

# PA - TEMA 2

## - Gigel și Grafurile -

Responsabili:  
Victor Nonea, Andrei Preda,  
Gabriel-Oprea Groza, Bogdan Cazan

Deadline hard: **02.06.2022 23:55**

### CUPRINS

1	Curățare multi-agent	3
1.1	Enunț . . . . .	3
1.2	Date de intrare . . . . .	3
1.3	Date de ieșire . . . . .	3
1.4	Restricții și precizări . . . . .	3
1.5	Testare și punctare . . . . .	4
1.6	Exemple . . . . .	4
2	Fortificații	6
2.1	Enunț . . . . .	6
2.2	Exemplu detaliat . . . . .	6
2.3	Date de intrare . . . . .	7
2.4	Date de ieșire . . . . .	7
2.5	Restricții și precizări . . . . .	7
2.6	Testare și punctare . . . . .	7
2.7	Exemple . . . . .	8
3	Beamdrone	10
3.1	Enunț . . . . .	10
3.2	Date de intrare . . . . .	10
3.3	Date de ieșire . . . . .	10
3.4	Restricții și precizări . . . . .	10
3.5	Testare și punctare . . . . .	11
3.6	Exemple . . . . .	11
4	Curse	13
4.1	Enunț . . . . .	13

4.2	Date de intrare . . . . .	13
4.3	Date de ieșire . . . . .	13
4.4	Restricții și precizări . . . . .	13
4.5	Testare și punctare . . . . .	14
4.6	Exemple . . . . .	14
5	Punctare . . . . .	15
5.1	Checker . . . . .	15
6	Format arhivă . . . . .	17
7	Links . . . . .	18
8	Modificări . . . . .	19

# 1 CURĂȚARE MULTI-AGENT

## 1.1 Enunț

$|R|$  roboței vor să curețe o casă reprezentată de un grid  $N \times M$ , în casă fiind exact  $|S|$  spații murdare. Fiecare spațiu din casă poate fi:

- Liber: Un roboțel ajunge pe el într-o unitate de timp și se poate deplasa în stânga, dreapta, sus sau jos, într-o unitate de timp, dacă spațiul vecin este de asemenea liber
- Murdar: Un roboțel trebuie să treacă prin spațiu măcar o dată pentru a fi considerat curat (nu e nevoie să se oprească un interval de timp)
- Punct de plecare pentru un roboțel
- Ocupat: roboțelii nu pot trece prin el

Spațiile murdare respectiv punctele de plecare pentru roboței funcționează ca spații libere pe lângă funcțiile lor specifice. Roboțelii pot trece prin același spațiu liber în același timp fără să se blocheze sau să interacționeze în orice fel.

Ajutați roboțelii să afle timpul minim în care pot curăța toată casa.

## 1.2 Date de intrare

Fișierul *curatare.in* va conține pe prima linie 2 numere,  $N$  și  $M$ . Pe următoarele  $N$  linii va conține un șir de  $M$  caractere cu următoarele semnificații:

- . - Spațiu liber.
- R - Spațiu de pornire pentru un roboțel.
- S - Spațiu murdar.
- X - Spațiu ocupat.

## 1.3 Date de ieșire

În fișierul *curatare.out* se va scrie un singur număr: durata minimă în care roboțelii pot curăța toată casa.

