PROGRAMAREA CALCULATOARELOR
Tema #1 Funcții, Vectori și Matrici

Deadline soft: 15.11.2022 23:55. Deadline: 19.11.2022 23:55

Responsabili: Radu Nichita, Alexandru Buzea, Bogdan Mocanu, Rares Croicia, Adrian Pana, Taisia Coconu, Ada Dobrica, Doru Cheresdi

Contents

Problema 1 - Un produs infinit **2**

Problema 2 - Simple query pe litere **4**

Problema 3 - Gigel și tabla de șah **7**

Problema 4 - Queries again ... **10**

Regulament **13**

Arhivă . . . 13
Checker . . . 13
Punctaj . . . 13
Reguli și precizări . . . 14
Alte precizări . . . 14

Problema 1 - Un produs infinit

Enunț

Gigel abia a început primul lui an la ACS și a dat de greu cu matematica, în special algebra. Deși participă activ la cursuri și seminarii, el nu reușească să se prindă cum se fac exercițiile cu vectori. Din păcate, acesta nu e prea bun nici la înmulțiri, dar se se pricepe să scrie cod. Pentru că nu dorește să rămână în urmă nici cu programarea, alege să își rezolve problemele la matematică folosind limbajul C.

Tema lui curentă este de a calcula produsul scalar a doi vectori **a** și **b**. Fiind curios din fire, dar și pentru a-și îmbunătăți noțiunile de programare, Gigel se mai gândește la alte noțiuni din matematică pe care le poate calcula.

Se dau **N** perechi de numere, reprezentând coordonatele (**ai**, **bi**) a 2 vectori **a** și **b**. Gigel dorește să calculeze următoarele:

- **ps** = produsul scalar al vectorilor **a** și **b** (în baza 8)
- **a_max** și **b_max** = al doilea maxim din vectorul **a**, respectiv **b** (de asemenea, în baza 8)
- **n_a** și **n_b** = norma 2 pentru fiecare dintre vectorii **a** și **b** (în baza 10)

Pentru a face lucrurile mai interesante, Gigel vine cu următoarea cerință pentru voi: citirea vectorilor **a** și **b** se va face având in vedere numere in **baza 8**.

Restricții și precizări

- $0 < N \leq 10^7$
- $0 < a_i, b_i < 10^{14}$
- dacă al doilea maxim nu există, se va afișa -1 pentru vectorul respectiv.

ATENȚIE! N poate fi foarte mare, astfel încât stocarea vectorilor nu este permisă. O rezolvare care reține valorile în 2 vectori a și b va obține maxim 4 puncte din 15 posibile, datorită restricțiilor de memorie de pe VMchecker.

Date de intrare

Toate datele se vor citi de la **STDIN**.

```
N
a_0 b_0
a_1 b_1
a_2 b_2
...
a_{n - 1} b_{n -1}
```

Date de ieșire

Toate datele se vor afișa la **STDOUT**. Numerele reale se vor afișa cu **7 zecimale** exacte.

```
ps
a_max b_max
n_a n_b
```

Exemplu

Date de intrare

```
9
1 1
2 2
2 4
2 4
2 5
2 5
3 5
3 2
12 12
```

Date de ieșire

```
162
3 5
11.7898261 14.6969385
```

Explicație

1. vectorul A este format din tuplul (1, 2, 2, 2, 2, 2, 3, 3, 10) - elemente în baza 10.
2. vectorul B este format din tuplul (1, 2, 4, 4, 5, 5, 5, 2, 10) - elemente în baza 10.

Produsul scalar al vectorilor a și b este dat de formula:

$$a \cdot b = \sum_{i=1}^{9} a_i * b_i = (1*1 + 2*2 + 2*4 + ... + 12*12)_{(8)} = 162_{(10)}$$

Dacă sortăm coordonatele vectorilor A și B, (ca și numere), în ordine descrescătoare obținem:

- pentru vectorul A: (10, **3**, 3, 2, 2, 2, 2, 2, 1)
- pentru vectorul B: (10, **5**, 5, 5, 4, 4, 2, 2, 1)

Al doilea maxim pentru vectorul A este 3, în timp ce pentru vectorul B este 5.

