PA - TEMA 2
- ROBIN HOOD AND GRAPHS-

Responsabili:
Stefan Popa, Iustin Sirbu, Liviu Chirimbu, Andreea Oltean, Mihai Musat

Deadline soft: **17.05.2020**
Deadline hard: **17.05.2020**

CUPRINS

1 Problema 1: Save Robin Hood 3
1.1 Enunț . . . 3
1.2 Date de intrare . . . 3
1.3 Date de ieșire . . . 3
1.4 Restricții și precizări . . . 4
1.5 Testare și punctare . . . 4
1.6 Exemple . . . 4
1.6.1 Exemplu 1 . . . 4
1.6.2 Exemplu 2 . . . 5
2 Problema 2: Robin Hood stealing for the poor 6
2.1 Enunț . . . 6
2.2 Date de intrare . . . 6
2.3 Date de ieșire . . . 6
2.4 Restricții și precizări . . . 6
2.5 Testare și punctare . . . 7
2.6 Exemple . . . 7
2.6.1 Exemplu 1 . . . 7
3 Problema 3: Fooling the guards 8
3.1 Enunț . . . 8
3.2 Date de intrare . . . 8
3.3 Date de ieșire . . . 8
3.4 Restricții și precizări . . . 9
3.5 Testare și punctare . . . 9
3.6 Exemple . . . 9
3.6.1 Exemplu . . . 9

4 Bonus: Robin Hood pe Lac 10
4.1 Enunț . . . 10
4.2 Date de intrare . . . 11
4.3 Date de ieșire . . . 11
4.4 Restricții și precizări . . . 11
4.5 Testare și punctare . . . 12
4.6 Exemple . . . 12
4.6.1 Exemplu 1 . . . 12
5 Punctare 14
5.1 Checker . . . 14
6 Format arhivă 16
7 Links 17

1 PROBLEMA 1: SAVE ROBIN HOOD

1.1 Enunț

Robin Hood, un aventurier al vremurilor trecute, a luptat întotdeauna pentru dreptate și adevăr, încercând să ia de la cei bogați și să dea celor săraci. Astfel, a reușit să-și facă o sumedenie de dușmani în rândul lorzilor care sunt acum în cautarea lui.

Aria de căutare a lui Robin este formată din N orașe conectate între ele prin M drumuri bidirecționale. Lorzii vor incepe cautarea în K orașe din totalul de N existente si vor cauta in toate orasele care sunt accesibile din cele K orase. Robin se ascunde in orasul 1 si are la dispozitie o armata care poate bloca o parte din orasele de pe harta. Aventurierul va fi găsit dacă se poate ajunge din unul din cele K orase in orașul 1, unde Robin se ascunde, mergând doar prin orașe care nu sunt blocate (inclusiv orașul de inceput trebuie să nu fie blocat). Armata lui Robin are ca scop blocarea unui numar minim de orase astfel incat sa il protejeze pe Robin. Orasul in care se afla Robin nu va fi niciodata blocat, intrucat asta l-ar da de gol. Ordinea în care armata lui Robin blochează orașele este dată de o permutare a acestora de la 2 la N: p[1], p[2], ... p[N - 1]. Astfel, dacă orașul p[i] este blocat, atunci toate orașele dinaintea lui p[i] în permutare trebuie să fie și ele blocate.

Ajutați armata lui Robin să găsească numărul minim de orașe ce trebuie blocate astfel încat Robin să nu se afle în pericol. Atenție! Orașul în care se ascunde Robin nu poate fi blocat.

1.2 Date de intrare

Pe prima linie a fișierului **p1.in** se afla 3 numere intregi N, M si K.

Pe a doua linie se află K numere întregi între 2 și N, reprezentând orașele din care lorzii vor incepe cautarile.

Pe a treia linie se află N - 1 numere întregi, reprezentând permutarea orașelor de la 2 la N.