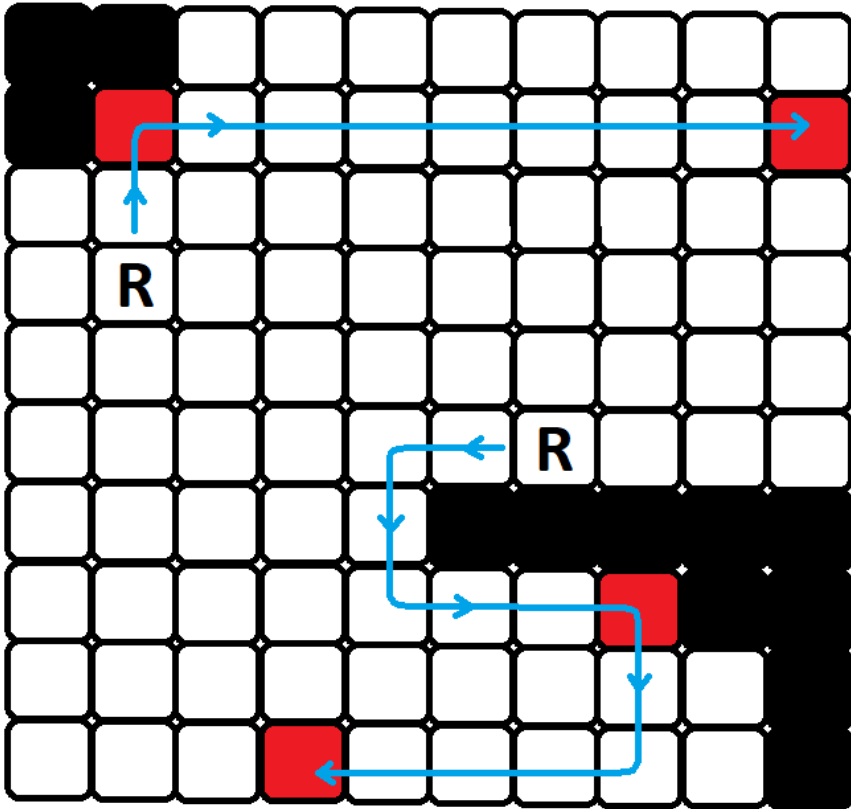
## 1.4 Restricții și precizări

- $1 \leq N, M \leq 1000$
- $1 \leq |R|, |S| \leq 4$
- Se garantează că orice roboțel poate ajunge în orice poziție liberă sau echivalentă.

### 1.5 Testare și punctare

- Punctajul maxim este de **40** puncte.
- Timpul de execuție:
  - C/C++: **2.5 s**
  - Java: **4 s**
- Sursa care conține funcția main trebuie obligatoriu denumită:  
*curatare.c, curatare.cpp sau Curatare.java.*

### 1.6 Exemple

Exemplul 1		
curatare.in	curatare.out	Explicație
<pre> 10 10 XX..... XS.....S ..... .R..... ..... .....R... .....XXXXX .....SXX .....X ...S.....X           </pre>	13	 <p>Roboțelul de sus își termină traseul în 10 unități de timp, cel de jos în 13.</p>

Exemplul 2		
curatare.in	curatare.out	Explicație
10 10 ..... ..S.....S. X.....X XX.....XX X.....XXX .....XXX .....RX RXXXXX...X .....XR...X .....SXXXXX	11	<p>Cei 2 roboți din dreapta trec în același timp prin spațiul de deschidere al camerei lor.</p> <p>Observați că robotul cel mai din stânga nu merge către cel mai apropiat spațiu murdar, fiindcă lucrează cooperativ cu ceilalți roboți.</p> <p>Durata finală e determinată de traseul robotului din mijloc.</p>

## 2 FORTIFICAȚII

### 2.1 Enunț

În imperiul lui Caesar circulă zvonul că populațiile barbare se pregătesc să atace. Împăratul se teme de armata lor, dar și de cultura lor neobișnuită (mai ales de gusturile muzicale pe care nu le înțelege, complet ne-romane), așa că dorește să îngreuneze înaintarea barbarilor către capitală cu orice preț. S-a hotărât să facă acest lucru construind fortificații pe rutele din imperiu.

