out</td><td>Explicație</td></tr>
<tr><td>Ana are mere mari
CodificareA merele
STOP</td><td>n m erre ariAe ama</td><td>Matricea:
<table><tr><th>m</th><th>e</th><th>r</th><th>e</th><th>l</th><th>e</th></tr><tr><td>A</td><td>n</td><td>a</td><td></td><td>a</td><td>r</td></tr><tr><td>e</td><td></td><td>m</td><td>e</td><td>r</td><td>e</td></tr><tr><td></td><td>m</td><td>a</td><td>r</td><td>i</td><td></td></tr></table>
Ordinea parcurgerii:
<table><tr><th>e</th><th>e</th><th>e</th><th>l</th><th>m</th><th>r</th></tr><tr><td>n</td><td></td><td>r</td><td>a</td><td>A</td><td>a</td></tr><tr><td></td><td>e</td><td>e</td><td>r</td><td>e</td><td>m</td></tr><tr><td>m</td><td>r</td><td></td><td>i</td><td></td><td>a</td></tr></table>
Output: n m erre ariAe ama

Nu trebuie interschimbate coloanele matricei ci doar considerata ordinea!</td></tr>
</table>

3.6.3. Exemplul 3

<table>
<tr><th colspan="3">Exemplul 3</th></tr>
<tr><td>in</td><td>out</td><td>Explicație</td></tr>
<tr><td>Eu pup poala popii popa pupa poala mea
CodificareA cadelnita
STOP</td><td>ulp aoipm Ea ae aop p po ppa upaa po l piu</td><td>Vezi exemplul 2</td></tr>
</table>

3.6.4. Exemplul 4

Exemplul 4		
in	out	Explicație
Ana are mere CodificareB 5 STOP	rjwjAna are	În urma apelului codificării B, ultimele 5 caractere din șir au fost mutate la început rezultând " mereAna are" și apoi, celor 5 caractere mutate li s-a adăugat valoarea 5, șirul devenind " rjwjAna are". Caracterul ' ' (spațiu) nu a fost modificat.

3.6.5. Exemplul 5

Exemplul 5		
in	out	Explicație
Un vultur sta pe pisc cu un pix in plisc CodificareB 5 STOP	uqnxhUn vultur sta pe pisc cu un pix in	Vezi exemplul 4

3.6.6. Exemplul 6

Exemplul 6		
in	out	Explicație
Ana are mere CodificareA Bob CodificareB 6 DecodificareB 6 DecodificareA Bob STOP	“A eearmena r“ “sktg xA eear” “A eearmena r“ “Ana are mere“	Ghilimelele nu vor apărea in fișierul de output! Le-am pus in acest exemplu pentru a pun in evidenta spatiile de la final!

4. CAR DEALER

4.1. Enunț

Ați fost aleși de un car dealer pentru a organiza eficient mașinile și consumul acestora de combustibil, în cadrul unui proiect de cercetare pentru creșterea eficienței vehiculelor și scăderea poluării. O mașină va fi reprezentată prin urmatoarele date:

char *brand; - șir de caractere ce reprezintă marca mașinii

char *numar; - șir de caractere ce reprezintă numărul de înmatriculare

char *combustibil; - poate fi “benzina”, “motorina”, “hibrid”, “electric”

double consum; - număr real ce reprezintă consumul mediu al mașinii

int km; - numărul de kilometri parcurși de acea mașină

Pentru fiecare tip de mașină se poate calcula consumul total astfel:

(double)((consum * km) / 100).

Estimăm prețul benzinei ca fiind de 8.02 lei/L, iar cel al motorinei de 9.29 lei/L. Se consideră că mașinile hibride funcționează cu benzină.

Va trebui să rezolvați următoarele cerințe, fiecare într-o funcție separată:

a) O statistică a numărului de mașini de fiecare tip (după combustibil), afișată (în *main*) în formatul:

<combustibil> – <nr_mașini_combustibil>

b) Calculați consumul total de combustibil (double) si suma totală plătită (double) de dealership pentru numărul de kilometri rulați de fiecare marcă de mașină din cele existente în stoc. Afișati rezultatul cu două zecimale, în funcția *main*, în formatul:

<Brand> a consumat <nr_total_litri> – <suma_lei> lei

c) Să se determine care numere de înmatriculare nu sunt corecte. Dacă toate numerele sunt corecte se afișează în funcția *main*: “Numere corecte!”. Un număr de înmatriculare corect este format din una sau două litere, urmate de două sau trei cifre, urmate de exact trei litere. Pentru fiecare număr incorect se va afișa (în *main*):

<Brand> cu numarul <nr_înmatriculare>: numar invalid

4.2. Date de intrare

- Pe prima linie se află N, numărul de mașini.
- Urmează N linii ce conțin datele pentru fiecare mașină. Pe fiecare linie se află detaliile despre o singură mașină, informațiile fiind separate printr-un spațiu.
- Ultima linie conține unul dintre caracterele ‘a’, ‘b’ sau ‘c’ ce reprezintă cerința care va fi rezolvată.

