Olimpiada NAȚIONALĂ de Informatică

În perioada 27 aprilie-3 mai a avut loc la Brăila ediția din acest an a Olimpiadei Naționale de Informatică. Vă prezentăm în continuare enunțurile celor 24 de probleme propuse spre rezolvare la ONI 2002.

Clasa a IX-a

P050201: Pentagon

(lect. univ. Ovidiu Domșa, Alba Iulia)

În urma bombardamentelor din 11 septembrie 2001, unul dintre pereții clădirii *Pentagonului* a suferit daune majore. Imaginea codificată a peretelui avariat se reprezintă sub forma unei matrice cu *m* linii și *n* coloane ale cărei elemente sunt numere întregi din mulțimea {0, 1}. Valoarea 1 are semnificația "zid intact", în timp ce semnificația valorii 0 este "zid avariat".

Sumele alocate de *Bin Laden* pentru refacerea *Pentagonului* vor fi donate celor care îi vor ajuta pe americani să refacă această clădire prin plasarea, pe verticală, a unor blocuri de diverse înălțimi și lățime 1, în zonele care au fost avariate.

Pentru o structură dată a unui perete din clădirea *Pentagonului*, determinați numărul minim de blocuri necesare pentru refacerea clădirii.

Date de intrare

Fișierul de intrare **PENTAGON.IN** conține, pe prima linie, dimensiunile *m* și *n* ale peretelui clădirii, iar pe următoarele *m* linii câte *n* valori din mulțimea {0, 1}, neseparate prin spații.

Date de ieșire

Fișierul de ieșire **PENTAGON.OUT** va conține, pe câte o linie, secvențe de forma *k nr*, unde *k* indică înălțimea unui bloc, iar *nr* este numărul de blocuri de înălțime *k*. Cele două numere vor fi separate prin câte un spațiu, iar liniile vor fi ordonate crescător, în funcție de valoarea *k*.

Restricții și precizări

- $1 \le m, n \le 200$;
- în fișierul de ieșire nu vor apărea linii de forma *k* 0.

Exemplu

PENTAGON.IN
```
5 10
1110000111
1100001111
1000000011
1111101111
1110000111
```

PENTAGON.OUT
```
1 7
2 1
3 2
5 1
```

Timp de execuție: 1 secundă/test

P050202: Pod

(prof. Marinel Șerban, Iași)

Între două maluri ale unei văi adânci s-a construit un pod suspendat format din *N* bucăți de scândură, legate cu liane. Vom considera că scândurile sunt numerotate de la 1 la *N*, începând de pe malul pe care ne aflăm. În timp, unele bucăți de scândură s-au deteriorat, iar altele chiar au dispărut. Pentru traversarea podului se știe că:

- se pot face pași doar de lungime 1, 2 sau 3;
- scândurile deteriorate sunt nesigure, deci pe ele și de pe ele se pot face doar pași de lungime 1;
- evident, nu se poate păși pe o scândură care lipsește.

Scrieți un program care să determine numărul de modalități de traversare a podului (mai exact, de a ajunge pe celălalt mal), precum și o soluție de traversare, dacă o astfel de soluție există.

Date de intrare

Prima linie a fișierului de intrare **POD.IN** conține numărul *n* al scândurilor. Primul număr de pe a doua linie este *k* și indică numărul scândurilor care lipsesc. Această linie mai conține numerele de ordine ale celor *k* scânduri. Primul număr de pe a treia linie este *h* și indică numărul scândurilor deteriorate. Linia mai conține numerele de ordine ale celor *h* scânduri. Numerele de pe o linie sunt separate prin spații.

Date de ieșire
Prima linie a fișierului de ieșire **POD.OUT** va conține numărul de posibilități de a traversa podul. În cazul în care podul nu poate fi traversat, linia va conține valoarea -1. Dacă podul poate fi traversat, pe a doua linie va fi descrisă o modalitate de traversare. Pe această linie se vor afla numerele de ordine ale scândurilor pe care se pășește, în ordinea în care acestea sunt parcurse.

Restricții și precizări
- $3 \le n \le 300$;
- $0 \le k, h \le n$;
- prima scândură nu poate lipsi;
- numerele de ordine ale scândurilor deteriorate și numerele de ordine ale scândurilor care lipsesc formează două mulțimi disjuncte;
- numărul posibilităților de traversare conține cel mult 80 de cifre.

Exemple

POD.IN	POD.OUT
5 0 0	24 3

POD.IN	POD.OUT
10 2 2 7 1 5	48 3 6 8

POD.IN	POD.OUT
6 2 2 4 1 3	-1

Timp de execuție: 1 secundă/test

P050203: Suma

(Florin Ghețu, București)

Considerăm șirul format din primele N numere naturale: $a_i = i$ pentru $i = 1, ..., N$. Fiecărui element al acestui șir i se asociază un semn (+ sau -).

Pentru o valoare S dată se cere să se determine cel mai mic număr N astfel încât, după asocierea semnelor, elementele obținute să aibă suma S.

Date de intrare
Fișierul **SUMA.IN** va conține un singur număr întreg care reprezintă suma care trebuie obținută după asocierea semnelor.

Date de ieșire
Fișierul de ieșire **SUMA.OUT** va conține, pe prima linie, numărul N al elementelor șirului. Pe următoarele linii se vor afla indicii elementelor care au semn negativ, câte unul pe o linie.

Restricție
- $0 < S \le 100000$.

Exemplu

SUMA.IN	SUMA.OUT
12	7 1 7

Timp de execuție: 1 secundă/test

P050204: Becuri

(prof. Cristina Bărbieru, Timișoara)

Un panou publicitar, de formă dreptunghiulară conține becuri, unul lângă altul, aliniate pe linii și coloane. Fiecare linie și fiecare coloană are un comutator care schimbă starea tuturor becurilor de pe acea linie sau coloană, din starea în care se află în starea opusă. Inițial panoul are toate becurile stinse.

Să se realizeze un program care, acționând asupra unui număr minim de linii și coloane, aduce panoul din starea inițială, la o configurație dată, dacă acest lucru este posibil.

Date de intrare
Pe prima linie a fișierului de intrare **BECURI.IN** se află două numere naturale, m și n, separate printr-un spațiu, reprezentând numărul de linii, respectiv coloane, ale panoului. Pe următoarele m linii se află câte n numere, separate prin câte un spațiu, reprezentând configurația finală a panoului. Un bec aprins este codificat prin 1, în timp ce un bec stins este codificat prin 0.