Pentru a calcula norma 2 a unui vector n-dimensional **v** ne folosim de formula:

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$$

Astfel, pentru vectorii a și b obținem:

$$\|a\|_2 = \sqrt{\sum_{i=1}^{9} a_i^2} = \sqrt{(1*1 + 2*2 + 2*2 + ... + 10*10)} = 11.789826$$

$$\|b\|_2 = \sqrt{\sum_{i=1}^{9} b_i^2} = \sqrt{(1*1 + 2*2 + 4*4 + ... + 10*10)} = 14.696938$$

Problema 2 - Simple query pe litere

Enunț

După ce bazele de numerație nu mai reprezintă pentru Gigel niciun secret, acesta decide să analizeze un flux de caractere printabile, toate litere mici ale alfabetului englez. Pentru fluxul dat, el își dorește să realizeze o serie de operații pe măsură ce primește literele, cum ar fi:

- Numărul de apariții ale unei litere date de la tastatură.
- Cele mai importante **K** litere din punctul de vedere al numărului de apariții, unde **K** este citit de la tastatură.
- Eliminarea numărului de apariții corespunzătoare unei litere până în acel moment.

De asemenea, el își dorește să primească alerte - mesaje standard, care îi semnalează momentul în care frecvența unei litere este cel puțin egală cu **50% din totalul numărului de litere primite până în acel moment**. Există însă alte câteva condiții necesare pentru ca alerta să se declanșeze:

- Numărul de litere distincte primite până în acel moment este de cel puțin 2 (nu putem declanșa o alertă pentru apariții unice ale unei litere).
- O alertă nu poate fi declanșată la mai puțin de 5 litere primite de la momentul declanșării ultimei alerte.
- În particular, pentru prima alertă, aceasta nu poate fi declanșată dacă avem primite un număr de mai puțin de 5 litere.

Scrieți un program care citește de la tastatură litere până la primirea unui caracter invalid (diferit de cele menționate mai sus), astfel:

- La apariția unei litere mici în stream, nu se execută niciun query.
- La apariția literei **Q** (interogarea numărului de apariții), programul va citi o literă **L** pentru care se dorește numărul de apariții până în acel moment.
- La apariția literei **T** (interogarea celor mai importante **K** litere primite în flux până la acel moment), programul va citi un număr întreg **K** care va reprezenta numărul celor mai importante **K** litere care se doresc a fi afișate
- La apariția literei **E** (eliminarea numărului de apariții pentru o literă), programul va citi o literă **L** pentru care se dorește eliminarea numărului de apariții până în acel moment.

Programul va răspunde la cele trei tipuri de interogări și va oferi ca output:

- În cazul declanșării unei alerte, va printa la STDOUT litera dominantă, împreună cu raportul dintre numărul de apariții ale literei și numărul total de litere, exprimat ca **fracție ireductibilă**.
- În cazul unei interogări a numărului de apariții (**Q**), va afișa la **STDOUT** litera, împreună cu numărul de apariții ale acelei litere.
- În cazul unei interogări a celor mai importante k litere (**T**), va printa la **STDOUT** cele mai importante k litere, în ordine descrescătoare a numărului de apariții, iar lexicografic în cazul de egalitate al numărului de apariții.
- În cazul unei operații de eliminare a numărului de apariții, nu se va afișa nimic la **STDOUT**.

Restricții și precizări

- $0 \leq N \leq 2^{64} - 1$, unde N este numărul total de litere;
- $1 \leq K \leq 26$
- Programul își termină execuția atunci când întâlnește în stream un caracter printabil invalid (diferit de cele utilizate până acum: nu este literă mică sau una dintre cele 3 litere mari folosite pentru interogări).

Exemplul 1

Date de intrare

```
a
b
a
Q a
a
a
b
b
c
T 2
E a
T 2
X
a
a
b
c
c
```

Date de ieșire

```
a 2
a 4/5
a b
b c
```

Explicație

Algoritmul parcurge următorii pași:

1. La primirea primelor litere, nu se va afișa nimic (liniile 1-3)
2. La interogarea **Q a**, se va afișa **a 2**, deoarece litera **a** apare de 2 ori printre literele primite până în acest moment.
3. După primirea a încă 2 litere (astfel având un total de 5 litere primite), avem 4 apariții ale lui a și doar o apariție a lui b, de aceea se declanșează o alertă și se va afișa **a 4/5**.
4. Se primesc încă 3 litere (liniile 7-9).
5. Se solicită un top al primelor 2 litere după cele mai multe apariții. Litera **a** are 4 apariții, **b** are 3, **c** are 1, iar restul literelor au 0, de aceea se va afișa **a b**.
6. Se elimină numărul de apariții ale lui **a**.
7. La realizarea unui nou top, au rămas doar litera **b** cu 3 apariții și **c** cu o singură apariție, de aceea se va printa **b c**.
8. Se primește caracterul **X**, așa că execuția se termină, iar caracterele primite ulterior se ignoră.