Pe urmatoarele M linii se află 2 numere întregi u și v, semnificând faptul că între orașele u și v există un drum bidirecțional.

1.3 Date de ieșire

Fișierul **p1.out** va conține un singur număr întreg, reprezentând numărul minim de orașe ce trebuie blocate.

1.4 Restricții și precizări

- $2 \leqslant N \leqslant 10^5$
- $1 \leqslant M \leqslant 2 * 10^5$
- $1 \leqslant K \leqslant N - 1$

1.5 Testare și punctare

- Punctajul maxim este de **35** de puncte.
- Timpul de execuție:
  - C/C++: **1.5 s**
  - Java: **3 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **p1.c**, **p1.cpp** sau **P1.java**.

1.6 Exemple

1.6.1 *Exemplu 1*

Exemplu 1		
p1.in	p1.out	Explicație
4 4 1 3 2 3 4 1 2 1 4 2 3 2 4	1	Inchizand orasul 2, care se afla pe prima pozitie in permutare, lorzii ce pleaca din orasul 3 nu vor putea ajunge in orasul 1, unde se afla Robin.

1.6.2 *Exemplu 2*

Exemplu 2		
p1.in	p1.out	Explicație
6 7 3 3 4 6 2 3 4 6 5 1 2 1 5 2 3 2 4 2 6 3 4 5 6	4	Solutia optima este inchiderea primelor 4 orase din permutare: 2, 3, 4 si 6. Astfel, lorzii ce incep cautarea din orasele 3, 4 si 6 nu vor putea ajunge in orasul 1, unde se afla Robin.

2 PROBLEMA 2: ROBIN HOOD STEALING FOR THE POOR

2.1 Enunț

Pentru că a scăpat fără probleme, Robin Hood s-a hotărât să-i ajute pe locuitorii orașului Notthingam, care sunt supuși unor taxe inimaginabile de către șeriful din Notthingam. Acesta a decis să ia cât mai mult din galbenii vistieriei regale, situate într-un castel bine păzit.

Anglia este formata din N orase conectate prin M drumuri **unidirectionale**. Fiecare drum are asociat un efort (care poate fi si negativ) pe care Robin trebuie sa il depuna ca sa parcurga drumul respectiv. Din modul în care sunt construite drumurile, dacă Robin pornește din orasul A, acesta nu va mai putea ajunge niciodata înapoi în orasul A, oricare ar fi acel oras A.

Misiunea voastră este să-l ghidați pe Robin Hood pe ruta optimă către vistieria regală. Ruta optimă reprezintă drumul pe care Robin Hood depune cel mai puțin efort cumulat. Având la dispoziție orasul din care pleacă Robin și locul în care acesta trebuie să ajungă, ajutați-l să găsească drumul pe care depune cel mai puțin efort.

2.2 Date de intrare

Pe prima linie a fișierul **p2.in** se află două numere întregi N, M, reprezentând numărul de orase și de drumuri.

Pe a doua linie se află două numere întregi, reprezentând orasul sursă și orasul destinație.

Pe următoarele M linii se afla M triplete de numere întregi (i, j, e), insemnand că exista un drum unidirecțional de la orasul i la orasul j care necesită efortul e.

2.3 Date de ieșire

Fisierul **p2.out** va avea o singură linie ce va conține efortul minim pentru a ajunge de la sursă la destinație.

2.4 Restricții și precizări

- $2 \leqslant N \leqslant 10^5$
- $1 \leqslant M \leqslant 2 * 10^5$
- $1 \leqslant i, j \leqslant N$
- $-10000 \leqslant e \leqslant 10000$
- Se garanteaza ca se poate ajunge de la sursa la destinatie.
- Se garanteaza ca daca se pleaca dintr-un oras A, nu se poate ajunge inapoi in orasul A folosind drumurile existente. Acest lucru este valabil pentru orice oras A de pe harta.