Date de ieșire
Pe prima linie a fișierului de ieșire **BECURI.OUT** se vor afla indicii coloanelor asupra cărora s-a acționat. Cea de-a doua linie va conține indicii liniilor asupra cărora s-a acționat. Numerele de pe o linie vor fi separate prin câte un spațiu. Dacă nu se acționează asupra nici unei linii sau asupra nici unei coloane, pe linia corespunzătoare va fi scrisă doar valoarea 0. Numerotarea liniilor și a coloanelor începe de la 1. În cazul în care nu există nici o soluție, fișierul de ieșire va conține o singură linie pe care se va afla valoarea -1.

Restricție
- $1 \le m, n \le 100$.

Exemplu

BECURI.IN	BECURI.OUT
5 6 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0	2 5 1 2 3

Timp de execuție: 1 secundă/test

P050205: Discuri

(Florin Ghețu, București)

Se dau N numere reale care reprezintă razele a N discuri. Considerăm că așezăm un disc în sistemul xOy dacă îl plasăm la o coordonată x pozitivă suficient de mare, tangent cu axa Ox și deasupra ei, apoi îl împingem spre Oy până când devine tangent cu Oy sau cu primul disc întâlnit, așezat anterior. În figura rezultată după așezarea tuturor discurilor în ordinea dată, unele dintre ele pot fi considerate ***dispensabile*** deoarece, prin eliminarea lor, nu se modifică lățimea totală a figurii rezultate, adică nici un disc nu se mai poate deplasa spre stânga. Sarcina voastră este de a scrie un program care identifică toate discurile dispensabile dintr-o configurație dată.

Date de intrare

Fișierul de intrare **DISCURI.IN** conține pe prima linie numărul N al discurilor, iar pe următoarele N linii câte un număr real care indică raza unui disc. Razele apar în ordinea amplasării discurilor.

Date de ieșire

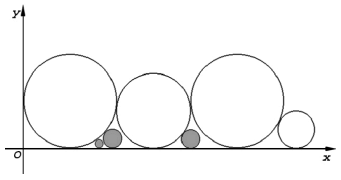
Prima linie a fișierului de ieșire **DISCURI.OUT** va conține numărul K al discurilor dispensabile. Pe următoarele K linii se vor afla numerele de ordine ale discurilor dispensabile.

Restricție

- $1 \le N \le 1000$.

Exemplu

DISCURI.IN	DISCURI.OUT
7	3
4	2
0.1	3
0.5	5
3	
0.5	
4	
1	



cercurile haşurate sunt dispensabile

Timp de execuție: 1 secundă/test

P050206: Cod

(lect. univ. Ovidiu Domșa, Alba Iulia)

Transmiterea și memorarea informațiilor necesită diverse sisteme de codificare în vederea utilizării optime a spațiilor disponibile. Un sistem foarte des întâlnit este acela prin care unei secvențe de caractere i se asociază un număr.

Se consideră cuvintele formate numai cu literele mici ale alfabetului englez. Dintre toate aceste cuvinte le considerăm doar pe cele ale căror caractere sunt în ordine strict lexicografică (caracterul de pe orice poziție este strict mai mic decât un eventual succesor). Sistemul de codificare se obține astfel:

- se ordonează cuvintele în ordinea crescătoare a lungimilor lor;
- cuvintele de aceeași lungime se ordonează lexicografic;
- cuvinte sunt codificate prin numerotarea lor (în șirul obținut după ordonare):

```
1.      - a
2.      - b
...
26.     - z
27.     - ab
...
51.     - az
52.     - bc
...
83681.    - vwxyz
...
```

Dându-se un cuvânt, să se precizeze dacă poate fi codificat conform sistemului descris. În caz afirmativ să se precizeze codul său.

Date de intrare

Fișierul de intrare **COD.IN** conține o singură linie pe care se află un cuvânt.

Date de ieșire

În cazul în care cuvântul poate fi codificat, fișierul de ieșire **COD.OUT** va conține o singură linie pe care se va afla codul acestuia. În caz contrar, fișierul va conține doar valoarea 0.

Restricție

- un cuvânt conține cel mult 10 litere.

Exemple

COD.IN	COD.OUT
bf	55

COD.IN	COD.OUT
aab	0

COD.IN	COD.OUT
vwxyz	83681

Timp de execuție: 2 secunde/test

Clasa a X-a

P050207: Hotel

(prof. Doru Popescu Anastasiu, Slatina)

Departamentul administrativ al unui hotel are n angajați. Patronul hotelului hotărăște să schimbe costumele personalului din acest departament, astfel încât angajații care lucrează la etaje diferite să fie îmbrăcați în haine colorate diferit, iar cei care lucrează la același etaj să fie îmbrăcați în haine colorate la fel. Angajații sunt identificați printr-un cod unic, care constă într-un număr natural format din cel mult patru cifre.

Să se determine numărul modalităților de alegere a costumelor, astfel încât să fie respectate condițiile descrise

anterior. De asemenea, trebuie determinată și una dintre aceste modalități.

Date de intrare
Prima linie a fișierului de intrare **HOTEL.IN** conține numărul *n* al angajaților și numărul *k* al culorilor disponibile. Cele două numere sunt separate printr-un spațiu. Pe următoarele *n* linii se afla câte două numere naturale, separate printr-un spațiu, primul reprezentând codul, iar al doilea etajul asociat unui angajat.

Date de ieșire
Prima linie a fișierului de ieșire **HOTEL.OUT** va conține numărul modalităților de alegere a costumelor. Știind că fiecare culoare este codificată printr-un număr natural nenul mai mic sau egal cu *k* și că numerele asociate culorilor sunt distincte, în fișier se va scrie pe câte o linie (începând cu a doua) codul unei persoane și culoarea costumului, valori separate prin câte un spatiu. Ordinea apariției în fișierul de iesire va fi aceeași cu cea din fișierul de intrare. Dacă nu există nici o soluție, în fișier se va scrie o singură linie care va conține valoarea 0.

Restricții și precizări
- $1 \le n \le 1000$;
- $1 \le k \le 200$;
- hotelul are cel mult 200 de etaje.

Exemple

HOTEL.IN
```
4 5
123 2
35 1
430 2
13 0
```

HOTEL.OUT
```
60
123 1
35 2
430 1
13 3
```

HOTEL.IN
```
5 2
12 1
13 0
14 1
10 2
11 0
```

HOTEL.OUT
```
0
```

Timp de execuție: 1 secundă/test

P050208: Lac

(prof. Dan Grigoriu, București)

O zonă mlăștinoasă are forma dreptunghiulară, având *nl* linii și *nc* coloane. Ea este formată din celule cu latura de o unitate. O parte din acestea reprezintă *uscat*, iar altele reprezintă *apă*, uscatul fiind codificat cu 0, iar apa cu 1. Se dorește să se obțină un drum de pe malul de nord spre cel de sud, trecând doar pe uscat. Celulele cu apă pot fi transformate în uscat, parașutând într-un loc cu apă câte un ponton (o plută) de dimensiunea unei celule. Deoarece parașutarea este periculoasă, se dorește minimizarea numărului de parașutări. Pentru deplasare, dintr-o celulă se poate trece într-o celulă vecină (pe linie, coloană sau diagonală).