Exemplul 2

Date de intrare

```
x
y
z
z
z
Q x
Q y
Q z
x
x
y
y
z
z
z
E z
x
x
T 3
Y
Q a
E x
```

Date de ieșire

```
z 3/5
x 1
y 1
z 3
z 1/2
x y a
```

Explicație

Algoritmul parcurge următorii pași:

1. La liniile 1-5, primește primele 5 caractere.
2. După primirea caracterelor, se declanșează o alertă, întrucât s-au primit 5 caractere și avem o literă dominantă.
3. Se printează rezultatele celor 3 queries (numărul de apariții corespunzător lui **x**, **y** și **z**).
4. Se primesc caracterele de la liniile 9-15, iar alerta se declanșează abia după primirea și a ultimului z (când vom avea 12 litere primite, dintre care 6 sunt caractere z). Până acum, avem 6 apariții ale lui **z**, 3 apariții ale lui **y** și 3 apariții ale lui **x**.
5. Se elimină numărul de apariții ale lui **z**.
6. Se realizează top 3 litere în ordinea numărului de apariții, în ordine lexicografică (**x** are 5 apariții, **y** are 3 apariții, iar **a** are 0 apariții, însă este primul din punct de vedere lexicografic).
7. Se citește **Y**, caracter invalid, deci execuția se oprește.

Problema 3 - Gigel și tabla de șah

Enunț

După ce a dovedit și statisticile în ceea ce privește numărul de litere, Gigel își propune să termine tema la PCLP în timp util, însă fratele său mai mic, Gigelinho, îi distrage atenția. Astfel, pentru a-l ține ocupat, Gigel pune bilete pe care sunt scrise numere întregi. Apoi, cu ajutorul unei piese speciale de șah, amplasată în colțul din stânga-sus al tablei, Gigelinho trebuie să mute piesa respectând regulile:

- Dacă piesa se află pe un pătrat alb, atunci aceasta va fi mutată de-a lungul liniei pe care se află, cu un număr de pătrate egal cu numărul scris pe bilet, astfel: dacă numărul este pozitiv, atunci piesa se va muta la dreapta, iar în caz contrar, piesa se va muta la stânga.
- Dacă piesa se află pe un pătrat negru, atunci aceasta va fi mutată de-a lungul coloanei pe care se află, cu un număr de pătrate egal cu numărul scris pe bilet, astfel: dacă numărul este pozitiv, atunci piesa se va muta în jos, iar în caz contrar, piesa se va muta în sus.
- În ambele situații, biletul este folosit doar o dată (în cazul în care piesa revine pe aceeași poziție, se oprește).

La final, Gigelinho trebuie să determine distanța totală parcursă de piesă, precum și coordonatele câmpului pe care se află piesa la momentul final (ca la jocul de șah - vezi figura 1).

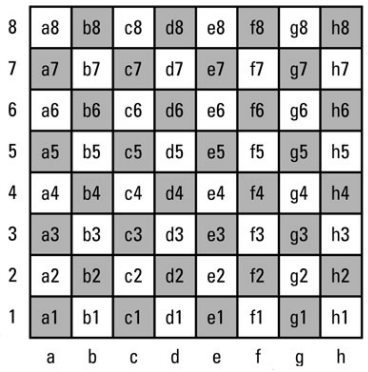

Figure 1: Coordonatele unui câmp pe tabla de șah 8 x 8.

Restricții și precizări

- $1 \leq N \leq 1000$;
- Pentru coloană, în cazul în care coloana nu poate fi reprezentată folosind un caracter, va fi folosită reprezentarea existentă în Microsoft Excel (după caracterul **Z**, va urma **AA** drept index de coloană).
- Se consideră că pătratul din colțul stânga-sus al tablei este alb.

