PA - TEMA 1
- GIGEL SI MUNTELE DE BANI -

Responsabili:
Cristian Pătrașcu, Andrei Preda, Ionela Nicuta,
Iustin Sirbu, David Iancu, Bianca Ciuche, Alex Delicostea

Deadline soft: **21.04.2020**
Deadline hard: **28.04.2020**

CUPRINS

1 Problema 1: Gigel și criptomonedele 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
1.6.1 Exemplu 1 . . . 4
2 Problema 2: Gigel investeste la bursă 5
2.1 Enunț . . . 5
2.2 Date de intrare . . . 5
2.3 Date de ieșire . . . 5
2.4 Restricții și precizări . . . 5
2.5 Testare și punctare . . . 5
2.6 Exemple . . . 6
2.6.1 Exemplu 1 . . . 6
3 Problema 3: Gigel merge la munte 7
3.1 Enunț . . . 7
3.2 Date de intrare . . . 7
3.3 Date de ieșire . . . 7
3.4 Restricții și precizări . . . 7
3.5 Testare și punctare . . . 8

3.6 Exemple . . . 8
3.6.1 Exemplu 1 . . . 8
3.6.2 Exemplu 2 . . . 8
4 Problema 4: Gigel si creasta montana 9
4.1 Enunț . . . 9
4.2 Date de intrare . . . 9
4.3 Date de ieșire . . . 9
4.4 Restricții și precizări . . . 9
4.5 Testare și punctare . . . 10
4.6 Exemple . . . 10
4.6.1 Exemplu 1 . . . 10
5 Problema 5 (bonus): Trigigel 11
5.1 Enunț . . . 11
5.2 Date de intrare . . . 11
5.3 Date de ieșire . . . 11
5.4 Restricții și precizări . . . 11
5.5 Testare și punctare . . . 11
5.6 Exemple . . . 12
5.6.1 Exemplu 1 . . . 12
6 Punctare 13
6.1 Checker . . . 13
7 Format arhivă 15
8 Links 16

1 PROBLEMA 1: GIGEL ȘI CRIPTOMONEDELE

1.1 Enunț

Gigel s-a hotărât că este timpul să se apuce de minat criptomonede G.
Și-a cumpărat N calculatoare și le-a conectat între ele, într-o rețea de minat.

În rețeaua sa, minatul reprezintă efectuarea unor calcule complexe. Primul calculator începe, îsi face partea lui, apoi dă mai departe rezultatul calculatorului C2, și imediat preia o nouă sarcină. C2 face același lucru, până la ultimul calculator. O monedă este produsă după ce ultimul calculator și-a terminat calculele pentru moneda respectivă. Fiecare calculator are o anumită putere de calcul, care îi permite să își facă partea lui de muncă pentru maxim P_i monede pe oră.

Gigel are niște bani strânși, cu care vrea să își mai cumpere procesoare pentru calculatoarele lui pentru a mina mai repede. Pentru fiecare calculator el știe cât costă un upgrade pentru a creste cu 1 numarul de monede la care un calculator poate contribui într-o oră, și anume U_i.

Care este numărul maxim de criptomonede pe care rețeaua lui Gigel le poate mina pe oră dupa ce își cumpără procesoare pentru calculatoarele sale?

Numărul de monede pe oră se măsoară după ce deja a fost produsă prima monedă.

1.2 Date de intrare

Pe prima linie a fișierului **crypto.in** se află 2 numere întregi: **N** si **B**.
Pe fiecare din urmatoarele N linii se află o pereche (P_i , U_i)

1.3 Date de ieșire

În fișierul **crypto.out** se va scrie numarul maxim de criptomonede pe care rețeaua lui Gigel le poate mina pe oră după ce își cumpară procesoare pentru calculatoarele sale.

1.4 Restricții și precizări

- $1 \leqslant N \leqslant 10^6$
- $1 \leqslant B \leqslant 10^9$
- $1 \leqslant Pi \leqslant 10^9$
- $1 \leqslant Ui \leqslant 10^9$

1.5 Testare și punctare

- Punctajul maxim este de **25** puncte.
- Timpul de execuție:
  - C/C++: **1 s**
  - Java: **4 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **crypto.c**, **crypto.cpp** sau **Crypto.java**.