Scrieți un program care determină numărul minim de pontoane și coordonatele acestora.

Date de intrare
Fișierul de intrare **LAC.IN** conține, pe prima linie, numerele naturale *nl* și *nc*, separate printr-un singur spațiu. Pe următoarele *nl* linii se află câte *nc* valori binare, separate prin câte un spațiu, reprezentând configurația zonei (0 pentru uscat și 1 pentru apă).

Date de ieșire
Prima linie a fișierului de ieșire **POD.OUT** va conține numărul minim *k* al pontoanelor necesare. Pe următoarele *k* linii se vor afla câte două numere naturale, separate prin câte un spațiu, reprezentând linia, respectiv coloana, în care a fost amplasat unul dintre pontoane.

Restricție
- $1 \le nl, nc \le 100$.

Exemplu

LAC.IN
```
8 9
0 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1
1 0 1 1 1 0 1 1 1
1 1 0 0 1 1 0 1 1
1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 1 1
```

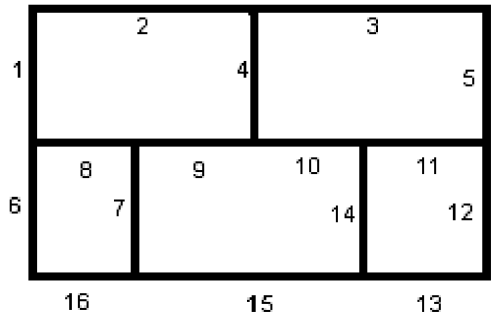
LAC.OUT
```
2
4 5
7 8
```

Timp de execuție: 1 secundă/test

P050209: Logic

(Marius Andrei, București)

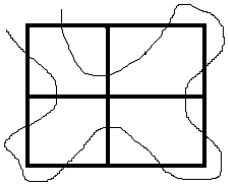
Într-o zi a venit la mine un coleg din clasa a *X*-a și mi-a propus un joc de logică. Mi-a arătat următoarea figură:



Pe această figură am numerotat segmentele, pentru a fi mai clară noțiunea de segment. Eu am la dispoziție un creion care se află pe hârtie în zona exterioară și trebuie să trasez curbe pe foaie, fără să ridic creionul, astfel încât să

trec prin toate segmentele (fără a atinge extremitățile) exact o dată. La sfârșit trebuie să mă aflu tot în exterior. Liniile (curbele) mele se pot intersecta. Am început și am încercat de mai multe ori, dar n-am reușit.

Acum, când am mai crescut, am reușit să demonstrez că nu se poate, dar am observat că pentru alte figuri este posibil. Un exemplu în care problema are soluție este:



Eu mi-am dat seama de ce uneori se poate și alteori nu, dar vreau să văd dacă reușiți și voi. De aceea vă voi da câteva figuri și, pentru fiecare, trebuie să răspundeti cu DA sau NU la întrebarea: *se poate sau nu trasa o curbă având tipul descris mai sus?*

Date de intrare

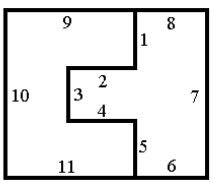
Fișierul de intrare **LOGIC.IN** are următoarea structură:

- pe prima linie se află numărul *T* al figurilor;
- pe următoarele *T* grupuri de linii se află datele corespunzătoare celor *T* figuri, după cum urmează:
  - pe prima linie a grupului se află valoarea *N* care reprezintă numărul de linii și de coloane ale matricei pătratice care descrie figura;
  - pe următoarele *N* linii se află câte *N* numere, separate printr-un spațiu (numere întregi cuprinse între 0 și 255), care reprezintă elementele matricei.

Această matrice codifică figura astfel: definim o zonă ca fiind o parte continuă a matricei care conține același număr și este aria maximă cu această proprietate; altfel spus, două căsuțe alăturate (care diferă la o singură coordonată printr-o unitate) care conțin același număr, se lipesc. Cu alte cuvinte, în interiorul zonei nu vor exista segmente. La aceste zone se mai adaugă și exteriorul matricei ca fiind o nouă zona. În matrice pot fi zone cărora le corespunde același număr, dar care "nu se ating" și se consideră a fi zone diferite. Segmentele sunt segmente de dreaptă care separă două zone și au lungimea maximă. De exemplu, figura definită prin

```
3
1 1 2
1 2 2
1 1 2
```

este următoarea:



Se observă că între zonele 1 și 2 există cinci segmente, între zona 1 și exterior există trei segmente, iar între zona 2 și exterior există tot trei segmente. Trebuie remarcat faptul că segmentul numerotat în figura anterioară cu 10 este compus din trei segmente mici, dar este considerat a fi un singur segment, conform definiției.

Date de ieșire

În fișierul **LOGIC.OUT** se vor scrie *T* linii, corespunzătoare celor *T* figuri. Pe fiecare linie va fi scris unul dintre mesajele DA sau NU, în funcție de răspunsul la întrebare.

Restricții

- $1 \le T \le 10$;
- $1 \le N \le 100$.

Exemplu

LOGIC.IN	**LOGIC.OUT**
2 2 1 2 3 4 4 1 1 2 2 1 1 2 2 3 4 4 1 3 4 4 1	DA NU

Timp de execuție: 1 secundă/test

P050210: Foto

(prof. Roxana Tâmplaru, Craiova)

Gigel, specialist în editare grafică pe calculator, se confruntă cu o problemă. El trebuie să aranjeze patru fotografii, disponibile în format electronic, într-o pagină de prezentare, astfel încât suprafața paginii să fie complet "acoperită" de cele patru fotografii și fără ca acestea să se suprapună în vreun fel. *Gigel* poate modifica dimensiunile inițiale ale fotografiilor, însă fără a deforma imaginile. Pentru aceasta el trebuie să păstreze neschimbat raportul dintre lungimea și înălțimea fotografiilor. Nu contează ordinea așezării fotografiilor, ele putând fi translatate oriunde în cadrul paginii, însă operațiile de rotație nu sunt permise.

Determinați pentru fiecare fotografie dimensiunile finale, cunoscându-se dimensiunile paginii, precum și dimensiunile inițiale ale fotografiilor.