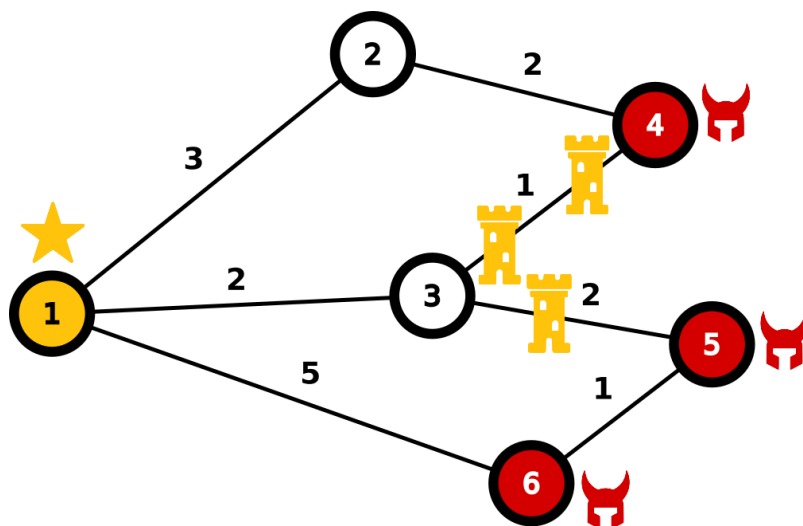
Harta imperiului conține mai multe localități, dintre care unele sunt controlate de barbari. Capitala imperiului **nu** este controlată de barbari. Localitățile sunt conectate prin rute ce pot fi străbătute în ambele sensuri, fiecare rută fiind parcursă într-o anumită durată de timp. Între două localități nu există mai multe rute directe.

Caesar poate construi fortificații **doar pe rutele conectate la o așezare barbară**. Celelalte rute trebuie lăsate libere, pentru a nu afecta comerțul în imperiu. O fortificație va întârzia armata barbarilor cu o unitate de timp față de durata normală a acelei rute. De asemenea, mai multe fortificații pot fi construite pe aceeași rută, caz în care întârzierile provocate se adună.

Din păcate, resursele imperiului sunt limitate și permit construirea a cel mult **K** fortificații, așa că ele trebuie plasate inteligent. Dacă am plasa fortificațiile astfel încât să încetinim adversarii cât mai mult, care este timpul minim în care o armată a barbarilor poate ajunge în capitală?

### 2.2 Exemplu detaliat

Să presupunem că harta imperiului arată ca în figura următoare:



În imperiu există 6 localități, dintre care 3 sunt controlate de barbari (4, 5 și 6). Capitala are mereu indicele 1. Inițial, timpul minim în care capitala poate fi atacată

este 3, deoarece o armată barbară poate pleca din localitatea 4 și trece prin localitatea 3. Toate celelalte drumuri au o durată mai mare (din localitatea 5 ajungem în 4 unități de timp, iar din localitatea 6, în 5).

Dacă avem posibilitatea să construim 3 fortificații, ar trebui să plasăm două pe ruta 3-4 și una pe ruta 3-5. Astfel, durata rutei 3-4 devine 3, iar a rutei 3-5 devine tot 3. Indiferent din ce localitate controlată pleacă, barbarii nu mai pot ajunge în capitală în mai puțin de 5 unități de timp. Din acest motiv, răspunsul pentru acest test ar fi 5.

### 2.3 Date de intrare

Fișierul de intrare *fortificatii.in* conține pe prima linie: numărul **N** de localități din imperiu, numărul **M** de rute, și numărul **K** de fortificații pe care le putem construi.

Pe a doua linie din fișier se află numărul **B** de localități controlate de barbari, urmat de cei **B** indici ai acestor localități ( $b_i$ ).

Următoarele **M** linii conțin triplete de forma  $x_i \ y_i \ t_i$ , reprezentând câte o rută ce leagă localitățile  $x_i$  și  $y_i$  și care este parcursă în  $t_i$  unități de timp.

### 2.4 Date de ieșire

În fișierul de ieșire *fortificatii.out* se va scrie un singur număr: durata minimă în care una din armatele barbare poate ajunge în capitală după ce am plasat fortificațiile.

