PA - TEMA 1
- ANOTHER DAY IN GIGELAND -

Responsabili:
Ștefania Budulan, Radu Visan,
Cosmin Ciocan, Adrian Harmasel, Oana Iliescu, Cristian Patrascu
Responsabil checker: Cosmin Ciocan
Responsabil infrastructura: Darius Neatu

Autor: Cristian Patrascu

Deadline soft: **09.04.2020**
Deadline hard: **16.04.2020**

CUPRINS

1 PROBLEMA 1: GIGEL TREZORIER LA BNR

1.1 Enunț

Gigel s-a angajat ca trezorier la BNR. Pentru primul sau task el are de pus in seiful numarul 42 al băncii N bancnote. El primește o hârtie cu instrucțiuni de aranjare a bancnotelor.

Exista 2 seturi de instrucțiuni:

Tipul 1:

(a) Dupa o bancnota de 10 lei, va urma mereu una de 50 lei sau una de 100 de lei.

(b) Dupa o bancnota de 50 lei, va urma mereu una de 10 lei sau de 200 de lei.

(c) Dupa o bancnota de 100 de lei, va urma mereu una de 10 lei sau una de 100 de lei.

(d) Dupa o bancnota de 200 de lei, va urma mereu una de 50 sau 500 de lei.

(e) Dupa o bancnota de 500 de lei, va urma mereu una de 200 de lei sau una de 10 lei.

Tipul 2:
Instrucțiunea (d) din Tipul 1 este înlocuită cu instrucțiunea:
"Dupa o bancnota de 200 de lei, va urma mereu una de 50, 100 sau 500 de lei."

Acum Gigel vrea să știe în câte moduri poate aranja bancnotele.
Banca are bancnote nelimitate din fiecare valoare.

1.2 Date de intrare

Pe prima linie a fișierului **bani.in** se află 2 numere întregi: set instrucțiuni, **N**.

1.3 Date de ieșire

În fișierul **bani.out** se va scrie numarul de moduri in care Gigel poate aranja bancnotele.

Pentru că rezultatul poate fi mare, se dorește restul modulo 1000000007.

1.4 Restricții și precizări

- Pentru setul de instructiuni de tipul 1: $1 \leqslant N \leqslant 10^9$
- Pentru setul de instructiuni de tipul 2: $1 \leqslant N \leqslant 10^6$

1.5 Testare și punctare

- Punctajul maxim este de **30** puncte.
- Pentru teste in valoare de **10** puncte, setul de instrucțiuni este de tipul 1.
- Timpul de execuție:
  - C/C++: **2 s**
  - Java: **2 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **bani.c**, **bani.cpp** sau **Bani.java**.

1.6 Exemple

1.6.1 Exemplu 1

Exemplu 1		
bani.in	bani.out	Explicație
1 3	20	(10, 50, 10), (10, 50, 200), (10, 100, 10), (10, 100, 100), (50, 10, 50), (50, 10, 100), (50, 200, 50), (50, 200, 500), (100, 10, 50), (100, 10, 100), (100, 100, 10), (100, 100, 100), (200, 50, 10), (200, 50, 200), (200, 500, 200), (200, 500, 10), (500, 200, 50), (500, 200, 500), (500, 10, 50), (500, 10, 100)

1.6.2 Exemplu 2

Exemplu 1		
bani.in	bani.out	Explicație
2 2	11	(10, 50), (10, 100), (50, 10), (50, 200), (100, 10), (100, 100), (200, 50), (200, 100), (200, 500), (500, 200), (500, 10)

2 PROBLEMA 2: GIGEL SI GARDUL

2.1 Enunț

Cu banii câștigați la bancă, Gigel s-a hotărât să își construiască un gard în fața casei. Pentru asta, el și-a cumpărat N bucăți de gard și le-a pus la întâmplare. A doua zi a constatat ca unele bucati erau in plus.

Acum el vrea să dărâme bucățile redundante. A notat coordonatele capetelor bucatilor de gard $(xStart_i, xEnd_i)$ și v-a rugat pe voi să îi spuneți câte bucăți sunt redundante. O bucată de gard este redundantă dacă există o altă bucată care o include complet. Gardul (xStart, xEnd) este inclus în gardul (yStart, yEnd) daca yStart $\leqslant$ xStart si xEnd $\leqslant$ yEnd

2.2 Date de intrare

Pe prima linie a fișierului **gard.in** se află **N**.
Pe fiecare din urmatoarele **N** linii se află o pereche de numere $(xStart_i, xEnd_i)$

2.3 Date de ieșire

În fișierul **gard.out** se va afla numărul de bucați de gard redundante.

2.4 Restricții și precizări

- $xStart_i \leqslant xEnd_i$
- $0 \leqslant xStart_i, xEnd_i \leqslant 10^9$
- $\mathbf{N} \leqslant 10^5$

2.5 Testare și punctare

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

2.6 Exemple

2.6.1 Exemplu 1

Exemplu 1		
gard.in	gard.out	Explicație
5 0 6 4 12 5 10 2 4 9 12	3	Gardurile redundante sunt (2,4) (5,10) si (9, 12)

3 PROBLEMA 3: GIGEL BOMBONEL

3.1 Enunț

Gigel în timpul liber este profesor de algoritmică. El le-a promis celor N elevi ai săi la începutul anului că o să le dea bomboane în funcție de numărul de probleme de algoritmică pe care o să le rezolve.