Date de intrare

Prima linie a fișierului de intrare **FOTO.IN** conține două numere naturale *l* și *h*, separate printr-un spațiu, reprezentând lungimea, respectiv înălțimea paginii. Pe următoarele patru linii se află câte două numere naturale, separate printr-un spațiu, reprezentând lungimea, respectiv înălțimea, uneia dintre cele patru fotografii care trebuie aranjate.

Date de ieșire

Fișierul de ieșire **FOTO.OUT** va conține patru linii pe care se vor afla câte două numere naturale care reprezintă dimen-

siunile (lățime și înălțime) finale ale celor patru fotografii. Prima linie va corespunde primei fotografii din fișierul de intrare, a doua linie celei de-a doua fotografii etc.

Restricții și precizări
- dimensiunile paginii și ale fotografiilor sunt numere naturale cuprinse între 1 și 2000;
- dacă există mai multe soluții va fi determinată doar una dintre ele;
- există întotdeauna cel puțin o posibilitate de amplasare a fotografiilor.

Exemplu

FOTO.IN	FOTO.OUT
140 140	20 10
24 12	40 130
4 13	100 140
10 14	20 10
4 2	

Timp de execuție: 1 secundă/test

P050211: Balanță

(Mihai Stroe, București)

Gigel are o "balanță" mai ciudată pe care vrea să o echilibreze. De fapt, aparatul este diferit de orice balanță pe care ați văzut-o până acum. Balanța lui *Gigel* dispune de două brațe de greutate neglijabilă, fiecare având lungimea 15. Din loc în loc, pe aceste brațe se află cârlige, de care *Gigel* poate atârna greutăți distincte din colecția sa de *G* greutăți (numere naturale cuprinse între 1 și 25). *Gigel* poate atârna oricâte greutăți de orice cârlig, dar trebuie să folosească toate greutățile de care dispune.

Folosindu-se de experiența participării la *Olimpiada Națională de Informatică*, *Gigel* a reușit să echilibreze balanța relativ repede, dar acum dorește să știe în câte moduri poate fi ea echilibrată.

Cunoscând amplasamentul cârligelor și greutățile pe care *Gigel* le are la dispoziție, scrieți un program care calculează în câte moduri se poate echilibra balanța.

Date de intrare

Fișierul de intrare **BALANTA.IN** are următoarea structură:
- pe prima linie se află numărul *C* al cârligelor și numărul *G* al greutăților; cele două numere sunt separate printr-un spațiu;
- pe a doua linie se află *C* numere întregi distincte, separate prin spațiu, cu valori cuprinse între -15 și 15 inclusiv, reprezentând amplasamentele cârligelor față de centrul balanței; valoarea absolută a numerelor reprezintă distanța față de centrul balanței, iar semnul indică brațul balanței pe care se află cârligul, `'-'` pentru brațul stâng și `'+'` pentru brațul drept;
- pe a treia linie se află *G* numere naturale distincte, cuprinse între 1 și 25, reprezentând valorile greutăților pe care *Gigel* le va folosi pentru a echilibra balanța.

Date de ieșire

Fișierul de ieșire **BALANTA.OUT** conține o singură linie, pe care se află un număr natural *M*, numărul de variante de plasare a greutăților care duc la echilibrarea balanței.

Restricții și precizări
- $2 \le C, G \le 20$;
- numărul de variante este cuprins între 1 (există întotdeauna cel puțin una) și 100.000.000;
- balanța se echilibrează dacă suma produselor dintre greutăți și coordonatele unde ele sunt plasate este 0 (suma momentelor greutăților față de centrul balanței este 0).

Exemplu

BALANTA.IN	BALANTA.OUT
2 4	2
-2 3	
3 4 5 8	

Timp de execuție: 1 secundă/test

P050212: Aliniere

(prof. Moț Nistor, Brăila)

În armată, o companie este alcătuită din *n* soldați. La inspecția de dimineață soldații stau aliniați în linie dreaptă în fața căpitanului. Acesta nu e mulțumit de ceea ce vede; e drept că soldații sunt așezați în ordinea numerelor de cod din registru, dar nu în ordinea înălțimii. Căpitanul cere câtorva soldați să iasă din rând, astfel ca cei rămași, fără a-și schimba locurile, doar apropiindu-se unul de altul (pentru a nu rămâne spații mari între ei) să formeze un șir în care fiecare soldat vede privind de-a lungul șirului, cel puțin una din extremități (stânga sau dreapta). Un soldat vede o extremitate dacă între el și capătul respectiv nu există un alt soldat cu înălțimea mai mare sau egală cu a lui.

Scrieți un program care determină, cunoscând înălțimea fiecărui soldat, numărul minim de soldați care trebuie să părăsească formația, astfel ca șirul rămas să respecte condiția impusă de căpitan.

Date de intrare

Pe prima linie a fișierului de intrare **ALINIERE.IN** se află numărul *n* al soldaților din șir, iar pe linia următoare se află un șir de *n* numere reale, cu maximum cinci zecimale fiecare, separate prin spații. Al *k*-lea număr de pe această linie reprezintă înălțimea soldatului cu codul *k*.

Date de ieșire

Fișierul **ALINIERE.OUT** va conține pe prima linie numărul soldaților care trebuie să părăsească formația, iar pe linia următoare, codurile acestora în ordine crescătoare, separate două câte două printr-un spațiu.

Restricții și precizări
- înălțimile soldaților sunt numere reale cuprinse între 0,5 și 2,5;

• $2 \le n \le 1000$;
• dacă există mai multe soluții posibile, se va alege doar una dintre ele.

Exemplu

ALINIERE.IN
```
8
1.86 1.86 1.30621 2 1.4 1 1.97 2.2
```

ALINIERE.OUT
```
4
1 3 7 8
```

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P050213: Arbore

(prof. Emanuela Cerchez, Iași)

Se consideră un arbore cu N vârfuri, numerotate de la 1 la N. Scrieți un program care adaugă un număr minim de muchii, astfel încât fiecare vârf al grafului obținut să aparțină exact unui singur ciclu.

Date de intrare

Prima linie a fișierului de intrare **ARBORE.IN** conține numărul N al vârfurilor arborelui. Pe următoarele N - 1 linii se află câte două numere, separate printr-un spațiu, care reprezintă extremitățile uneia dintre muchiile arborelui.

Date de ieșire

Prima linie a fișierului de ieșire **ARBORE.OUT** conține numărul Nr al muchiilor adăugate. Pe următoarele Nr - 1 linii se află câte două numere, separate printr-un spațiu, care reprezintă extremitățile uneia dintre muchiile adăugate. În cazul în care problema nu are soluție, fișierul de ieșire va conține o singură linie pe care se va afla valoarea -1.