### 2.5 Restricții și precizări

- $2 \leq N \leq 10^5$
- $1 \leq M \leq \min(\frac{N \cdot (N-1)}{2}, 2 \cdot 10^5)$
- $1 \leq B \leq N - 1$
- $1 \leq b_i \leq N$  (locațiile sunt indexate de la 1 la **N**)
- $1 \leq K \leq 10^9$
- $1 \leq t_i \leq 10^9$
- Capitala imperiului are mereu indicele 1.
- Există cel puțin un drum (o secvență de rute) între oricare două localități din imperiu.

### 2.6 Testare și punctare

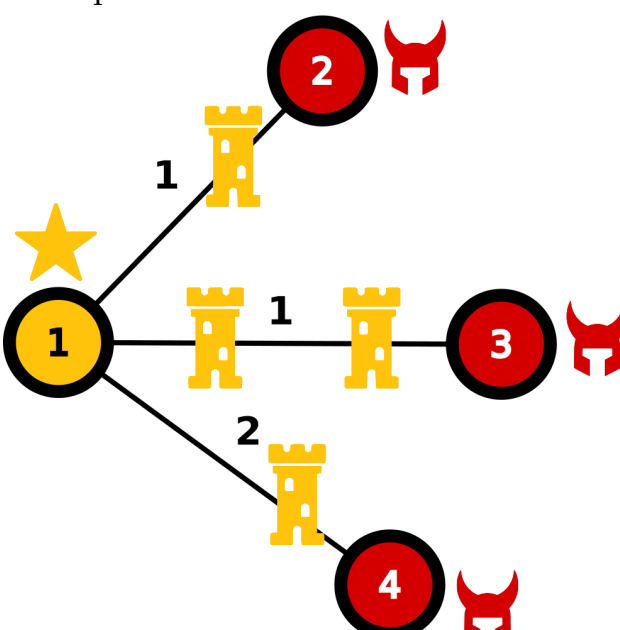
- Punctajul maxim este de 35 puncte.
- Timpul de execuție:
  - C/C++: 1 s
  - Java: 3 s

- Sursa care conține funcția main trebuie obligatoriu denumită:  
*fortificatii.c, fortificatii.cpp* sau *Fortificatii.java*.

## 2.7 Exemple

Exemplul 1		
fortificatii.in	fortificatii.out	Explicație
4 3 5 2 3 4 1 2 1 2 3 3 2 4 2	6	<p>Putem construi 5 fortificații astfel:</p> <p>Observați că nu putem fortifica decât rutele conectate la așezări barbare.</p>



Exemplul 2		
fortificatii.in	fortificatii.out	Explicație
4 3 4 3 2 3 4 1 2 1 1 3 1 1 4 2	2	Harta poate arăta astfel: 

Exemplul 3		
fortificatii.in	fortificatii.out	Explicație
6 7 3 3 4 5 6 1 2 3 1 3 2 1 6 5 2 4 2 3 4 1 3 5 2 5 6 1	5	Exemplul este cel detaliat mai sus, în secțiunea 2.2.

### 3 BEAMDRONE

#### 3.1 Enunț

Gigel și-a descoperit o pasiune pentru hardware, așa că a decis să învețe intens și să își aplice cunoștințele construind o dronă care să ajungă cât mai rapid la o destinație și să participe la curse cu alte drone.

Drona lui Gigel folosește un motor cu reacție de ultima generație, astfel încât timpul în care se deplasează în orice punct de pe direcția ei de mers poate fi considerat nul.

Totuși, pentru a nu se cionci de un perete, drona ar avea nevoie de un senzor care să îi permită să detecteze pereții și să se rotească înainte de a se lovi de aceștia. Din păcate, Gigel încă este la început de drum, astfel încât a reușit să facă drona să se rotească doar cu  $90^\circ$  într-o secundă. Astfel, dacă ar vrea să urmeze o direcție perpendiculară i-ar lua o secundă, iar dacă ar vrea să facă o întoarcere i-ar lua 2 secunde.