4.3. Date de ieșire

- Se va afișa rezultatul operației specificate, așa cum este descris în cerință.

4.4. Restricții și precizări

- $1 \leq N \leq 100$
- Datele mașinilor vor fi memorate în vector/vectori alocat/alocați dinamic. Vectorul are alocată memoria exactă pentru numărul de mașini!
- Fiecare șir de caractere din cadrul unei mașini trebuie să aibă alocată exact atâta memorie câtă este nevoie (numarul de caractere +1).
- Se asigură că toate datele de intrare sunt corecte (ex: nu se introduce int în loc de char*).
- Șirurile nu vor avea mai mult de 20 caractere.
- Asigurați-vă că rezultatele consumului total de combustibil și costului total sunt double!
- Pentru consumul total al unei mașini, faceți cast la double la finalul calculului (ordine operații: înmulțire, împărțire, cast). Pentru cost, faceți cast la double după înmulțirea consumului total cu prețul corespunzător.
- Puteți crea o structură care să vă ajute, dar *nu este obligatoriu.*
- Fiecare cerință va fi rezolvată într-o funcție separată. Puteți folosi și alte funcții auxiliare. Atenție la transmiterea rezultatelor funcțiilor către main!
- Nu se fac afișări în funcții, ci doar în *main*! Pentru a transmite rezultatele folosiți parametri ai funcțiilor! **Rezolvările cu afișări în funcții nu vor fi punctate!**
- NU se folosesc variabile globale!
- **Rezolvările fară alocare dinamică vor fi depunctate!**

4.5. Testare și notare

- Punctajul maxim este de 40 puncte.
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **masini.c**

4.6. Exemple

4.6.1. Exemplul 1

Exemplul 1		
in	out	Explicație
5 Toyota B123ABC benzina 5.7 143 Mazda B141PIC motorina 5.9 141 Dacia BT12ALD benzina 9.3 169 Tesla B53APO electric 0 400 Renault CT43KIS hibrid 5.3 300 a	benzina - 2 motorina - 1 hibrid - 1 electric - 1	Sunt 2 mașini care folosesc benzină, și câte 1 pentru motorină, hibrid și electrică.

4.6.2. Exemplul 2

Exemplul 2		
in	out	Explicație
5 Toyota B123ABC benzina 5.7 143 Toyota BT12ALD benzina 9.3 169 Renault CT43KIS hibrid 5.3 300 Toyota B141PIC motorina 5.9 141 Renault B53APO hibrid 5 400 b	Toyota a consumat 32.19 - 268.70 lei Renault a consumat 35.90 - 287.92 lei	Se calculează, pentru fiecare mașină, consumul total și prețul plătit, conform formulei din cerință după care se adună consumul și costurile pentru mașinile care au aceeași marcă.

4.6.3. Exemplul 3

Exemplul 3		
in	out	Explicație
3 Toyota B123ABC benzina 5.7 143 Dacia BT12ALD benzina 9.3 169 Tesla B53APO electric 0 400 c	Numere corecte!	Numerele de înmatriculare respectă specificațiile din cerință.

4.6.4. Exemplul 4

Exemplul 4		
in	out	Explicație
5 Toyota B123ABC benzina 5.7 143 Mazda B141PIC motorina 5.9 141 Dacia BT12A4D benzina 9.3 169 Tesla B53AP electric 0 400 Renault 3T43KIS hibrid 5.3 300 c	Dacia cu numarul BT12A4D: numar invalid Tesla cu numarul B53AP: numar invalid Renault cu numarul 3T43KIS: numar invalid	BT12A4D – nu se termină cu exact 3 litere. B53AP – nu se termină cu exact 3 litere. 3T43KIS - trebuie să înceapă cu cel puțin o literă