Restricție

• $3 \le N \le 1000$.

Exemple

ARBORE.IN
```
4
1 2
2 3
2 4
```

ARBORE.OUT
```
-1
```

ARBORE.IN
```
7
1 2
1 3
3 5
3 4
5 6
5 7
```

ARBORE.OUT
```
2
6 7
2 4
```

Timp de execuție: 1 secundă/test

P050214: Decodificare

(Mugurel-Ionuț Andreica, București)

Serviciul Român de Informații a dat de urma unei organizații teroriste care își are sediul pe teritoriul țării noastre. Folosind cei mai pricepuți și mai bine antrenați spioni și ofițeri, *SRI* a reușit să identifice computerul principal al organizației teroriste. Dacă va reuși să acceseze informațiile din acest computer, *SRI* îi va putea aresta pe toți membrii organizației și va asigura menținerea păcii mondiale. Singura problemă este spargerea codului de acces. Tot ceea ce se știe despre acest cod este că el este reprezentat de către o permutare de lungime N. Specialiștii *SRI* au încercat diverse metode de a descoperi codul, însă tot ceea ce au reușit să obțină este un program care, transmițându-i-se ca parametru o permutare de lungime N, specifică în câte poziții codul de acces coincide cu aceasta.

Scrieți un program care, folosind programul-ajutător (reprezentat sub forma unui modul), determină codul de acces în computerul teroriștilor.

Date de intrare

Programul dumneavoastră nu va citi date din nici un fișier de intrare. El va apela întâi funcția `GetN` a modulului `PROG`, care va returna valoarea N - numărul de elemente ale permutării care trebuie descoperită.

Apoi va apela numai funcția `Check`, căreia îi va fi transmisă ca parametru, de fiecare dată, o permutare de lungime N. Această funcție va returna numărul de poziții în care permutarea coincide cu permutarea care trebuie descoperită. Programul dumneavoastră trebuie ca, după un număr finit de apelări ale funcției `Check`, să descopere permutarea căutată.

Date de ieșire

Programul dumneavoastră va trebui să tipărească în fișierul **DECOD.OUT** o permutare de lungime N. Toate cele N elemente vor fi tipărite pe prima linie a fișierului, fiind separate de câte un spațiu.

Restricție

• $5 \le N \le 256$.

Instrucțiuni pentru programatorii în C/C++

Programatorii în *C/C++* au la dispoziție *header*-ul **PROG.H**. În acest fișier sunt declarate următoarele funcții:

```
int GetN(void)
int Check(int p[256])
```

Funcția `Check` are ca parametru un vector cu elemente întregi. Ea va returna una dintre următoarele valori:
• valoarea -1 dacă primele N elemente (începând cu poziția 0) ale vectorului transmis ca parametru nu constituie o permutare de lungime N;
• în cazul în care permutarea este corectă, se returnează un număr cuprins între 0 și N care reprezintă numărul de

poziții în care permutarea de lungime N transmisă ca parametru coincide cu permutarea care trebuie descoperită.

Instrucțiuni pentru programatorii în Pascal

Modulul extern este implementat sub forma *unit*-ului **PROG**. În acest *unit* sunt declarate următoarele tipuri și funcții care vor fi folosite în programul vostru:

```
type perm=array[1..256] of Integer;
function GetN:Integer;
function Check(var permut:perm):Integer;
```

Funcția `Check` are ca parametru un vector de tipul `perm`. Ea va returna una dintre următoarele valori:

- valoarea -1 dacă primele N elemente (începând cu poziția 0) ale vectorului transmis ca parametru nu constituie o permutare de lungime N;
- în cazul în care permutarea este corectă, se returnează un număr cuprins între 0 și N care reprezintă numărul de poziții în care permutarea de lungime N transmisă ca parametru coincide cu permutarea care trebuie descoperită.

Exemplu

Să presupunem că permutarea căutată este `2 1 3 4 5`. La început va fi apelată funcția `GetN`; aceasta va returna valoarea `5`. Vor urma apeluri succesive ale funcției `Check`, până în momentul în care aceasta returnează valoarea `5`. O posibilă succesiune este:

- apel pentru permutarea `1 2 3 4 5` - se returnează `3`;
- apel pentru permutarea `3 2 1 4 5` - se returnează `2`;
- apel pentru permutarea `2 1 3 4 5` - se returnează `5`.

Permutarea corectă este scrisă în fișierul **DECOD.OUT**.

Timp de execuție: 1 secundă/test

P050215: SETI

(Mihai Pătrașcu, Craiova)

Se pare că, în sfârșit, căutătorii vieții extraterestre au descoperit ceva! În cursul proiectului *SETI@home* a fost izolată o secvență care ar putea reprezenta un semnal de la alte forme de viață inteligentă. Ca urmare, proiectul *SETI@ONI* își propune să verifice dacă acel semnal provine într-adevăr de la extratereștri sau doar de la niște puști care beau *Fanta*.

Pentru comoditate, porțiunea de semnal care trebuie analizată vi se pune la dispoziție sub forma unei succesiuni de litere ale alfabetului latin. De asemenea, aveți la dispoziție și un dicționar de cuvinte extraterestre, codificate în același mod. Scopul dumneavoastră este să numărați de câte ori apare fiecare dintre aceste cuvinte în posibilul mesaj extraterestru. Pornind de la aceste date, lingviștii pot să înceapă lucrul la traducerea mesajului.

Date de intrare

Pe prima linie a fișierului de intrare **SETI.IN** se află numărul N al liniilor mesajului. Urmează N linii, fiecare conținând exact 64 de litere ale alfabetului latin, urmate de marcajul de sfârșit de linie. Prin alipirea acestor linii se obține mesajul de analizat, format din $64 \cdot N$ litere.

Pe prima linie a celui de-al doilea fișier de intrare **DIC.IN** este scris numărul M al cuvintelor din dicționar. Urmează M linii, fiecare conținând un cuvânt din dicționar, reprezentat ca o secvență de cel puțin una și cel mult 16 litere. Cuvintele nu sunt neapărat distincte.

Date de ieșire

Fișierul de ieșire **SETI.OUT** va conține exact M linii. Pe linia cu numărul i va fi scris numărul de apariții ale celui de-al i-lea cuvânt din dicționar. Orice apariție a unui cuvânt trebuie numarată, chiar dacă se suprapune peste alte apariții. Se va face distincție între literele mari și literele mici.

Restricții și precizări

- $0 \le N < 2048$;
- $0 \le M \le 32000$;
- un cuvânt poate avea cel mult 65535 de apariții.