Date de intrare

Toate datele se vor citi de la **STDIN**.

```
n
a_00 a_01 a_02 ... a_0(n-1)
a_10 a_11 a_12 ... a_1(n-1)
...
a_(n-1)0 a_(n-1)1 a_(n-1)2 ... a_(n-1)(n-1)
```

Date de ieșire

Toate datele se vor afișa la **STDOUT**. Se vor afișa distanța parcursă de piesă și respectiv coordonatele finale ale acesteia, în formatul precizat.

```
distance
x_final y_final
```

Exemplul 1

Date de intrare

```
2
1 1
0 -1
```

Date de ieșire

```
3
2 B
```

Explicație

Piesa parcurge următoarele transformări:

$$a_{00}(+1/alb) \rightarrow a_{01}(+1/negru) \rightarrow a_{11}(-1/alb) \rightarrow a_{10}(0/negru) \rightarrow STOP$$

- Distanța parcursă este $1 + 1 + 1 = \mathbf{3}$
- Poziția finală: linia din matrice 0, corespunzătoare liniei de pe tabla de șah bf2 și coloana 1, corespunzătoare indexării cu **B**

Exemplul 2

Date de intrare

```
4
1 2 -1 0
2 -1 0 -1
1 -1 1 -2
1 -1 1 0
```

Date de ieșire

```
8
4 A
```

Explicație

Piesa parcurge următoarele transformări:

$$a_{00}(+1/alb) \rightarrow a_{01}(+2/negru) \rightarrow a_{21}(-1/negru) \rightarrow a_{11}(-1/negru) \rightarrow a_{10}(+2/negru)$$

$$\rightarrow a_{30}(+1/negru) \rightarrow a_{00}(0/alb) \rightarrow STOP$$

- Distanța parcursă este $1 + 2 + 1 + 1 + 2 + 1 = \mathbf{8}$
- Poziția finală: linia din matrice 0, corespunzătoare liniei de pe tabla de șah **4** și coloana 0, corespunzătoare indexării cu **A**

Problema 4 - Queries again ...

Enunț

La final de zi, Gigel e fericit că și-a terminat toate temele pentru facultate și se poate juca într-un final. Deși se apropie parțialele și alte teste, el decide că merită să se relaxeze. Așadar, decide să se distreze cu jocul lui favorit, Sudoku. Fiind ceva mai experimentat la programare, după câteva zile lungi de codat, încearcă să determine dacă poate folosi programarea pentru a termina mai repede tabla și a reduce numărul de mutări.

El încă nu știe backtracking, așa că pentru moment dorește doar să verifice câteva lucruri pentru o tablă de sudoku. El are în față o tablă de $\mathbf{N^2} * \mathbf{N^2}$ elemente pe care dorește să efectueze **M** operații. O operație poate fi de următoarele tipuri:

- operația de tipul 1 : se dă un număr natural **X**, între 1 și N^2. Se cere să se determine ce numere pot pune pe linia **X**. Se va afișa o listă cu N^2 elemente astfel:
  1. 1, dacă valoarea i se poate pune pe linia X.
  2. 0, altfel.
- operația de tipul 2 : se dă un număr natural **Y**, între 1 și N^2. Se cere să se determine ce numere pot pune pe coloana **Y**. Se va afișa o listă cu N^2 elemente astfel:
  1. 1, dacă valoarea i se poate pune pe coloana Y.
  2. 0, altfel.
- operația de tipul 3 : se dă un număr natural **Z**, între 1 și N^2. Se cere să se determine ce numere pot pune în careul **Z**. Se va afișa o listă cu N^2 elemente astfel:
  1. 1, dacă valoarea i se poate pune în careul Z.
  2. 0, altfel.
- operația de tipul 4 : se dau două numere naturale **X** și **Y**, între 1 și N^2. Se cere să se afișeze ce putem pune în celula cu coordonatele **X** și **Y**. Se va afișa o listă cu N^2 elemente astfel:
  1. 1, dacă valoarea i se poate pune în celula de coordonate X și Y.
  2. 0, altfel.
- operația de tipul 5 : se dau trei numere naturale **X**, **Y**, **C**, între 1 și N^2. Să se pună valoarea **C** în celula cu coordonatele **X** și **Y**. Pentru această operație nu se va afișa nimic.
- operația de tipul 6 : să se determine starea curentă a tablei. Se va afișa :
  1. **0** - dacă încă se poate completa jocul
  2. **1** - dacă am ajuns într-o configurație invalidă a tablei
  3. **2** - dacă jocul a fost câștigat