Pentru fiecare elev avea notat pe o foaie câte probleme a rezolvat. A mers la magazin și a luat bomboane pentru toți, în total M bomboane, dar pe drum spre casă a pierdut foaia. Cum Gigel are o memorie foarte bună, a reținut pentru fiecare elev "cam câte" probleme a rezolvat, și anume un interval $[X_i, Y_i]$. Acum el vrea să ofere fiecărui elev un număr de bomboane ce este inclus în intervalul corespunzător.

Ajutați-l pe Gigel să afle în câte moduri poate oferi bomboane elevilor!

3.2 Date de intrare

Pe prima linie a fișierului **bomboane.in** se află **N** și **M**.

Pe următoarele **N** linii se află intervalul corespunzător fiecărui elev.

3.3 Date de ieșire

În fișierul **bomboane.out** se va afla numărul de moduri în care poate oferi Gigel bomboane elevilor.

Pentru că rezultatul poate fi mare, se dorește restul modulo 1000000007.

3.4 Restricții și precizări

- $1 \leqslant N \leqslant 50$
- $1 \leqslant M \leqslant 2000$
- $0 \leqslant X_i, Y_o \leqslant 500$

3.5 Testare și punctare

- Punctajul maxim este de **45** puncte.
- Timpul de execuție:
  - C/C++: **2 s**
  - Java: **2 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **bomboane.c**, **bomboane.cpp** sau **Bomboane.java**.

3.6 Exemple

3.6.1 Exemplu 1

Exemplu 1		
bomboane.in	bomboane.out	Explicație
3 6 1 3 0 2 2 4	7	Posibilitățile sunt: (1,1,4), (1,2,3), (2, 0, 4), (2,1,3), (2,2,2), (3,0,3), (3,1,2)

3.6.2 Exemplu 2

Exemplu 2		
bomboane.in	bomboane.out	Explicație
2 4 0 2 0 2	1	Singura posibilitate este (2,2).

3.6.3 Exemplu 3

Exemplu 3		
bomboane.in	bomboane.out	Explicație
3 20 2 4 4 5 3 7	0	Nu exista niciun mod de a imparti toate bomboanele.

4 BONUS: GIGEL LA SALĂ

4.1 Enunț

Pentru că vrea să impresioneze fetele, Gigel s-a apucat de sală.

La sală sunt N gantere de diverse greutăți (G_i). Pentru fiecare ganteră, Gigel știe câte repetări poate să facă cu ea (R_i). Gigel vrea să folosească **maxim** M gantere distincte din cele N din sală, pentru a-și crește mușchii cât mai mult.

Mușchii săi vor crește cu

$$numarTotalRepetari * greutateaCeleiMaiUsoareGantereFolosite$$

Ajutați-l pe Gigel să afle cu cat poatel să își crească maxim mușchii.

Formal, mușchii lui Gigel vor creste cu $(\sum_{i=1}^{M} R_i) * min(G_i)$, unde R_i si G_i reprezintă numărul de ridicări, respectiv greutățile ganterelor alese.

Print gantere **distincte** se intelege faptul că nu poate folosi de 2 ori aceeași ganteră. În schimb, poate folosi ,daca vrea, doua gantere i și j care au aceeasi greutate si cu care poate face acelasi numar de repetări.

4.2 Date de intrare

Pe prima linie a fișierului **sala.in** se află **N** și **M**.

Pe următoarele N linii se află câte o pereche de numere reprezentând greutatea și numărul de repetări pentru fiecare ganteră.

4.3 Date de ieșire

În fișierul **sala.out** se va afla un număr, reprezentând cu cât poate Gigel să își crească maxim mușchii.

4.4 Restricții și precizări

- N, M, R_i, $G_i \leqslant 10^6$.

4.5 Testare și punctare

- Punctajul maxim este de **25** puncte.

- Timpul de execuție:
  - C/C++: **2.5 s**
  - Java: **3.5 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **sala.c**, **sala.cpp** sau **Sala.java**.

4.6 Exemple

4.6.1 Exemplu 1

Exemplu 1		
sala.in	sala.out	Explicație
6 3 3 8 5 3 6 4 2 10 4 7 9 4	60	Gigel va alege ganterele 2, 4 si 5 (indexat de la 0) Minimul greutatilor celor 3 gantere este min(6, 4, 9) = 4 Numarul total de repetari este 4 + 7 + 4 = 15 Raspunsul este 4 * 15 = 60

5 PUNCTARE

- Punctajul temei este de 125 puncte, distribuit astfel:
  - Problema 1: 30p
  - Problema 2: 40p
  - Problema 3: 45p
  - 5 puncte vor fi acordate pentru comentarii și README
  - 5 puncte vor fi acordate pentru coding style in mod automat de catre checker.
  - Se pot realiza depunctari de pană la 20 de puncte pentru coding style neadecvat la corectarea manuală 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 Gigel la sală. 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 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>
./check.sh <bani | gard | bomboane | sala >
./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 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 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
  - **clean**, care va șterge executabilele generate
  - **run-p4**, care va rula executabilul pentru problema bonus (**doar dacă** ați implementat și bonusul)

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

  - **bani.c**, **bani.cpp** sau **Bani.java** - pentru problema 1
  - **gard.c**, **gard.cpp** sau **Gard.java** - pentru problema 2
  - **bomboane.c**, **bomboane.cpp** sau **Bomboane.java** - pentru problema 3
  - **sala.c**, **sala.cpp** sau **Sala.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