Exemplu

SETI.IN

```
2
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaBaba
babaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaBaB
```

DIC.IN

```
3
b
bab
b
```

SETI.OUT

```
3
2
3
```

Timp de execuție: 1 secundă/test

P050216: Suma divizorilor

(Mihai Pătrașcu, Craiova)

Se consideră două numere naturale A și B. Fie S suma tuturor divizorilor naturali ai lui A^B. Să se determine restul împărțirii lui S la 9901.

Date de intrare

Pe prima linie a fișierului de intrare **SUMDIV.IN** se află numerele A și B, separate prin cel puțin un spațiu.

Date de ieșire

Fișierul de ieșire **SUMDIV.OUT** va conține o singură linie pe care se va afla restul împărțirii numărului S la 9901.

Restricție

- $0 \le A, B \le 50.000.000$.

Exemplu

SUMDIV.IN	SUMDIV.OUT
2 3	15

Timp de execuție: 0,5 secunde/test

P050217: Sistem

(Mugurel-Ionuț Andreica, București)

Județul în care are loc *Olimpiada Națională de Informatică* din acest an este special. În județ există N orașe, numerotate de la 1 la N. Fiecare dintre cele N orașe ale județului este legat de ***exact*** alte două orașe, prin străzi bidirecționale. Și mai ciudat este faptul că, în cadrul acestui sistem stradal, nu este întotdeauna posibil să ajungi din orice oraș în oricare alt oraș mergând pe străzi. Oricum, locuitorii județului sunt mândri de acest sistem al lor și sunt de părere că nu mai există altul la fel. Dumneavoastră vreți să le demonstrați contrariul și pentru aceasta vreți să calculați câte sisteme stradale distincte cu proprietatea de mai sus există. Două sisteme sunt considerate distincte dacă există cel puțin o stradă între o pereche de orașe i și j în cadrul primului sistem, care nu exista în cadrul celui de-al doilea.

Date de intrare

Din fișierul **SISTEM.IN** veți citi valoarea N, care reprezintă numărul de orașe ale județului.

Date de ieșire

În fișierul **SISTEM.OUT** se va scrie numărul de sisteme stradale distincte, cu proprietatea că orice oraș este legat prin străzi directe de exact alte două orașe.

Restricție

- $3 \leq N \leq 100$.

Exemple

SISTEM.IN	SISTEM.OUT
4	3

SISTEM.IN	SISTEM.OUT
6	70

Timp de execuție: 1 secundă/test

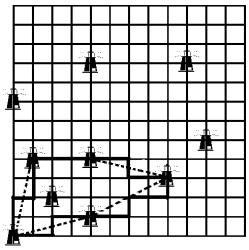
P050218: Comitat

(prof. Rodica Pintea, București)

Toate semințiile conviețuitoare pe *Terra* au hotărât ca *hobbiții*, păstrătorii *Inelului Puterii*, să fie izolați într-o zonă a pământului numită *Comitat*. Hotarele *Comitatului* trebuie să fie reprezentate de un poligon convex cu câte un turn de pază în fiecare vârf.

Se cunosc pozițiile tuturor turnurilor din regiune (două numere naturale raportate la un sistem de axe rectangulare). Un paznic pe cal alb veghează hotarele *Comitatului* parcugând, pe rând, toate distanțele dintre două turnuri succesive mergând pe drum minim, numai pe cărări paralele cu axele.

Se cunoaște lungimea maximă a drumului pe care-l poate parcurge paznicul la un tur complet al hotarelor *Comitatului* și se cere să se determine un poligon cu un număr maxim de turnuri pe contur, poligon care poate constitui hotarul comitatului. În plus, hotarul trebuie să conțină turnul din *Mordor* (de coordonate 0 și 0) într-un vârf, iar în fiecare dintre celelalte vârfuri se află, obligatoriu, unul dintre turnurile existente.



De exemplu, pentru amplasamentul turnurilor ilustrat alăturat și pentru limita de 25 Km a unui tur efectuat de paznic, hotarul *Comitatului* poate fi format, în această ordine, din turnurile de coordonate (0, 0), (4, 1), (8, 3), (4, 4), (1, 4), (0, 0). Se observă că poligonul determinat de aceste turnuri este un poligon convex cu cinci turnuri pe contur.

Poligonul cu vârfurile (0, 0), (4, 1), (4, 12), (0, 7), (0, 0) are tot cinci turnuri pe contur, dar un tur complet al acestui poligon depășește 25 Km.

Date de intrare

Pe prima linie a fișierului de intrare **COMITAT.IN** se află numărul n al turnurilor din ținut (turnul din *Mordor* nu este numărat). Pe fiecare dintre următoarele n linii se află câte două numere naturale, despărțite printr-un spațiu, care reprezintă coordonatele unuia dintre turnuri. Ultima linie a fișierului conține lungimea maximă L a unui tur complet al poligonului.

Date de ieșire

Prima linie a fișierului de ieșire **COMITAT.OUT** va conține numărul v al turnurilor de pe conturul poligonului (incluzând turnul din *Mordor*). Pe a doua linie se vor afla v - 1 numere întregi, reprezentând numerele de ordine ale turnurilor de pe contur, pornind de la turnul din *Mordor* (care nu este afișat) și respectând succesiunea în sens trigonometric sau antitrigonometric a turnurilor de pe contur.

Restricții și precizări

- $0 < n \leq 50$;
- $0 < L < 1000$;
- coordonatele turnurilor sunt numere întregi cuprinse între 0 și 200;
- pot exista turnuri în interiorul poligonului, dar acestea nu sunt luate în considerare pentru numărarea turnurilor corespunzătoare unui poligon;
- există posibilitatea ca soluția să fie dată de un poligon degenerat format dintr-un singur vârf (*Mordor*), din două vârfuri (*Mordor* și un alt turn) sau din mai multe vârfuri coliniare;
- pe conturul poligonului determinat pot exista turnuri coliniare;
- dacă există mai multe soluții care respectă condițiile din enunț, se va furniza doar una dintre acestea;
- în fișierul de intrare nu există două turnuri ale căror poziții să coincidă și singurul turn din poziția (0, 0) este cel din *Mordor*.

Exemplu

COMITAT.IN	COMITAT.OUT
9	5
0 7	4 7 5 2
1 4	
2 2	
4 1	
4 4	
4 9	
8 3	
9 9	
10 5	
25	

Timp de execuție: 1 secundă/test

Barajul de selecție

P050219: EQS

(Mihai Pătrașcu, Craiova)

Se consideră ecuații de forma

$$a_1 \cdot x^3 + a_2 \cdot y^3 + a_3 \cdot z^3 + a_4 \cdot u^3 + a_5 \cdot v^3 = 0.$$

Se consideră soluție un pentuplu (x, y, z, u, v) care verifică ecuația. Să se determine numărul de soluții ale ecuației, pentru care valorile necunoscutelor sunt numere întregi ***nenule*** din intervalul [-50, 50].

