Etapa FINALĂ

Ediția a patra a concursului de programare organizat de Gazeta de Informatică s-a încheiat. Primii clasați după cele 20 de runde desfășurate prin e-mail au participat la faza finală care a avut loc pe data de 17 mai 2003 la Liceul "Emil Racoviță" din Cluj-Napoca.

Dintre cei zece concurenți care s-au calificat la faza finală, doar șapte au reușit să se prezinte la concurs.Aceștia sunt:

Ștefan Ciobâcă, Suceava
Silvestru-Cosmin Negrușeri, Bistrița
Ciprian Cană, Vaslui
Adrian Cârcu, Bistrița
Tiberiu Chiorean, Baia Mare
Csaba Pătcaș, Oradea
Aurelian Marinescu, Brașov

Bursa de 50$/lună oferită de editura *Agora Media* a fost câștigată de *Ștefan Ciobâcă*. Pe locul al doilea s-a clasat concurentul de origine rusă *Ivan Kazmenko* care, datorită unor dificultăți nu a putut ajunge în România, dar a participat *on-line* la etapa finală. Deoarece nu s-au acordat burse concurenților străini, bursa de 30$/lună a fost câștigată de *Silvestru-Cosmin Negrușeri*.

Toți participanții la finală au primit cărți oferite de *Agora Media* și *Computer Libris Agora*.

În continuare vă prezentăm enunțurile celor patru probleme propuse spre rezolvare la etapa finală a celei de-a patra ediții a concursului de programare organizat de revista noastră.

P060314: Dragoni

Noul maestru al dragonilor controlează un număr de N puncte de observație ale căror coordonate se cunosc. Nu există trei puncte de observație ale căror coordonate sunt puncte coliniare. Pentru a se proteja de următorul atac al megadragonilor, el decide să trimită câte un dragon la fiecare dintre punctele de observație aflate pe perimetrul teritoriului. Acest perimetru este dat de poligonul de perimetru minim care conține în interior sau în vârfuri toate punctele de observație.

Datorită faptului că maestrul dragonilor știe că, mai devreme sau mai târziu, acest perimetru va fi străpuns, el organizează apărarea în continuare astfel:

- punctele de observație aflate pe perimetru (în vârfurile poligonului) sunt considerate a fi pierdute;
- pentru punctele de observație rămase se determină un nou perimetru și la punctele de observație aflate pe noul perimetru sunt trimiși câte doi dragoni;
- evident, și acest perimetru va fi străpuns, deci este organizat un al treilea nivel de apărare care va fi dat de perimetrul punctelor de observație rămase; la punctele de observație de pe acest perimetru vor fi trimiși câte trei dragoni;
- procedeul va continua (numărul dragonilor crește cu unu pentru fiecare nou perimetru) până în momentul în care nu va mai rămâne nici un punct de observație.

Va trebui să determinați, pentru fiecare punct de observație în parte, numărul dragonilor care vor fi trimiși la punctul de observație respectiv.

Date de intrare

Prima linie a fișierului de intrare **DRAGONS.IN** conține numărul N al punctelor de observație de pe teritoriul maestrului dragonilor. Fiecare dintre următoarele N linii va conține o pereche de numere reale, reprezentând coordonatele unui punct de observație, separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire **DRAGONS.OUT** trebuie să conțină N linii; fiecare linie corespunde unui punct de observație. O astfel de linie va conține un singur număr care reprezintă numărul dragonilor trimiși la punctul de observație respectiv. Ordinea punctelor de observație din fișierul de ieșire trebuie să respecte ordinea acestora din fișierul de intrare.

Restricții

- $3 \le n \le 1000$;
- coordonatele punctelor de observație sunt numere reale cu cel mult trei zecimale exacte;
- eventual, ultimul "perimetru" poate conține unul sau două puncte de observație.

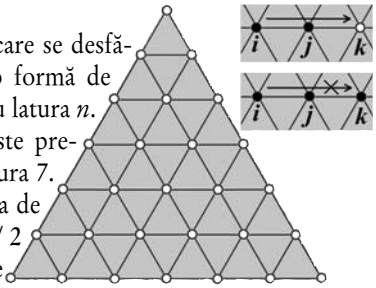
Exemplu

DRAGONS.IN	DRAGONS.OUT
16	1

```
8  2                1
2  4                1
17  4               2
6  5                3
8  7                3
11  9               2
15  10              2
12  18              2
5.1  16             1
4  20               1
15  18              3
7  15               3
6  14               4
11  13              4
12  13              3
13  14
```

