PROBLEME propuse pentru REZOLVARE

Vă prezentăm enunțurile celor șase probleme propuse spre rezolvare concurenților de la Olimpiada Balcanică de Informatică, desfășurată în perioada 8-13 septembrie 2001 în stațiunea Durres din Albania.

P080101: Mozaic

Presupunem că avem un mozaic într-o zonă dreptunghiulară H formată din $M \times N$ celule dispuse în M linii și N coloane. În acesta este încorporat un model original. Acesta se află într-o altă zonă dreptunghiulară O, formată din $P \times Q$ celule dispuse în P linii și Q coloane. Toate celulele din cele două zone sunt colorate în alb sau negru ($1 < M, N, P, Q \leq 100$).

Toate celulele negre ale zonei dreptunghiulare O fac parte din mozaic. Toate celulele albe situate între două celule negre pe una din direcțiile orizontală, verticală sau diagonală fac parte din mozaic.

Dacă se dă mozaicul original în zona O, scrieți un program care determină dacă în zona dreptunghiulară H se regăsește o copie identică a acestuia (având aceeași orientare, fără a fi rotit). În caz afirmativ, veți scrie `YES` pe prima linie a fișierului de ieșire, iar în linia a doua două numere întregi care reprezintă coordonatele primei celule a mozaicului. Prima celulă a mozaicului este considerată a fi cea care are coordonata cea mai mică a liniei și, în cadrul liniei respective, coordonata cea mai mică a coloanei.

Date de intrare

Fișierul de intrare **`MOSAIC.IN`** conține pe prima linie două numere întregi P și Q, reprezentând dimensiunile zonei dreptunghiulare în care se află mozaicul original O.

Următoarele P linii conțin fiecare câte Q numere întregi, cu valori `0` sau `1`. Valoarea `0` reprezintă o celulă de culoare albă, iar valoarea `1` reprezintă o celulă de culoare neagră. Numerele sunt separate prin câte un spațiu.

Următoarea linie conține două numere întregi M și N care reprezintă dimensiunile zonei dreptunghiulare H, în care se presupune că există o copie a mozaicului. Urmează apoi M linii, conținând fiecare câte N numere întregi din mulțimea {`0`, `1`}.

Date de ieșire

Fișierul de ieșire **`MOSAIC.OUT`** va avea una sau două linii. În prima linie a fișierului se va scrie `YES` sau `NO`, în funcție de existența unei copii a mozaicului din zona O în zona dreptunghiulară H.

Dacă prima linie conține `YES`, în linia a doua vor fi scrise două numere întregi, separate printr-un spațiu, reprezentând coordonatele primei celule negre a copiei mozaicului.

Exemple

`MOSAIC.IN`
```
7 8
0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 1
0 1 1 0 0 1 1 0
0 0 1 1 1 1 0 1
0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0
8 9
1 1 0 0 1 1 1 1 0
0 0 0 0 0 0 1 1 1
0 0 0 1 1 0 0 1 0
1 1 1 1 1 1 0 1 0
1 1 1 0 0 1 1 0 1
1 0 1 1 1 1 0 1 0
1 1 0 1 1 0 0 0 1
1 1 0 0 0 0 1 1 0
```

`MOSAIC.OUT`
```
YES
3 4
```

`MOSAIC.IN`
```
3 3
0 1 0
1 0 1
0 1 0
3 3
0 1 0
1 1 1
0 1 0
```

`MOSAIC.OUT`
```
NO
```

Timp maxim de execuție/test: 1 secundă

P080102: Șirul secret

Trebuie să descoperiți un șir secret format din patru cifre din mulțimea {0, 1, …, 9}. Pentru a descoperi acest șir, la început veți presupune că este un șir oarecare. Apoi veți apela rutina `ShowFit()`, care va compara acest șir cu cel secret și vă va arăta cum se ***potrivesc*** cele două șiruri. Rutina are doi parametri (`rp` și `wp`) transmiși prin referință; după executarea rutinei, ei vor avea valori cuprinse între `0` și `4`. Parametrul `wp` va indica numărul de cifre comune celor două șiruri, dar care nu ocupă aceleași poziții în aceste șiruri. Parametrul `rp` va indica numărul de cifre comune celor două șiruri, care ocupă aceleași poziții.

De exemplu, dacă șirul secret este `173`**`3`**, atunci pentru șirul presupus `237`**`3`** rutina `ShowFit()` va furniza valorile `wp=2` (deoarece în ambele șiruri există un `7` și un `3`, dar nu pe aceleași poziții), și `rp=1` (deoarece există un `3` care ocupă în ambele șiruri aceeași poziție).