2.5 Testare și punctare

- Punctajul maxim este de **40** puncte.
- Timpul de execuție:
  - C/C++: **1.5 s**
  - Java: **4 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **p2.c**, **p2.cpp** sau **P2.java**.

2.6 Exemple

2.6.1 *Exemplu 1*

Exemplu 1		
p2.in	p2.out	Explicație
8 12 1 8 1 2 1 1 3 2 1 4 5 2 5 4 2 6 11 3 5 9 3 6 5 3 7 16 4 7 2 7 8 2 5 8 18 6 8 -2	5	Drumul de timp minim de la 1 la 8 este prin 1 - 3 - 6 - 8 si necesita un efort total egal cu 5.

3 PROBLEMA 3: FOOLING THE GUARDS

3.1 Enunț

După ce a reușit să adune suficienti galbeni, Robin Hood trebuie să meargă să îi distribuie la săracii oameni din Notthingam. Acesta trebuie să fie atent căci șeriful din Notthingam a fost anunțat de faptele lui Robin Hood și i-a pregătit acestuia câteva surprize. Totuși, aventurierul nostru nu vrea să depună prea mult efort și vrea să ajungă în Notthingam cu cât mai multă energie. Acesta a primit de la ajutoarele sale o hartă care îi arată poziția fiecărui oraș, drumurile dintre orașe și numărul gărzilor pentru fiecare drum. Având această hartă, Robin Hood a calculat câtă energie va pierde în medie pentru fiecare drum pe care l-ar putea alege. Astfel, el știe procentul din energia rămasă pe care îl va pierde dacă va alege să meargă pe un anumit drum.

De exmplu, dacă Robin Hood avea la un moment dat energia E și merge din orașul A în orașul B, pierde p1% din energie, iar din orașul B în orașul C pierde p2% din energie. Robin rămâne cu $E * (1 - p1/100) * (1 - p2/100)$ energie la finalul drumului A-B-C.

Sarcina este să-l ghidați pe Robin Hood din York până în Notthingam astfel încât să-i rămână cât mai multă energie.

3.2 Date de intrare

Pe prima linie a fișierului de intrare p3.in se află trei numare intregi N, M, E reprezentând numărul de orașe din harta pe care a primit-o, numărul de drumuri dintre aceste orașe și energia inițială a lui Robin Hood.

Robin Hood se află inițial în orașul 1 și trebuie să ajungă în orașul N.

Pe următoarele M linii se află M triplete (i, j, p) semnificând faptul că exista un drum **unidirectional** de la orasul i la orasul j, iar dacă Robin Hood merge pe acest drum va pierde p% din energia pe care o avea în acel moment.

3.3 Date de ieșire

Pe prima linie a fișierului p3.out se va afla un numar real reprezentând energia cu care Robin Hood a ajuns în Notthingam, iar pe a doua linie se va afla drumul ales de acesta pentru a-i maximiza energia rămasă la final. Drumul va consta într-o listă de noduri ce începe cu 1 și se termină cu N.

3.4 Restricții și precizări

- $1 \leqslant E \leqslant 10^9$
- $2 \leqslant N \leqslant 10^5$
- $1 \leqslant M \leqslant 2 * 10^5$
- Tripletele (i, j, p) au $1 \leqslant i, j \leqslant N$ și p un număr întreg, $0 < p < 100$.
- Energia rămasă va fi considerată corectă dacă eroarea absoluta este de maxim 10^{-3}. Se recomandă afișarea soluției cu minim 4 zecimale.
- Se garantează că există **exact** un drum care maximizează energia rămasă la final.

3.5 Testare și punctare

- Punctajul maxim este de **40** puncte.
- Timpul de execuție:
  - C/C++: **1.5 s**
  - Java: **4 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **p3.c**, **p3.cpp** sau **P3.java**.