Timp maxim de execuție/test: 1 secundă

P060315: Joc



Se consideră un joc care se desfășoară pe o tablă sub formă de triunghi echilateral cu latura n. În figura alăturată este prezentată o tablă cu latura 7. Se observă că pe tabla de joc se află $n \cdot (n + 1) / 2$ puncte de intersecție în care se află scobituri. La început, în unele dintre scobiturile de pe tabla de joc se vor afla bile. Scobiturile vor fi numerotate de la 1 la $n \cdot (n + 1) / 2$ de sus în jos și de la stânga la dreapta. Jocul constă din mai multe mutări și pentru fiecare mutare există o singură regulă: o bilă de pe poziția i poate sări peste o bilă de pe poziția j și se va pune într-o locație liberă k vecină poziției j, dacă poziția i este vecină cu poziția j; pozițiile i, j și k se află pe același segment. După efectuarea unei astfel de mutări bila de pe poziția j va fi eliminată de pe tabla de joc. Procedeul continuă până în momentul în care nu se mai pot efectua mutări.

Cunoscând latura n a tablei de joc și amplasamentul inițial al bilelor, se cere să se determine mutările care trebuie efectuate, astfel încât să se ajungă într-o configurație care conține cât mai puține bile.

Date de intrare

Fișierul de intrare **GAME.IN** conține pe prima linie două numere întregi strict pozitive n și m separate printr-un singur spațiu, care reprezintă latura tablei de joc, respectiv numărul de scobituri de pe tabla de joc în care se află bile. Pe următoarea linie se află m numere întregi strict pozitive separate prin câte un spațiu, care reprezintă numerele de ordine ale scobiturilor care conțin bile.

Date de ieșire

Fișierul de ieșire **GAME.OUT** va conține cel mult m - 1 linii. Pentru fiecare mutare se va se afla pe câte o linie, o pereche formată din două numere naturale strict pozitive i și j, separate prin spațiu, care reprezintă numărul de ordine al scobiturii din care se ia o bilă și numărul de ordine al scobiturii care conține bila peste care se sare. Mutările se vor scrie în fișier în ordinea în care se efectuează.

Restricții

- $4 \le n \le 20$;
- $1 \le m < n \cdot (n + 1) / 2$.

Exemplu

GAME.IN	GAME.OUT
4 5	8 5
4 5 6 7 8	7 4
	6 3
	1 2

Modalitatea de acordare a punctajului

Vom considera că pentru fiecare test se vor putea obține cel mult X puncte. Concurenții care vor obține cea mai mică valoare *NrMin* pentru numărul de bile rămase vor primi X puncte pentru testul respectiv.

Ceilalți concurenți, care au efectuat mutări corecte, dar numărul total al bilelor rămase pe tablă este $Nr > NrMin$, vor obține $X \cdot NrMin / Nr$ puncte pentru testul respectiv. Această valoare va fi aproximată cu două zecimale exacte.

Punctajul final va fi obținut prin adunarea punctajelor de la fiecare test și rotunjirea acestuia la cel mai apropiat număr întreg. Dacă mutările efectuate nu sunt corecte, concurenții nu vor primi nici un punct pentru testul respectiv.

De exemplu, dacă pentru un test se pot obține cel mult 5 puncte, cel mai bun rezultat obținut de un concurent constă într-o succesiune de mutări în urma cărora pe tablă rămân 2 bile, iar un alt concurent obține o succesiune de mutări în urma cărora pe tablă rămân 4 bile, atunci, pentru testul respectiv, punctajul concurentului va fi de $5 \cdot 2 / 4 = 2{,}5$ puncte.

Timp maxim de execuție/test: 1 secundă

P060316: Produse

Se consideră o matrice pătratică A de dimensiune N ale cărei elemente sunt date de formula $a_{i,j} = i^2 \cdot j$. Un produs va fi obținut prin alegerea unei permutări p a mulțimii $\{1, 2, ..., N\}$ și înmulțirea tuturor elementelor de forma $a_{i,x}$, unde $x = p_i$.

Va trebui să determinați numărul de cifre, în baza 10, al sumei tuturor produselor distincte posibile. Două produse se consideră a fi diferite dacă permutările corespunzătoare sunt diferite.

Date de intrare

Fișierele de intrare sunt deschise, iar numele acestora au forma **PRODUCTSXX.IN**, unde **XX** ia valori între 01 și 20 și reprezintă numărul testului.

Fiecare dintre cele 20 de fișiere de intrare conține o singură linie pe care se află valoarea N.

Date de ieșire

Pentru această problemă nu va trebui să trimiteți un program care să o rezolve, ci doar cele 20 de fișiere de ieșire corecte. Acestea vor fi denumite **PRODUCTSXX.OUT**, unde **XX** ia valori între 01 și 20 și reprezintă numărul testului.