1.6 Exemple

1.6.1 Exemplu 1

Exemplu 1		
crypto.in	crypto.out	Explicație
5 16 1 3 4 2 3 3 1 2 2 1	3	Inițial rețeaua produce o monedă pe oră. Gigel va face: 2 upgrade-uri pentru primul calculator, cu prețul 6, 2 upgrade-uri pentru calculatorul 4, cu prețul 4 si un upgrade pentru calculatorul 5, cu prețul 1. Astfel, rețeaua va putea mina 3 criptomonede pe oră. Prețul plătit este 11, care se încadrează în buget.

2 PROBLEMA 2: GIGEL INVESTESTE LA BURSĂ

2.1 Enunț

Pentru că în ultima perioada Gigel a stat doar acasa, a reușit să economisească suma de B dolari, pe care acum dorește să îi investească la bursa.

El a găsit online o listă cu N acțiuni studiate de cei mai buni specialiști ai lumii. Ei au determinat pentru fiecare acțiune prețul minim și prețul maxim pe care aceasta îl va putea avea la sfârșitul anului.

Convins că specialiștii nu pot să greșească, Gigel s-a hotărât să cumpere acțiuni din această listă și să vândă tot la final de an. El dorește să aleagă niște acțiuni astfel încât în cel mai rău scenariu posibil, să nu piardă mai mult de L dolari și să obțină profitul maxim în cel mai bun caz posibil.

Pentru că a auzit că este bine să îți diversifici portofoliul de actiuni, el a decis că nu va cumpara mai mult de o acțiune de un anumit tip. De asemenea, pe platforma online pe care Gigel tranzacționează, se pot cumpăra doar unități întregi (nu poate cumpăra fracții precum 0.3 dintr-o acțiune).

Care este profitul maxim pe care îl poate obtine Gigel?

2.2 Date de intrare

Pe prima linie a fișierului **stocks.in** se află **N**, **B** și **L** .

Pe fiecare din urmatoarele **N** linii se află 3 numere, currentValue, minValue si maxValue.

2.3 Date de ieșire

În fișierul **stocks.out** se va afla profitul maxim pe care il poate obtine Gigel

2.4 Restricții și precizări

- $1 \leqslant \mathbf{N} \leqslant 100$
- $1 \leqslant \mathbf{B} \leqslant 500$
- $1 \leqslant \mathbf{L} \leqslant 500$

2.5 Testare și punctare

- Punctajul maxim este de **30** puncte.

- Timpul de execuție:
  - C/C++: **1 s**
  - Java: **1 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **stocks.c**, **stocks.cpp** sau **Stocks.java**.

2.6 Exemple

2.6.1 Exemplu 1

Exemplu 1		
stocks.in	stocks.out	Explicație
5 30 6 9 7 17 12 5 16 7 6 8 3 2 6 5 2 6	12	Buget = 30, maxLoss = 6 Gigel va alege acțiunile 1 3 si 5. Va investi 9 + 7 + 3 = 19 dolari < Buget În cel mai rau caz, actiunile vor valora 7 + 6 + 2 = 15 dolari. 19 - 15 = 4. Mai mic decat maxLoss In cel mai bun caz, actiunile vor valora 17 + 8 + 6 = 31 de dolari. 31 - 19 = 12 dolari profit.

3 PROBLEMA 3: GIGEL MERGE LA MUNTE

3.1 Enunț

Cu banii câștigați din afaceri, Gigel și-a cumpărat excavator. Succesul l-a determinat să mute munții (la propriu), așa că s-a apucat de excavat o vale în care să își construiască statuie.

Zona montană e reprezentată sub forma unui șir de numere: înălțimile vârfurilor care o compun.

Numim "vale" un șir de numere care poate fi impărțit în două jumătăți (prefix și sufix), și care îndeplinește următoarele condiții:

- prefixul este descrescător
- sufixul este crescător
- prefixul și sufixul ocupă cel puțin două poziții
- prefixul și sufixul au o poziție în comun.