Restricții și precizări

- $1 \leq N \leq 20$
- $1 \leq M \leq 200.000$
- pentru operațiile de tipul 1, se vor ține cont exclusiv de restricțiile de linie.
- pentru operațiile de tipul 2, se vor ține cont exclusiv de restricțiile de coloană.
- pentru operațiile de tipul 3, se vor ține cont exclusiv de restricțiile de careu.
- numerotarea liniilor, coloanelor și careurilor va începe de la 0.
- celula din stânga sus a grid-ului are coordonatele $(\mathbf{0}, \mathbf{0})$.
- pentru operația 4, dacă celula de coordonate **X** și **Y** conține deja o valoare, atunci se va afișa $\mathbf{N^2}$ zero-uri.
- pentru operația 5, dacă celula de coordonate **X** și **Y** conține deja o valoare, atunci operația se va ignora (**NU** se va modifica valoarea deja existentă).

Date de intrare

Datele de intrare se vor citi de la **STDIN** astfel:

- Pe prima linie se vor regăsi două numere naturale **N** și **M**, reprezentând dimensiunea grid-ului, respectiv numărul de operații.
- Pe următoarele $\mathbf{N^2}$ linii se vor găsi câte $\mathbf{N^2}$ elemente reprezentând valorile de pe linia **i** a grid-ului. O valoare de 0 reprezintă o celulă goală. Se garantează că grid-ul inițial are o configurație validă.
- Pe următoarele **M** linii se vor găsi operații de tipul celor menționate anterior în formatul:

$$< numr\_operatie > [parametru1] [parametru2] [parametru3] ...$$

Date de ieșire

Programul va afișa, pentru operațiile de tipul 1,2,3,4,6 rezultatul interogării corespunzătoare, conform precizărilor anterioare.

Exemplul

Date de intrare

```
3 9
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
1 5
2 6
3 7
6
4 2 0
4 8 8
5 8 0 1
5 8 1 6
6
```

Date de ieșire

```
1 0 1 1 1 0 0 1 1
1 0 1 1 1 1 1 1 1
0 1 1 0 1 1 1 0 0
0
1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
1
```

Explicație

Datele de intrare reprezintă grid-ul din Figura 2.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 2: Grid de Sudoku din exemplul problemei 4.

Avem 9 queries de rezolvat:

- 1 5 -> numerele ce pot fi puse pe linia 5 (numerotarea liniilor începe de la 0) sunt 1, 3, 4, 5, 8, 9.
- 2 6 -> numerele ce pot fi puse pe coloana 6 sunt 1, 3, 4, 5, 6, 7, 8, 9.
- 3 7 -> numerele ce pot fi puse în careul 7 sunt 2, 3, 5, 6, 7.
- 6 -> nu am făcut încă nicio mutare, suntem în configurația inițială, deci putem continua jocul.
- 4 2 0 -> numerele ce pot fi puse în celula (2, 0) sunt 1 și 2.
- 4 8 8 -> celula (8, 8) conține deja o valoare deci
- 5 8 0 1 -> în celula (8, 0) se pune valoarea 1.
- 5 8 1 2 -> în celula (8, 1) se pune valoarea 6.
- 6 -> ajungem într-o configurație invalidă (în careul 6 avem pe coloana 1 a careului pe aceeași coloană două valori de 6).

Regulament

Regulamentul general al temelor se găsește pe ocw (Temele de casă). Vă rugăm să îl citiți integral înainte de continua cu regulile specifice acestei teme.

Arhivă

Soluția temei se va trimite ca o arhivă **zip**. Numele arhivei trebuie să fie de forma **Grupă_NumePrenume_TemaX.zip** - exemplu: `311CA_NichitaRadu_Tema1.zip`.

Arhiva trebuie să conţină în directorul **RĂDĂCINĂ** doar următoarele:

- Codul sursă al programului vostru (fișierele **.c** și eventual **.h**).
- Un fișier **Makefile** care să conțină regulile **build** și **clean**.
  - Regula **build** va compila codul vostru și va genera următoarele executabile:
    - **infinite_prod** pentru problema 1
    - **simple_queries** pentru problema 2
    - **gigel_and_the_checkboard** pentru problema 3
    - **another_queries** pentru problema 4
  - Regula **clean** va șterge **toate** fișierele generate la build (executabile, binare intermediare etc).
- Un fișier **README** care să conțină prezentarea implementării alese de voi. **NU** copiați bucăți din enunț.