3.6 Exemple

3.6.1 *Exemplu*

Exemplu		
p3.in	p3.out	Explicație
7 9 10000 1 2 64 2 3 87 2 4 95 2 5 98 4 3 87 4 7 64 4 6 64 5 6 95 6 7 87	64.8 1 2 4 7	Pentru a ajunge în orașul 7 ne avantajeaza ruta 1-2-4-7, orice altă rută făcându-ne să rămânem cu mai puțin de 64.8 energie în orașul 7.

4 BONUS: ROBIN HOOD PE LAC

4.1 Enunț

Pe drumul spre un jaf de zile mari, Robin Hood o aude pe dragostea viestii sale, Maid Marian, tipand din mijlocul lacului. Neavand o barca, dar fiind ingenios, el vrea sa ajunga pana la Maid Marian sarind pe bustenii care plutesc pe lac. Totusi, din cauza curentilor, bustenii isi schimba poziția incontinuu, asa ca Robin Hood are nevoie de ajutorul vostru pentru a isi calcula traseul.

Robin Hood stie ca are la dispozitie timpul T pentru a ajunge la domnita, inainte ca aceasta sa se inece. De asemenea, el stie ca pe lac sunt N busteni, iar pentru fiecare dintre acestia el poate vedea pozitiile initiale ale capetelor: $(x_i^{start}, y_i^{start}); (x_i^{end}, y_i^{end})$, $i \in \{1, ..., N\}$.
In plus, cunoscand acel lac ca pe propriul lui buzunar, el stie deja pentru fiecare bustean in ce directie se va misca acesta la fiecare moment de timp. m_i^j, $i \in \{1, .., N\}, j \in \{0, ..T-1\}$, reprezinta directia in care se va misca busteanul i la momentul de timp j si poate fi una din directiile N, S, E, V. Se considera ca un bustean se misca la un moment dat cu exact o unitate.

Robin Hood incepe de pe primul capat al busteanului 1 la timpul 0. La fiecare moment de timp, Robin Hood are o decizie de luat:

1. el poate sta pe loc, relativ la busteanul pe care se afla. In acest caz, va consuma E_1 energie si se va misca odata cu busteanul.

2. el poate sa faca un pas pe bustean. In acest caz va consuma E_2 energie si se va deplasa cu o unitate in una din directiile N, S, E, V. Pentru un bustean aflat in pozitie verticala doar miscarile N si S sunt disponibile, iar pentru un bustean aflat in pozitie orizontala, doar E si V. Daca Robin se afla intr-unul din capetele busteanului, una din cele 2 mutari nu va fi disponibila, intrucat Robin ar cadea in apa.

3. uneori, doi busteni se pot intersecta (din cauza curentilor, un bustean poate ajunge sa treaca pe sub altul). In aceasta situatie, Robin are si optiunea de a sari de pe un bustean pe celalalt. Cu alte cuvinte, daca la momentul t Robin se afla pe un bustean B_1 in punctul de intersectie cu un alt bustean B_2, Robin poate alege sa sara pe B_2, tot in punctul de intersectie cu B_1, urmand sa se miste in continuare pe lac odata cu B_2. In acest caz, energia consumata de Robin va fi E_3.

Intrucat nu conteaza cat timp ii ia lui Robin sa ajunga la Maid Marian, atat timp cat se incadreaza in timpul T, el isi doreste sa o salveze consumand cat mai putina energie, pentru a putea apoi sa continue si cu jaful.

Nefiind foarte priceput la calcule, el va cere voua sa ii indicati energia minima pe care trebuie sa o consume pentru a o salva pe Maid Marian, precum si miscarile

pe care ar trebui sa le faca.

4.2 Date de intrare

Pe prima linie a fisierului **bonus.in** se gasesc 2 intregi T si N, reprezentand timpul pe care il are Robin la dispozitie, respectiv numarul de busteni de pe lac.

Pe cea de-a doua linie se gasesc 2 intregi reprezentand coordonatele la care se afla Maid Marian.

Pe cea de-a treia linie se gasesc 3 intregi reprezentand energiile E_1, E_2 si E_3 corespunzatoare celor trei tipuri de actiuni posibile.