Scrieți un program care, utilizând rutina `ShowFit()`, descoperă șirul secret. Procedura nu poate fi apelată decât de cel mult 10 ori.

Date de ieșire

Fișierul de ieșire **`SECRET.OUT`** va avea o singură linie care va conține șirul secret.

Exemplu

Șiruri presupuse	Valorile `wp` și `rp`
`1234`	`1 1`
`2134`	`0 2`
`1134`	`1 2`
`1131`	`0 3`
`2131`	`0 3`
`3131`	`0 4`

deci `3131` este șirul secret.

`SECRET.OUT`
```
3131
```

Instrucțiuni pentru programatorii în Pascal

Programul vostru trebuie să conțină declarația **`uses`** `Show;`

Apelul procedurii `ShowFit()` trebuie să fie de forma:

```
ShowFit(guess,wp,rp);
```

unde variabila `guess` are tipul **`string`**`[4]`, iar variabilele `wp`, `rp` au tipul `Integer`.

Instrucțiuni pentru programatorii în C/C++

Va trebui să creați un proiect în care veți include fișierul sursă și fișierul **`SHOWFIT.OBJ`** care vă este pus la dispoziție.

În programul sursă trebuie să includeți fișierul header **`SHOWFIT.H`**.

Apelul funcției `ShowFit()` trebuie să fie de forma:

```
showfit(guess,&wp,&rp);
```

unde parametrul `guess` are tipul **`char`**`[4]`, iar parametrii `wp` și `rp` au tipul **`int`** și sunt transmiși prin referință.

Timp maxim de execuție/test: 1 secundă

P080103: Vizită

Un birou de consultanță este organizat astfel: toți clienții stau la o coadă de așteptare, iar consultarea se realizează conform principiului *primul sosit primul servit*. Consultantul unui client este determinat de sexul clientului anterior (femeie - `F` sau bărbat - `M`). Astfel, consultantul C_i, după ce primește un client, în funcție de sexul acestuia (`F`, `M`) direcționează următorul client la consultantul C_j, dacă clientul curent este femeie (`F`) sau la consultantul C_k, dacă clientul curent este bărbat (`M`). După aceea, consultantul ales va servi următorul client.

Un client isteț a descoperit că există cozi în care este posibil ca indiferent de consultantul cu care începe consultarea, parcurgând toți clienții dinaintea sa, el să fie consultat de același consultant, C_m.

Scrieți un program care determină dacă acest lucru este posibil și, în caz afirmativ, determină o coadă de lungime minimă care reprezintă clienții aflați în fața clientului isteț și consultantul care îl va trata.

Dacă o astfel de coadă nu există, programul vostru va furniza mesajul: `'Impossible'`.

Consultanții sunt reprezentați folosind numere întregi cuprinse între 1 și N.

Date de intrare

Fișierul de intrare **`VISIT.IN`** conține pe prima linie un număr întreg N ($2 < N \leq 40$), care reprezintă numărul de consultanți din birou.

Fiecare din următoarele N linii conțin câte trei numere întregi. Primul număr întreg reprezintă consultantul curent, al doilea întreg reprezintă consultantul care va continua consultarea în cazul în care clientul care a fost consultat a fost femeie, iar al treilea întreg reprezintă consultantul care va continua consultarea în cazul în care clientul consultat a fost bărbat.

Date de ieșire

Fișierul de ieșire **`VISIT.OUT`** va conține trei linii, dacă o astfel de coadă există, și o singură linie în caz contrar.

În cazul în care coada există, prima linie a fișierului de ieșire va conține lungimea cozii, pe linia a doua se va afla un șir de caractere `M` și `F` reprezentând coada, iar linia a treia va conține consultantul care va consulta clientul isteț.

În cazul în care coada nu există, fișierul va conține o singură linie care va conține mesajul `'Impossible'`.

Exemple

`VISIT.IN`
```
5
1 2 4
2 4 1
3 2 3
4 1 4
5 3 5
```

`VISIT.OUT`
```
4
FFMM
4
```

Timp maxim de execuție/test: 5 secunde

P080104: Joc

Doi jucători joacă următorul joc: se dă o grămadă care conține N bețe ($4 \le N \le 2000$); jucătorii pot lua din grămadă, pe rând, un număr de bețe; jucătorul care ia ultimul băț câștigă jocul.