Momentul mult așteptat a sosit: se organizează o întrecere între cele mai rapide drone din orașul lui Gigel. Pentru a fi cu un pas în fața concurenței sale și pentru a-și cunoaște șansele de victorie, Gigel vrea să știe în avans care este cel mai bun timp în care va putea termina cursa drona sa.

#### 3.2 Date de intrare

Aria pistei va fi reprezentată de un grid  $N \times M$ . Fișierul *beamdrone.in* va conține pe prima linie 2 numere, dimensiunile ariei pistei,  $N$  și  $M$ .

Pe a doua linie se vor găsi 4 numere  $x_i, y_i, x_f, y_f$ , ce semnifică coordonatele inițiale ale dronei, respectiv coordonatele spațiului de finish.

Pe următoarele  $N$  linii se vor găsi câte  $M$  caractere, câte unul pentru fiecare spațiu din pista de cursă, cu următoarea codificare:

- $\bar{w}$  - Spațiu care conține pereți
- .
 - Spațiu liber prin care poate trece drona.

#### 3.3 Date de ieșire

În fișierul *beamdrone.out* se va scrie un singur număr: timpul minim în care drona lui Gigel poate finaliza cursa.

#### 3.4 Restricții și precizări

- $1 \leq N, M \leq 1000$
- $0 \leq x_i, x_f \leq N - 1$
- $0 \leq y_i, y_f \leq M - 1$

- Drona poate fi orientată doar în direcțiile cardinale: nord, sud, est, vest.
- În momentul inițial de timp, drona poate fi orientată în orice direcție preferabilă pentru drumul minim.
- Se garantează că drona poate ajunge în orice spațiu liber de pe aria traseului.

### 3.5 Testare și punctare

- Punctajul maxim este de **40** puncte.
- Timpul de execuție:
  - C/C++: **2 s**
  - Java: **2.5 s**
- Sursa care conține funcția main trebuie obligatoriu denumită: *beamdrone.c*, *beamdrone.cpp* sau *Beamdrone.java*.

### 3.6 Exemple

Exemplul 1		
beamdrone.in	beamdrone.out	Explicație
<pre> 6 6 5 0 4 4 .....W .W.W.W .WW..W .W.W.. ...W.. .W...W </pre>	2	<p>Drona poate finaliza cursa cel mai rapid urmărind traseul cu linie continuă, efectuând 2 rotiri cu cost total de 2 secunde. Deși drumul punctat era mai scurt, timpul petrecut pe acesta ar fi fost de 4 secunde.</p>

Exemplul 2		
beamdrone.in	beamdrone.out	Explicație
<pre> 6 6 4 3 2 1 .W.... W..WW. W.W... W....W ..W..W ..... </pre>	2	<p>Drona poate urma oricare din cele 2 trasee, cu un cost de 2 secunde.</p>

## 4 CURSE

### 4.1 Enunț

Gigel vrea să câștige marea cursă de mașini care urmează să aibă loc în orașul său. În oraș există  $N$  piste pe care se poate antrena.

Pentru a își mări șansele de succes, Gigel și-a cumpărat  $M$  mașini care pot fi mai mult sau mai puțin performante. Vom presupune că toate mașinile au performanțe diferite una față de celalaltă, dar o anumită mașină va obține mereu același timp de finish pe o anumită pistă.

Pistele din oraș sunt suficient de similare între ele încât dacă o mașină obține un timp mai bun de finish decât o altă mașină pe o anumită pistă, această relație va fi valabilă și pentru orice altă pistă.

Gigel face  $A$  antrenamente. Un antrenament constă în  $N$  curse de antrenament, acoperind toate cele  $N$  piste. Pentru fiecare cursă în cadrul unui antrenament, Gigel folosește o mașină oarecare.

Managerul lui Gigel, Marcel, îi cronometrează toate cursele de antrenament, și îi redă lui Gigel toate antrenamentele ordonate după timpul de finish de pe fiecare pistă și după o ordine de prioritate a pistelor.