Pe urmatoarele N linii se gasesc cate 4 intregi, reprezentand coordonatele capetelor bustenilor: $x_i^{start}, y_i^{start}, x_i^{end}, y_i^{end}$, $i \in 1..N$.

Pe urmatoarele N linii se gasesc cate T caractere, reprezentand directia in care se vor misca bustenii la fiecare moment de timp (m_i^j, $i \in \{1, .., N\}, j \in \{0, ..T-1\}$).

4.3 Date de ieșire

Pe prima linie a fisierului **bonus.out** se va afisa un intreg reprezentand energia minima pe care trebuie sa o consume Robin Hood pentru a reusi sa o salveze pe Maid Marian la timp.

Pe a doua linie se va afla numarul de miscari M pe care trebuie sa le faca Robin pentru a ajunge la destinatie. Vor urma apoi M linii care vor contine cele M miscari:

1. pentru o actiune de tipul "stat pe loc" se va afisa caracterul 'H'
2. pentru o actiune de miscare pe busteanul curent in una din directiile cardinale se va afisa caracterul corespunzator directiei respective: N/S/E/V
3. pentru o actiune de tipul "sarit pe alt bustean" se va afisa caracterul 'J' urmat de un spatiu liber si apoi de indexul busteanului pe care trebuie sa sara Robin.

4.4 Restricții și precizări

- $1 <= N <= 80$
- $1 <= T <= 400$
- $1 <= L <= 10$, unde L este lungimea maxima a unui bustean
- $1 <= E_1, E_2, E_3 <= 1000$
- $-500 <= x_i^{start}, x_i^{end}, y_i^{start}, y_i^{end} <= 500$
- Bustenii vor avea intotdeauna orientarea verticala sau orizontala ($x_i^{start} = x_i^{end}$ sau $y_i^{start} = y_i^{end}$, $\forall i \in \{1, ..., N\}$)
- $x_i^{start} <= x_i^{end}$ si $y_i^{start} <= y_i^{end}$
- $m_i^j \in \{N, S, E, V\}$; $\forall i \in \{1, ..., N\}, \forall j \in \{0, ..., T-1\}$
- Indexarea bustenilor se va considera de la 1 la N

- Coordonatele se considera in sens matematic: prin miscare la N se intelege cresterea coordonatei y, iar prin miscare la E cresterea coordonatei x.
- Orice solutie care obtine efortul minim va fi acceptata.