Jocul se desfășoară respectându-se următoarele reguli:

- Jucătorul care începe jocul poate lua orice număr L_1 de bețe, dar nu întrega grămadă ($1 \le L_1 \le N - 1$).
- Apoi, cei doi jucători iau pe rând unul sau mai multe bețe, dar nu mai mult decât dublul numărului de bețe luate de celălalt jucător la mutarea precedentă. Astfel, jucătorul care este la mutare poate lua din grămadă L_2 bețe, unde $1 \le L_2 \le 2 \cdot L_1$ (L_1 este numărul de bețe luat de celălalt jucător la mutarea precedentă).

Scrieți un program care joacă acest joc împotriva calculatorului și câștigă. În cazul în care strategia voastră este corectă și sunteți jucătorul care începe jocul, veți câștiga întotdeauna.

La început se va citi din fișierul **GAME.IN**, numărul N al bețelor din grămadă, apoi trebuie să se determine numărul L_1 al bețelor pe care le extrage din grămadă primul jucător (programul vostru).

Folosind aceste valori, se va apela rutina `move(N1,L1,N2,L2);`. Variabila `N1` reprezintă numărul de bețe care se află în grămadă înaintea efectuării mutării, iar variabila `L1` reprezintă numărul de bețe pe care primul jucător le extrage din grămadă. După executarea rutinei, variabila `N2` va indica numărul de bețe care rămân în grămadă după mutarea primului jucător (programul vostru), iar variabila `L2` va indica numărul de bețe pe care le extrage al doilea jucător (calculatorul) din grămadă. Se observă că avem întotdeauna `N2=N1-L1` și că la primul apel `N1=N`.

După apelarea rutinei, va trebui să determinați următorul număr al bețelor care vor fi extrase, precum și numărul bețelor care au rămas după mutarea calculatorului. Evident, acest ultim număr va fi întotdeauna `N1=N2-L2`. Veți apela din nou rutina `move()`, și acești pași se vor repeta până când nu mai există nici un băț în grămadă. Jucătorul care extrage ultimul băț câștigă jocul.

Exemplu

GAME.IN

```
10
```

Mutarea programului		*Mutarea calculatorului*	
N1	L1	N2	L2
10	2	8	1
7	2	5	1
4	1	3	1
2	2	0	

Notă

Programul vostru trebuie să respecte restricțiile jocului. În cazul în care o mutare nu este corectă, programul se va încheia automat. Rutina `move()` verifică fiecare mutare. În cazul în care strategia folosită nu este optimă, calculatorul va câștiga jocul.

Instrucțiuni pentru programatorii în Pascal

Programul vostru trebuie să conțină declarația **uses** `moveun;`

Apelul procedurii `move()` trebuie să fie de forma:

```
move(N1,L1,N2,L2);
```

unde variabilele `N1`, `L1`, `N2` și `L2` au tipul `Integer`.

La început se va apela, o singură dată, procedura `newgame`. Aceasta nu are nici un parametru.

Instrucțiuni pentru programatorii C/C++

Va trebui să creați un proiect în care veți include fișierul sursă și fișierul **MOVEUN.OBJ** care vă este pus la dispoziție.

În programul sursă trebuie să includeți fișierul header **MOVEUN.H**, precum și declarațiile:

```
void newgame(void);
void move(int,int,int&,int&);
```

Apelul funcției `move()` trebuie să fie de forma:

```
move(N1,L1,&N2,&L2);
```

unde variabilele `N1`, `L1`, `N2` și `L2` au tipul **int**.

La început se va apela, o singură dată, funcția `newgame()`.

Timp maxim de execuție/test: 1 secundă

P080105: Orar

Un laborator acceptă vizitatori de la firmele interesate de utilizarea aparaturii proprii, doar două ore pe zi, într-un interval fix de zile.

Fiecare firmă trebuie să-și determine propriul orar pentru utilizarea laboratorului exact două ore pe zi. Aceste două ore pot fi în prima perioadă a zilei, în a doua perioadă sau în a treia perioadă.

Orarul este fixat, de la început, pentru toată perioada celor N zile și trebuie să respecte următoarele condiții:

- dacă o firmă selectează într-o zi a treia perioadă, atunci este obligată ca în următoarea zi să selecteze prima perioadă;
- prima perioadă poate fi selectată numai dacă în ziua precedentă a fost selectată perioada a treia.

Pentru un număr de zile N dat ($1 \le N \le 44$), scrieți un program care calculează numărul M_N al posibilităților de întocmire a orarului.