Gigel nu mai are timp să își testeze toate mașinile pe aceeași pistă, așa că trebuie să îl ajutați să deducă ordinea performanțelor mașinilor lui pe baza ordinii antrenamentelor făcute.

### 4.2 Date de intrare

Fișierul *curse.in* va conține pe prima linie 3 numere, reprezentând  $N$ ,  $M$  și  $A$ . Pistele sunt indexate de la 1 la  $N$  unde un index inferior reprezintă o prioritate mai mare. Mașinile sunt indexate de la 1 la  $M$ .

Pe următoarele  $A$  linii sunt redade cele  $A$  antrenamente ordonate, sub formă de  $N$  indecși de mașină, poziția fiecărui index reprezentând pista pe care a fost folosită respectiva mașină.

### 4.3 Date de ieșire

În fișierul *curse.out* se va afla un șir de  $M$  indecși reprezentând ordinea performanțelor mașinilor lui Gigel.

### 4.4 Restricții și precizări

- $1 \leq N \leq 20$
- $1 \leq M \leq 10^4$
- $M \leq A \leq 10^5$

- Se garantează că există o singură ordonare validă pentru mașini.

#### 4.5 Testare și punctare

- Punctajul maxim este de **25** puncte.
- Timpul de execuție:
  - C/C++: **1 s**
  - Java: **3 s**
- Sursa care conține funcția main trebuie obligatoriu denumită:  
*curse.c*, *curse.cpp* sau *Curse.java*.

#### 4.6 Exemple

Exemplul 1		
curse.in	curse.out	Explicație
3 3 4 2 1 3 1 3 2 1 2 2 1 2 1	3 2 1	Din ordinea mașinilor pe pista 1 deducem că mașina 2 este mai performantă decât mașina 1, $P(2) > P(1)$ , în particular din antrenamentele 1 și 2. În caz de egalitate pe pista 1, observăm pe pista 2, că $P(3) > P(2)$ , din antrenamente 2 și 3. În caz de egalitate și pe pista 1, și pe pista 2, observăm că $P(2) > P(1)$ , din antrenamentele 3 și 4. Astfel avem $P(3) > P(2) > P(1)$ .

Exemplul 2		
curse.in	curse.out	Explicație
3 4 6 3 2 1 3 4 3 3 4 4 2 3 3 2 1 1 2 1 2	3 1 2 4	$P(3) > P(2)$ , pista 1, antrenamentele 3, 4. $P(2) > P(4)$ , pista 2, antrenamentele 1, 2. $P(3) > P(1)$ , pista 2, antrenamentele 4, 5. $P(3) > P(4)$ , pista 3, antrenamentele 2, 3. $P(1) > P(2)$ , pista 3, antrenamentele 5, 6. Astfel avem $P(3) > P(1) > P(2) > P(4)$ . Observăm că unele relații deduse din ordinea antrenamentelor pot fi redundante.

## 5 PUNCTARE

- Punctajul temei este de **125** puncte, distribuit astfel:
  - Problema 1: **40p**
  - Problema 2: **35p**
  - Problema 3: **40p**
  - 5 puncte vor fi acordate pentru comentarii și README.
  - 5 puncte vor fi acordate automat de checker pentru coding style. Totuși, la corectarea manuala se pot aplica **depunctari de până la 20 de puncte** pentru **coding style neadecvat**.

Punctajul pe README, comentarii și coding style este condiționat de obținerea unui punctaj strict pozitiv pe cel puțin un test.

Se poate obține un **bonus** de **25p** rezolvând problema **Curse**. Acordarea bonusului **NU** este condiționată de rezolvarea celorlalte probleme. În total se pot obține 150 de puncte (**NU** se trunchiază).

Pentru detalii puteți să vă uitați și peste **regulile generale** de trimitere a temelor.