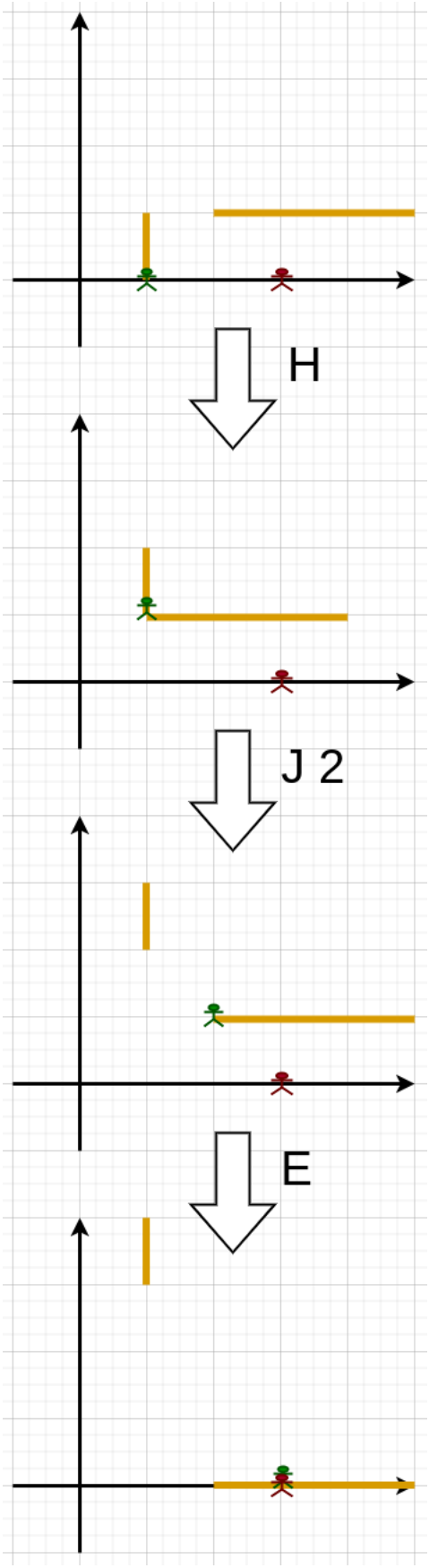
4.5 Testare și punctare

- Punctajul maxim este de **25** puncte.
- Timpul de execuție:
  - C/C++: **2 s**
  - Java: **4 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **bonus.c**, **bonus.cpp** sau **Bonus.java**.

4.6 Exemple

4.6.1 *Exemplu 1*

Exemplu 1		
bonus.in	bonus.out	Explicație
3 2 3 0 1 2 3 1 0 1 1 2 1 5 1 NNN VES	6 3 H J 2 E	Urmariti imaginea de pe pagina urmatoare. Robin Hood este reprezentat folosind culoarea verde, iar Maid Mirian folosind culoarea rosie.


H
J 2
E

5 PUNCTARE

- Punctajul temei este de 125 puncte, distribuit astfel:
  - Problema 1: 35p
  - Problema 2: 40p
  - Problema 3: 40p
  - 5 puncte vor fi acordate pentru comentarii si README
  - 5 puncte vor fi acordate pentru coding style in mod automat de catre checker
  - Se pot realiza depunctari de pana la 20 de puncte pentru coding style neadecvat la corectarea manuala a temelor.

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

Se poate obține un bonus de 25p rezolvând problema Robin Hood pe lac. Acordarea bonusului **NU** este condiționată de rezolvarea celorlate 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 o 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 va fi testata.
- Pentru testarea locala, aveți disponibil acelasi checker folosit si pe vmchecker pe pagina resurse a temei.
- Checkerul se poate rula fara niciun parametru, caz in care verifica toate problemele. De asemenea, se mai poate rula cu un parametru pentru a rula o anumita problema:

```
./check.sh <1 | 2 | 3 | 4>
./check cs      # pentru a testa doar coding style-ul
```

- **Punctajul pe teste** este cel de pe vmchecker.

- 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 corectare 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 in Java se recomanda folosirea **BufferedReader**.

6 FORMAT ARHIVĂ

• Temele vor fi testate automat pe vmchecker. Acesta suportă temele rezolvate în C/C++ și Java.
• Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_Tema2.zip** (ex: 399CX_PuiuGigel_Tema2.zip sau 399CX_BucurGigel_Tema2.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
- **clean**, care va șterge executabilele generate
- **run-p4**, care va rula executabilul pentru problema bonus (**doar dacă** ați implementat și bonusul)

• **ATENȚIE!** Daca folositi Java, se recomanda cresterea spatiului alocat stivei si heap-ului (recomandam 128MB) folosind flag-urile -Xmx -Xss la rulare.
Exemplu: `java -Xmx128m -Xss128m` Px

• **ATENȚIE!** Funcția **main** din rezolvarea unei probleme se va găsi într-o sursă ce trebuie obligatoriu denumită astfel:

- **p1.c**, **p1.cpp** sau **P1.java** - pentru problema 1
- **p2.c**, **p2.cpp** sau **P2.java** - pentru problema 2
- **p3.c**, **p3.cpp** sau **P3.java** - pentru problema 3
- **bonus.c**, **bonus.cpp** sau **Bonus.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 teme PA
- Google C++ Style Guide
- Google Java Style Guide
- Debugging și Structuri de Date