Exemplu

Pentru $N = 4$, cele $M_N = 5$ posibilități de întocmire a orarului sunt:

Ziua	**1**	**2**	**3**	**4**
Orar 1	2	2	2	2
Orar 2	3	1	3	1
Orar 3	2	3	1	2
Orar 5	2	2	3	1
Orar 5	3	1	2	2

Se observă că, respectând condițiile, nu se poate programa prima perioadă în prima zi și a treia perioadă în ultima zi.

Date de intrare
Fișierul de intrare **SCHEDULE.IN** conține o singură linie pe care se află numărul întreg N, care reprezintă numărul de zile.

Date de ieșire
Fișierul de ieșire **SCHEDULE.OUT** va conține o singură linie pe care se va afla numărul întreg M_N, care reprezintă numărul de posibilități de întocmire a orarului.

Timp maxim de execuție/test: 1 secundă

P080106: Votare

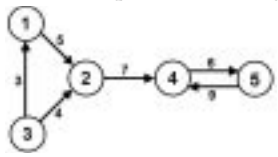
Într-o zonă care are N ($3 \le N \le 50$) localități, trebuie să fie construite centre de votare. Într-un centru de votare pot vota doar locuitorii din localitățile care sunt legate prin drumuri cu toate celelate localități ale căror locuitori votează în același centru.

Se cunosc localitățile și drumurile care le leagă. Drumurile sunt date într-o matrice D având elemente întregi, unde d_{ij} reprezintă lungimea drumului care leagă localitatea i de localitatea j, ($d_{ij} = 0$ are semnificația: nu există drum între localitatea i și localitatea j). Toate drumurile sunt cu sens unic.

Programul vostru trebuie să determine numărul minim K al zonelor de votare care pot fi formate.

După aceea, pentru fiecare zonă, trebuie determinate localitățile P_s ($s = 1, 2, ..., K$), în care trebuie construite centrele de votare, astfel încât suma lungimilor tuturor drumurilor de la fiecare localitate la acest centru să fie minimă. În cazul în care, pentru o zonă de votare există mai multe localități în care se poate construi centrul de votare care satisface condițiile de minim, se va determina una din ele.

Să considerăm zona reprezentată în figura următoare:



În acest caz există două zone de votare: (1, 2, 3) și (4, 5). Centrele de votare sunt în localitățile 1 și 5.

Date de intrare
Fișierul de intrare **VOTING.IN** conține pe prima linie numărul întreg N, care reprezintă numărul de localități. Următoarele N linii conțin fiecare câte N numere întregi, separate prin spații, care reprezintă lungimile drumurilor.

Date de ieșire
Fișierul de ieșire **VOTING.OUT** trebuie să conțină K linii; pe fiecare dintre ele se vor afla două numere întregi P și L, reprezentând localitatea în care se construiește centrul de votare și distanța minimă. Liniile vor fi ordonate crescător în funcție de numărul de ordine al localității.

Exemplu

VOTING.IN
```
5
0 5 0 0 0
5 6
0 0 4 7 0
3 0 0 0 0
0 0 0 0 6
0 0 0 9 0
```

VOTING.OUT
```
1 10
5 6
```

Timp maxim de execuție/test: 2 secunde

România la BOI 2001

La BOI 2001, România a obținut, pentru prima dată la un concurs internațional de programare, patru medalii de aur.

Elevii români care au realizat această performanță deosebită sunt:

Daniel Dumitran (București)
Bors Vencel (Oradea)
Victor Costan (București)
Vlad Dascălu (Bacău)

Delegația României l-a avut ca team leader pe dl. lect. univ. Ovidiu Domșa de la Universitatea "1 Decembrie 1918" din Alba Iulia și ca deputy leader pe dl. prof. Marinel Șerban de la Liceul "Grigore Moisil" din Iași.

Trebuie remarcat faptul că nici un alt concurent nu a mai obținut medalie de aur, elevii români clasându-se pe primele patru locuri. Evident, în clasamentul pe națiuni, România a ocupat primul loc, la mare distanță de Bulgaria, cea de-a doua clasată.

În cea de-a doua zi de concurs, trei dintre reprezentanții României au obținut punctajul maxim (300 de puncte), iar al patrulea a obținut "doar" 280 de puncte.

În final, punctajele concurenților români au fost de 580 (Daniel Dumitran), 560 (Victor Costan și Vencel Bors) și 520 (Vlad Dascălu) de puncte.

Concurentul clasat pe a cincia poziție a obținut 400 de puncte.