Date de intrare

Fișierul de intrare **EQS.IN** conține o singură linie pe care se află cei cinci coeficienți ai ecuației, separați prin câte un spațiu.

Date de ieșire

Fișierul de ieșire **EQS.OUT** va conține o singură linie pe care se va afla numărul soluțiilor ecuației.

Precizare

- valorile coeficienților, precum și cele ale necunoscutelor, sunt numere întregi cuprinse în intervalul [-50, 50].

Exemplu

EQS.IN	EQS.OUT
37 29 41 43 47	654

Timp de execuție: 2 secunde/test

P050220: Labirint

(Mihai Stroe, București)

Romeo și *Julieta* sunt prinși într-un labirint, descris printr-o matrice cu M linii și N coloane, având elemente din mulțimea {0, 1}. Un element cu valoarea 1 reprezintă un zid, în timp ce un element cu valoarea 0 reprezintă un spațiu liber. *Romeo* se află inițial în colțul din stânga-sus al labirintului (1, 1), iar *Julieta* în colțul din dreapta-jos (M, N).

În pozițiile inițiale ale celor doi nu pot exista ziduri, iar ei se pot deplasa doar în spațiile libere, fără a părăsi matricea.

Pentru a evada din labirint, *Romeo* trebuie să ajungă în poziția (i_1, j_1), iar *Julieta* în poziția (i_2, j_2).

Dumneavoastră dispuneți de un șir de mutări pe care cei doi le pot efectua. Șirul este format din K mutări, codificate prin literele N, E, S și V, reprezentând deplasări de un pătrățel spre *Nord*, *Est*, *Sud*, respectiv *Vest*. *Romeo* și *Julieta* trebuie să efectueze ***toate*** mutările din acest șir, ***în ordinea dată***. Dumneavoastră sunteți cel care decide dacă o anumită mutare va fi efectuată de *Romeo* sau de *Julieta*.

Scrieți un program care va decide care dintre mutări vor fi efectuate de *Romeo* și care de *Julieta* pentru a-i ajuta pe cei doi să ajungă la destinații.

Date de intrare

Fișierul de intrare **LABIRINT.IN** are următoarea structură:

- pe prima linie se află numerele M și N;
- pe următoarele M linii se află descrierea labirintului; fiecare linie conține N valori, separate prin câte un spațiu;
- pe următoarea linie se află patru numere întregi, separate prin câte un spațiu, care reprezintă valorile i_1, j_1, i_2 și j_2;
- pe următoarea linie se află numărul K al mutărilor care trebuie efectuate;
- pe ultima linie se află un șir de K litere care descriu direcțiile în care trebuie efectuate mutările.

Date de ieșire

În fișierul de ieșire **LABIRINT.OUT** se va scrie o singură linie, care conține K caractere. Al i-lea caracter este R, dacă *Romeo* va efectua a i-a mutare sau J dacă mutarea va fi efectuată de *Julieta*.

Restricții și precizări

- $3 \le M \le 20$;
- $3 \le N \le 60$;
- $1 \le K \le 200$;
- există întotdeuna cel puțin o soluție; în cazul în care există mai multe, se va scrie doar una dintre ele;
- prin deplasare la *Nord* se înțelege deplasarea pe linia precedentă, o deplasare la *Sud* înseamnă deplasarea pe linia următoare, deplasarea la *Vest* înseamnă deplasarea pe coloana precedentă, iar deplasarea la *Est* înseamnă deplasarea pe coloana următoare.

Exemplu

LABIRINT.IN	LABIRINT.OUT
5 5	RJRRJR
0 0 1 0 0	
0 0 0 1 1	
0 0 0 0 0	
1 0 0 0 0	
0 1 0 0 0	
2 2 4 4	
6	
SNEEVV	

Timp de execuție: 1 secundă/test

P050221: ATP

(șef lucrări Stelian Ciurea, Sibiu)

Se știe că jucătorii profesioniști de tenis sunt clasați, în funcție de rezultatele obținute, într-un clasament al *Asociației Tenismanilor Profesioniști* (*ATP*). Studiindu-se rezultatele ultimilor ani, s-a constatat că într-o partidă în care se întâlnesc doi jucători, dacă diferența dintre pozițiile celor doi în clasament este mai mare decât o valoare *k*, atunci câștigă în mod sigur cel mai bine clasat; în schimb, dacă diferența de poziții în clasament este de cel mult *k* poziții atunci, teoretic, poate să câștige oricare dintre ei.

La un concurs care urmează să se desfășoare în curând după sistemul eliminatoriu, și-au anunțat participarea primii *n* jucători din clasament (unde *n* este o putere a lui 2).

Să se determine care este cel mai slab jucător care ar putea, teoretic, să câștige concursul, precum și schema de desfășurare a partidelor din turneu și câștigătorul fiecărei partide, pentru ca jucătorul respectiv să câștige turneul.

Date de intrare

Fișierului de intrare **ATP.IN** conține o singură linie pe care se află valorile *n* și *k*, separate prin spații.

Date de ieșire

Fișierul de ieșire **ATP.OUT** va avea următoarea structură:

- pe prima linie, poziția în clasament a celui mai slab clasat jucător care, teoretic, ar putea câștiga concursul;
- pe a doua linie, întâlnirile din primul tur al turneului; o întâlnire va fi descrisă prin două numere, corespunzătoare pozițiilor din clasament ale jucătorilor care o susțin, iar primul jucător din pereche este câștigătorul; așadar, pe prima linie vor fi *n* / 2 perechi de numere;
- pe a treia linie, întâlnirile din al doilea tur, descrise la fel ca cele din prima linie; pe această linie vor fi *n* / 4 perechi;
- pe ultima linie partida din finală.

Restricție

- $2 \le N \le 4096$.

Exemplu

ATP.IN

```
16 3
```

ATP.OUT

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

Timp de execuție: 1 secundă/test

P050222: Gard

(Mugurel-Ionuț Andreica, București)

O echipă de *K* muncitori a fost angajată să vopsească un gard format din *N* scânduri numerotate de la 1 la *N*, de la stânga spre dreapta. Fiecare muncitor *i* se așează în fața scândurii S_i și poate vopsi numai un interval compact (numerele de ordine ale scândurilor din interval sunt consecutive), având maxim L_i scânduri, interval care trebuie să conțină scândura S_i. Pentru fiecare scândură vopsită, acesta este plătit cu suma P_i. Din motive de eficiență, oricare doi muncitori din echipă trebuie să vopsească intervale de scânduri disjuncte.