Prima linie a fișierului de ieșire va conține textul **PRODUCTS - TEST #XX**, unde **XX** reprezintă numărul testului. Cea de-a doua linie va conține numărul cifrelor sumei tuturor produselor posibile.

Exemplu

Vom considera că acest exemplu reprezintă testul 00.

PRODUCTS00.IN	PRODUCTS00.OUT
2	2

Modalitatea de acordare a punctajului

Pentru fiecare fișier de ieșire corect veți obține un anumit număr de puncte. Punctajul maxim care poate fi obținut (dacă toate cele 20 de fișiere de ieșire sunt corecte) este de 100 de puncte. Pentru primele patru teste (01 - 04) se vor acorda câte 3 puncte, pentru următoarele patru (05 - 08) câte 4 puncte, pentru următoarele patru (09 - 12) câte 5 puncte, pentru următoarele patru (13 - 16) câte 6 puncte, iar pentru ultimele patru (17 - 20) câte 7 puncte.

Timp maxim de execuție/test: 1 secundă

P060317: Mijloc

Se consideră un șir a cu $2 \cdot N$ elemente ale căror valori sunt numere întregi distincte, generate aleator, care nu sunt cunoscute. Se știe că atât primele N elemente ale șirului (cele de pe pozițiile 1, 2, ..., N), cât și ultimele N elemente ale acestuia (cele de pe pozițiile $N + 1$, $N + 2$, ..., $2 \cdot N$) sunt ordonate crescător.

Va trebui să determinați unul dintre indicii elementelor aflate pe pozițiile N sau $N + 1$ în șirul obținut prin sortarea în ordine crescătoare a șirului a. Pentru determinarea acestui indice, puteți efectua un număr limitat de comparări între două elemente ale șirului a și aveți dreptul la două încercări de a "ghici" acest indice.

Pentru aceasta aveți la dispoziție o bibliotecă externă care vă pune la dispoziție rutine pentru:
- inițializare;
- determinarea numărului de elemente ale șirului;
- determinarea numărului maxim de comparări pe care le aveți la dispoziție;
- compararea a două elemente ale șirului a;
- furnizarea rezultatului.

Bibliotecile Pascal și C/C++

Pentru a putea folosi biblioteca va trebui să includeți în programul dumneavoastră instrucțiunea `uses middle;` pentru limbajul *Pascal* și `#include "middle.h"` pentru limbajul *C/C++*.

În continuare sunt prezentate sintaxele funcțiilor și procedurilor incluse în biblioteci:

```
procedure Init;
void Init()
```
- trebuie apelată la începutul programului;
- este folosită de către bibliotecă pentru inițializare și întrerupe execuția programului dacă este apelată a doua oară.

```
function Get2N:Integer;
int Get2N()
```
- returnează numărul de elemente ($2 \cdot N$) ale șirului a.

```
function GetNoComp:Integer;
int GetNoComp()
```
- returnează numărul de comparări (apeluri ale funcției `IsGreater`) pe care le mai aveți la dispoziție (în momentul apelului funcției `GetNoComp`) pentru determinarea rezultatului.

```
function IsGreater(i,j:Integer):Boolean;
int IsGreater(int i,int j)
```
- returnează *true* (valoare nenulă) dacă $a[i] > a[j]$ și *false* (0) în caz contrar;
- întrerupe execuția programului dacă cel puțin una dintre valorile i și j nu este un număr întreg cuprins între 1 și $2 \cdot N$ sau dacă numărul de apeluri ale acestei funcții depășește numărul maxim permis.

```
procedure Result(x:Integer);
void Result(int x)
```
- acceptă ca rezultat faptul că elementul de pe poziția x a șirului a se află pe una dintre pozițiile N sau $N + 1$ în șirul obținut după ordonare;
- întrerupe execuția programului dacă valoarea x reprezintă un rezultat corect sau dacă este apelată a doua oară și valoarea x nu reprezintă un rezultat corect.

Toate funcțiile și procedurile (excepție: `Init`) întrerup execuția programului dacă sunt apelate înaintea inițializării.

Programul vostru nu va citi date din nici un fișier de intrare și nu va scrie date în nici un fișier de ieșire.

Restricții și precizări

- $2 \le N \le 2457$;
- atât indicele elementului de pe poziția N, cât și indicele celui de pe poziția $N + 1$ sunt considerate rezultate corecte;
- doar unul dintre apelurile permise ale rutinei `Result` trebuie să furnizeze un răspuns corect.

Exemplu de utilizare a bibliotecii

Subprogram apelat	*Valoare returnată*
`Init`	-
`Get2N`	6
`GetNoComp`	2
`IsGreater(2,5)`	`false (0)`
`IsGreater(1,4)`	`false (0)`
`Result(5)`	-
`Result(4)`	-

Timp maxim de execuție/test: 1 secundă