Arhiva temei **NU** va conține: fișiere binare, fișiere de intrare/ieșire folosite de checker, checkerul, orice alt fișier care nu este cerut mai sus.

Numele şi extensiile fişierelor trimise **NU** trebuie să conţină spaţii sau majuscule, cu exceptia fișierului README (care este are nume scris cu majuscule și nu are extensie).

Nerespectarea oricărei reguli din secțiunea **Arhivă** aduce un punctaj **NUL** pe temă.

Checker

Pentru corectarea aceste teme vom folosi scriptul **check** din arhiva **check_first_blood_remastered.zip** din secțiunea de resurse asociată temei. Vă rugăm să citiți **README.md** pentru a ști cum să instalați și utilizați checkerul.

Punctaj

Distribuirea punctajului:

- Problema 1: 15p
- Problema 2: 20p
- Problema 3: 25p
- Problema 4: 30p
- Modularizare : 10p
- Claritatea și calitatea codului: 10p
- Claritatea explicaţiilor din README: 10p

ATENȚIE! Punctajul maxim pe temă este 100p. Acesta reprezintă 0.5p din nota finală la această materie. La această temă se pot obține până la 120p (există un bonus de 20p), adică un punctaj maxim de 0.6p din nota finală.

Reguli și precizări

- Punctajul pe teste este cel acordat de script-ul **check**, rulat pe **vmchecker**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).
- Punctajul pe calitatea explicațiilor și a codului se acordă în mai multe etape:
  - **corectare automată**
    - Checkerul va încerca să detecteze în mod automat probleme legate de coding style și alte aspecte de organizare a codului.
    - Acesta va puncta cu maxim 30p dacă nu sunt probleme detectate.
    - Punctajul se va acorda proporțional cu numărul de puncte acumulate pe teste din cele 90p.
    - Checkerul poate să aplice însă și penalizări (exemplu pentru warninguri la compilare) sau alte probleme descoperite la runtime.
  - **corectare manuală**
    - Tema va fi corectată manual și se vor verifica și alte aspecte pe care checkerul nu le poate prinde. Recomandăm să parcurgeți cu atenție tutorialul de coding-style de pe ocw.cs.pub.ro.
    - Codul sursă trebuie să fie însoțit de un fișier README care trebuie să conțină informațiile utile pentru înțelegerea funcționalității, modului de implementare și utilizare a programului. Acesta evaluează, de asemenea, abilitatea voastră de a documenta complet și concis programele pe care le produceți și va fi evaluat, in mod analog CS, de către echipa de asistenți. In funcție de calitatea documentației, se vor aplica depunctări sau bonusuri.
    - Deprinderea de a scrie cod sursă de calitate, este un obiectiv important al materiei. Sursele greu de înteles, modularizate neadecvat sau care prezintă hardcodări care pot afecta semnificativ mentenabilitatea programului cerut, pot fi depunctate adițional.
    - În această etapă se pot aplica depunctări mai mari de 30p.
    - O temă care **NU** compilează cu -Wall -Wextra este depunctată la corectarea manuală cu 5p (punctajul echivalent pentru warnings).

Alte precizări

- Implementarea se va face in limbajul C, iar tema va fi compilată şi testată **DOAR** într-un mediu **LINUX**. Nerespectarea acestor reguli aduce un punctaj **NUL**.
- Tema trebuie trimisă sub forma unei arhive pe site-ul cursului curs.upb.ro si pe vmchecker.
- Tema poate fi submisă de oricâte ori fără depunctări până la deadline. Mai multe detalii se găsesc în regulamentul de pe ocw.
- O temă care **NU** compilează pe vmchecker **NU** va fi punctată.
- O temă care compilează, dar care **NU** trece niciun test pe vmchecker, NU va fi punctată.
- Punctajul pe teste este cel acordat de **check** rulat pe **vmchecker**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).
- Ultima temă submisă pe vmchecker poate fi rulată de către responsabili de mai multe ori în vederea verificării faptului că nu aveți buguri în sursă. Vă recomandăm să verificați **local** tema de mai multe ori pentru a verifica că punctajul este mereu același, apoi să încărcați tema.