5. PRECIZĂRI

- Separați logica programelor în mai multe funcții, așa cum vi s-a cerut în enunțuri!
- Atentie! Sa puneti linia noua de la final de output („\n”)!
- **Nu** se vor puncta sursele în care tot programul este scris în main!
- Toate citirile se fac de la tastatură in codul final al temei! Folosiți fișiere sau/si redirectarea intrarii/iesirii doar pentru testele voastre intermediare!
- Datele de intrare sunt descrise pentru fiecare cerință individual și vor fi citite de la tastatură.
- Este **interzisă** folosirea variabilelor globale!
- Soluția temei va fi scrisă în ANSII C! Nu folosiți sintaxă sau instrucțiuni specifice limbajului C++.
- Fiecare problemă va fi scrisă într-un fișier sursă separat, numit așa cum s-a precizat!
- În README precizați cât timp v-a luat implementarea cerințelor 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.
- **TEMELE CARE NU AU README NU SE VOR LUA ÎN CONSIDERARE!**
- Temele sunt strict individuale. Copierea temelor va fi sancționată cu punctaj 0 pentru toti cei care au portiuni de cod identice!!
- Persoanele cu porțiuni de cod identice **NU** vor primi niciun punctaj pe temă.
- **NU** copiați cod de pe Internet! Se poate ajunge la situația în care doi studenți să aibă același cod, preluat de pe Internet, caz în care ambele teme vor fi punctate in TOTALITATE cu 0, deși studenții nu au colaborat direct între ei.
- Temele trimise după deadline **NU** vor fi luate în considerare.

6. CODING STYLE

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:
  - 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 în coding style-ul ales

- ✓ Vă recomandăm să parcurgeți această resursă: 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

7. TRIMITEREA TEMEI

Veţi trimite o arhivă **ZIP** cu numele **EXACT** acesta: **GRUPA_Nume_Prenume_Tema1.zip**.
De exemplu: 311CC_Popescu_Maria_Tema1.zip.
Arhiva va conține următoarele fișiere:

1. README
2. cutii.c
3. lungi.c
4. codificari.c
5. masini.c

Atenție! Veți fi depunctați complet pentru formatarea incorectă a arhivei (alt nume, alt tip, alte nume pentru fișiere, alte fișiere în plus sau în minus etc.) - 0 puncte pe temă.

- Pentru întrebări legate de temă se va folosi în mod exclusiv **FORUM-ul** temei, pe care vă recomandăm să îl vizitați chiar și dacă nu aveți întrebări, întrucât este posibil să aflați informații noi din întrebările puse de colegii voștri, respectiv din raspunsurile date de noi.
- Compilarea nu ar trebui să producă warning-uri (verificați prin adăugarea flagului -Wall la gcc).
- Temele trebuie să fie încărcate pe **vmchecker**. **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.
  Link vmchecker: https://vmchecker.cs.pub.ro/ui/

8. PUNCTARE

EXERCIȚIU	PUNCTAJ
Problema cutiilor	20 puncte
Numere lungi	30 puncte
Transmisie bruiată	40 puncte
Car Dealer	40 puncte
TOTAL	**130 puncte**

Temele vor fi prezentate asistenților în cadrul primelor două laboratoare de după deadline!

Depunctările care se pot aplica:

- - 10 puncte pentru explicațiile din README
- - 10 puncte pentru lipsă comentarii din cod (atentie! nu trebuie comentata fiecare linie, doar ceea ce este esential pentru a se intelege usor rezolvarea – detalii la curs!)
- - 20 puncte pentru Coding Style
- Depunctare totală pentru lipsă fișier README (0 puncte pe temă)
- **Depunctare totală pentru formatarea incorectă a arhivei (alt nume, alt tip, alte nume pentru fișiere, alte fișiere în plus sau în minus)** - 0 puncte pe temă

O temă care NU compilează va fi punctată cu 0.

O temă care NU trece niciun test pe vmchecker va fi punctată cu 0.

Vor exista mai multe teste pentru fiecare problemă în parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste.

În fișierul README va trebui să descrieți pe scurt soluția pe care ați ales-o pentru fiecare problemă și alte lucruri pe care le considerați utile de menționat.

9. CHECKER

- Arhiva se va trimite pe vmchecker, unde tema se va testa automat.
- Pentru testarea locală, aveți disponibil un set de teste și un checker local.
- Punctajul acordat pe rularea testelor este cel de pe vmchecker.
- Corectorii își rezervă dreptul de a scădea puncte pentru orice problemă gsăsită în implementare, dacă vor considera acest lucru necesar.