De exemplu, șirul [4, 3, 2, 5, 6] este o "vale" datorită împărțirii în prefixul [4, 3, 2] și sufixul [2, 5, 6]. Pe de altă parte, șirul [1, 2, 3] nu este o "vale", deoarece nu are un prefix valid (este o rampă).

Gigel poate excava o unitate din înălțimea unui vârf într-o oră. Câte ore va dura formarea văii dacă vrea să termine cât mai repede?

3.2 Date de intrare

Pe prima linie a fișierului **valley.in** se află **N**.

Pe urmatoarea linie se află **N** numere, reprezentand inaltimile varfurilor de munte.

3.3 Date de ieșire

În fișierul **valley.out** se va afla numarul minim de ore in care Gigel poate forma valea.

3.4 Restricții și precizări

- $3 \leqslant N \leqslant 10^6$
- $1 \leqslant h_i \leqslant 10^9$

3.5 Testare și punctare

- Punctajul maxim este de **30** puncte.
- Timpul de execuție:
  - C/C++: **1 s**
  - Java: **3 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **valley.c**, **valley.cpp** sau **Valley.java**.

3.6 Exemple

3.6.1 Exemplu 1

Exemplu 1		
valley.in	valley.out	Explicație
5 3 4 5 2 4	3	Excavând 3 unități de la pozițiile 2 și 3, Gigel poate obține șirul [3, 3, 3, 2, 4], care este o vale, putând fi împărțit în [3, 3, 3, 2] și [2, 4].

3.6.2 Exemplu 2

Exemplu 2		
valley.in	valley.out	Explicație
4 3 3 3 3	0	Șirul este deja vale.

4 PROBLEMA 4: GIGEL SI CREASTA MONTANA

4.1 Enunț

Deși ați facut o treabă foarte bună cu prima cerință a lui Gigel legată de munte, acesta s-a răzgândit.

Nu mai vrea să facă o vale, ci vrea diversitate. Își ia excavatorul și se apucă din nou de săpat. La fel ca și prima dată, zona montană este reprezentat de un șir de numere naturale, fiecare numar reprezentând înălțimea muntelui i.

Acum el vrea să sape munții astfel încât să nu existe doi munți adiacenți de aceeași înălțime. Totuși el nu vrea să sape sub pământ, deci înălțimile munților nu pot să ajungă mai mici ca 0.

Pentru că a mai trecut timpul, a crescut prețul combustibilului și de asemenea, unii munți au devenit mai tari și mai greu de săpat. Acum pentru fiecare săpătură într-un munte, prin care înălțimea acestuia scade cu 1, Gigel trebuie să plătească pe combustibil un cost Ci.

Care este costul minim cu care Gigel poate să obțină creasta montană mult visată?

4.2 Date de intrare

Pe prima linie a fișierului **ridge.in** se află **N**, numarul de munți.

Pe următoarele N linii se află câte o pereche de numere H_i, C_i reprezentând înălțimea muntelui i și costul pentru a micșora cu 1 muntele i.

4.3 Date de ieșire

În fișierul **ridge.out** se va afla un număr, reprezentând costul minim cu care Gigel poate obține creasta mult dorita.

4.4 Restricții și precizări

- $N \leqslant 10^6$.
- $H_i, C_i \leqslant 10^9$.

4.5 Testare și punctare

- Punctajul maxim este de **30** puncte.
- Timpul de execuție:
  - C/C++: **1.5 s**
  - Java: **4 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **ridge.c**, **ridge.cpp** sau **Ridge.java**.

4.6 Exemple

4.6.1 Exemplu 1

Exemplu 1		
ridge.in	ridge.out	Explicație
4 3 2 3 3 4 2 4 1	3	Gigel va săpa o unitate din primul munte, cu costul 2 și o unitate din ultimul munte, cu costul 1. Se obține secvența de înălțimi 2 3 4 3.