- 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.
- Fiecare problemă va avea o limită de timp pe test (precizată mai jos și pe pagina cu enunțul). Dacă execuția programului pe un test al acelei probleme va dura mai mult decât limita de timp, veți primi automat 0 puncte pe testul respectiv și execuția va fi întreruptă.
- În fișierul README va trebui **să descrieți soluția** pe care ați ales-o pentru fiecare problemă, **să precizați complexitatea** pentru fiecare și alte lucruri pe care le considerați utile de menționat.

### 5.1 Checker

- Arhiva se va trimite pe **vmchecker**, unde tema se va testa folosind un set de teste private.
- Pentru testarea locală, aveți disponibil un set de teste publice (de aceeași dificultate) pe pagina cu **resurse** a temei.
- Checkerul se poate rula fără niciun parametru, caz în care va verifica toate problemele. De asemenea, se mai poate rula cu un parametru pentru a rula o anumită problemă:
 

```
./check.sh <1 | 2 | 3 | 4>
./check.sh <curatare | fortificatii | beamdrone | curse >
./check.sh cs
```

- **Punctajul pe teste** este cel de pe vmchecker și se acordă rulând tema doar cu testele private.
- Checkerul verifică doar existența unui README cu denumire corectă și conținut nenul. **Punctajul final pe README și comentarii** se acordă la corectarea manuală a temei.
- La corectarea manuală se poate depuncta pentru **erori de coding style** care nu sunt semnalate de checker.
- Corectorii își rezervă dreptul de a scădea puncte pentru orice problemă găsită în implementare, dacă vor considera acest lucru necesar.
- Pentru citirea în Java se recomandă folosirea **BufferedReader**.



## 6 FORMAT ARHIVĂ

- Temele pot fi testate automat pe vmchecker. Acesta suportă temele rezolvate în C/C++ și Java. Dacă doriți să realizați tema în alt limbaj, trebuie să-i trimiteți un email lui Traian Rebedea (traian.rebedea@cs.pub.ro), în care să îi cereți explicit acest lucru.
- Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa\_NumePrenume\_Tema1.zip** (ex: 399CX\_PuiuGigel\_Tema1.zip) și va conține:
  - Fișierul/fișierele sursă
  - Fișierul **Makefile**
  - Fișierul **README** (fără extensie)
- Fișierul pentru make trebuie denumit obligatoriu **Makefile** și trebuie să conțină următoarele reguli:
  - **build**, care va compila sursele și va obține executabilele
  - **run-p1**, care va rula executabilul pentru problema 1
  - **run-p2**, care va rula executabilul pentru problema 2
  - **run-p3**, care va rula executabilul pentru problema 3
  - **run-p4**, care va rula executabilul pentru problema bonus (**doar dacă** ați implementat și bonusul)
  - **clean**, care va șterge executabilele generate
- **ATENȚIE!** Funcția **main** din rezolvarea unei probleme se va găsi într-o sursă ce trebuie obligatoriu denumită astfel:
  - **curatare.c**, **curatare.cpp** sau **Curatare.java** - pentru problema 1
  - **fortificatii.c**, **fortificatii.cpp** sau **Fortificatii.java** - pentru problema 2
  - **beamdrone.c**, **beamdrone.cpp** sau **Beamdrone.java** - pentru problema 3
  - **curse.c**, **curse.cpp** sau **Curse.java** - pentru problema 4
- **ATENȚIE!** Tema va fi compilată și testată **DOAR pe Linux**.
- **ATENȚIE!** Numele regulilor și a surselor trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe testele asociate problemei rezolvate de regula respectivă.
- **ATENȚIE!** Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).
- **ATENȚIE!** Orice nerespectare a restricțiilor duce la pierderea punctajului (după regulile de mai sus).

## 7 LINKS

- [Regulament general PA](#)
- [Google C++ Style Guide](#)
- [Google Java Style Guide](#)
- [Debugging și Structuri de Date](#)

## 8 MODIFICĂRI

- 19.05.2022: Schimbat exemplul 1 la problema Curatare multi-agent.