5 PROBLEMA 5 (BONUS): TRIGIGEL

5.1 Enunț

Pentru a face și mai mulți bani, Gigel a venit cu o nouă idee. Calculatoarele pe care le are el pentru a mina criptomonede folosesc logica bivalentă (operații cu biți de 0 și 1). El vrea să își construiască un calculator ce foloseste logica trivalentă pentru a mina mai eficient. Pentru asta, el vrea să înțeleagă mai bine numerele 0 1 și 2.

Si-a scris un sir de lungime N cu numerele 0, 1 si 2, format din secventa 012, care se repeta. Pentru N = 7, sirul sau este 0120120. Pentru N = 13, sirul este 0120120120120.

El vrea să vadă în câte moduri poate să aleagă o mulțime de indici ai acestui șir, astfel încât subșirul format de numerele de la indicii aleși să fie o subsecvență a șirului inițial.

Pentru că rezultatul poate fi foarte mare, dorește afișarea sa modulo 1000000007.

5.2 Date de intrare

Pe prima linie a fișierului **trigigel.in** se află **N**.

5.3 Date de ieșire

În fișierul **trigigel.out** se va afla numarul pe care Gigel își dorește să il afle.

5.4 Restricții și precizări

- $N \leqslant 10^6$. pentru teste în valoare de 10 puncte.
- $N \leqslant 10^{12}$ pentru celelalte teste, în valoare de 15 puncte.

5.5 Testare și punctare

- Punctajul maxim este de **25** puncte.
- Timpul de execuție:
  - C/C++: **1 s**
  - Java: **2 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **ridge.c**, **ridge.cpp** sau **Ridge.java**.

5.6 Exemple

5.6.1 Exemplu 1

Exemplu 1		
trigigel.in	trigigel.out	Explicație
5	16	Șirul este 01201 Putem alege următoarele mulțimi de indici (indexare de la 0) 0 -> subșirul 0 1 -> subșirul 1 2 -> subșirul 2 3 -> subșirul 0 4 -> subșirul 1 0, 1 -> subșirul 01 0,4 -> subșirul 01 1,2 -> subșirul 12 2,3 -> subșirul 20 3,4 -> subșirul 01 0,1,2 -> subșirul 012 1,2,3 -> subșirul 120 2,3,4 -> subșirul 120 0,1,2,3 -> subșirul 0120 1,2,3,4 -> subșirul 1201 0,1,2,3,4 -> subșirul 01201

6 PUNCTARE

- Punctajul temei este de **125** puncte, distribuit astfel:
  - Problema 1: **25p**
  - Problema 2: **30p**
  - Problema 3: **30p**
  - Problema 4: **30p**
  - 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 pană la 20 de puncte** pentru **coding style neadecvat**.

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 TriGigel. 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.

6.1 Checker

- Arhiva se va trimite pe vmchecker, unde tema se va testa folosind un set de teste private.
- Pentru testarea locala, aveți disponibil un set de teste publice (de aceeași dificultate) pe pagina cu resurse a temei.
- Checkerul se poate rula fara niciun parametru, caz in 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 | 5>
  ./check.sh <crypto | stocks | valley | ridge | trigigel >
  ./check.sh cs

- **Punctajul pe teste** este cel de pe vmchecker și se acordă ruland 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 in Java se recomanda folosirea **BufferedReader**.

7 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 sau 399CX_BucurGigel_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 4
  - **run-p5**, care va rula executabilul pentru problema 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:

  - **crypto.c**, **crypto.cpp** sau **Crypto.java** - pentru problema 1
  - **stocks.c**, **stocks.cpp** sau **Stocks.java** - pentru problema 2
  - **valley.c**, **valley.cpp** sau **Valley.java** - pentru problema 3
  - **ridge.c**, **ridge.cpp** sau **Ridge.java** - pentru problema 4
  - **trigigel.c**, **trigigel.cpp** sau **Trigigel.java** - pentru problema 5

- **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).

8 LINKS

- Regulament general teme PA
- Google C++ Style Guide
- Google Java Style Guide
- Debugging și Structuri de Date