Fiind conducătorul echipei de muncitori, dumneavoastră doriți să determinați, pentru fiecare membru al echipei, intervalul de scânduri pe care acesta va trebui să îl vopsească, astfel încât câștigul total să fie maxim. Câștigul total este egal cu suma câștigurilor realizate de fiecare membru al echipei. Câștigul realizat de fiecare muncitor este egal cu produsul dintre numărul de scânduri vopsite de acesta și suma P_i corespunzătoare muncitorului.

Scrieți un program care determină câștigul maxim obținut de cei *K* muncitori.

Date de intrare

Pe prima linie a fișierului de intrare **GARD.IN** se află numărul *N* al scândurilor și numărul *K* al muncitorilor, separate printr-un spațiu.

Fiecare dintre următoarele *K* linii va conține trei numere *l*, *p* și *s* cu semnificația: *muncitorul corepunzător liniei poate vopsi cel mult* l *scânduri, venitul primit pentru fiecare scândură vopsită este* p *și, inițial, el se află în fața scândurii* s.

Date de ieșire

În fișierul de ieșire **GARD.OUT** se va scrie câștigul maxim care poate fi obținut de echipa de muncitori.

Restricție

- $1 \le N \le 16000$;
- $1 \le K \le 100$;
- $1 \le P_i \le 10000$;
- $1 \le L_i, S_i \le N$;
- inițial, doi muncitori nu se pot afla în fața aceleiași scânduri;
- nu trebuie vopsite neapărat toate cele *N* scânduri ale gardului;
- este posibil ca unul sau mai mulți muncitori să nu vopsească nici o scândură, caz în care scândura în fața căreia s-au așezat inițial poate fi vopsită, eventual, de către alt muncitor.

Exemplu

GARD.IN

```
8 4
3 2 2
3 2 3
3 3 5
1 1 7
```

GARD.OUT

```
17
```

Timp de execuție: 1 secundă/test

Despre concurs...

Ediția din acest an a Olimpiadei Naționale de Informatică a fost un prilej de întâlnire a celor mai talentați elevi din județele țării. Ca de obicei, numărul concurenților a fost foarte mare, la Brăila sosind peste 200 de participanți.

Organizatorii locali s-au achitat excelent de sarcina asumată, din acest punct de vedere neexistând probleme care să afecteze buna desfășurare a competiției.

Din nefericire pentru concurenți, anul acesta numărul premiilor a fost redus. S-au acordat doar trei premii la fiecare clasă, iar următorii clasați au primit mențiuni. În anii anteriori, aproximativ 30% dintre participanți primeau premii, iar alți 20% primeau mențiuni. La ediția din acest an procentul celor care au primit premii sau mențiuni a fost de doar 25%. Cu toate acestea, un aspect pozitiv demn de remarcat este faptul că, acum, un premiu obținut la olimpiada națională are o valoare mult mai mare.

Problemele propuse spre rezolvare au fost destul de dificile, nici unul dintre concurenți nereușind să obțină un punctaj apropiat de cel maxim. Din nefericire, trebuie să remarcăm o continuare a tendinței de scădere a nivelului de pregătire a participanților. Chiar dacă faptul că nu există punctaje apropiate de cel maxim nu reprezintă neapărat un indiciu în acest sens, este îngrijorător faptul că numărul celor care obțin punctaje apropiate de 0 este din ce în ce mai mare.

Punctaje foarte mici apar și la barajul de selecție al lotului național de informatică. Cel de-al cincisprezecelea clasat după cele două probe de baraj a obținut doar 175 de puncte din 600 posibile, deci nu a rezolvat corect decât cel mult una dintre cele șase probleme.

Primii clasați au obținut performanțe mai bune, cel mai mare punctaj obținut la această probă fiind de 473 de puncte.

Situația pare mai îngrijorătoare dacă privim clasamentul pe clase; primul clasat de la clasa a XII-a a obținut doar 305 puncte, adică puțin peste jumătate din punctajul maxim.

În ciuda dificultăților inerente care au apărut, Comisia Centrală și-a făcut din nou datoria, neexistând probleme majore nici din acest punct de vedere.

La această olimpiadă GInfo a fost reprezentată de dl. prof. univ. dr. Horia Georgescu, directorul științific al revistei noastre (președintele Comisiei Centrale) și de d-na lect. univ. Clara Ionescu, redactorul șef al publicației.

P050223: Sistem

(asist. univ. Iuliu Vasilescu, București)

Se consideră N monede identice ca formă și culoare, numerotate de la 0 la N - 1. Printre ele există exact o monedă falsă. Toate monedele adevărate au aceeași greutate, cea falsă fiind mai grea sau mai ușoară decât celelalte. Pentru găsirea monedei false se folosește o balanță cu două talere identice, pe care încap oricâte monede. O cântărire folosind această balanță constă în a pune pe cele două talere ale balanței câte un număr egal de monede. În urma cântăririi se determină dacă greutățile de pe talere sunt egale sau talerul care conține o greutate mai mare.

Scrieți un program care să comande cântăririle care trebuie efectuate, astfel încât să se determine moneda falsă și dacă este mai grea sau mai ușoară decât cele adevărate, în număr minim de cântăriri. Pentru a afla numărul de monede și rezultatul cântăririlor programul va trebui să folosească un modul extern.

Instrucțiuni pentru programatorii în C/C++

Programatorii în *C/C++* au la dispoziție *header*-ul **MON.H**. În acest fișier sunt declarate următoarele tipuri și funcții:

```
typedef char taler[1000];
int init();
int cantarire(taler stanga, taler dreapta);
void rezultat(int moneda, int tip);
```

Instrucțiuni pentru programatorii în Pascal

Programatorii în *Pascal* au la dispoziție *unit*-ul **MON**. În acest *unit* sunt declarate următoarele tipuri, funcții și proceduri:

```
type taler = array[0..999] of Byte;
function init:Integer;
function cantarire(var stanga, dreapta:
                                taler):Integer;
procedure rezultat(moneda, tip:Integer);
```

Instrucțiuni pentru utilizarea modulului

Tipul `taler` este folosit pentru a reprezenta monedele de pe un taler al balanței. Un vector de tipul `taler` va avea elemente din mulțimea {0, 1}. Un vector de acest tip conține pe o poziție valoarea 1, dacă și numai dacă moneda cu numărul corespunzător se află în balanță pe talerul reprezentat prin vector și 0 în caz contrar.

Prima funcție care trebuie apelată este `init`, funcție care returnează numărul de monede.

În continuare va fi apelată, de câte ori este nevoie, funcția `cantarire`, având ca parametri configurațiile celor două talere. Funcția returnează 0 dacă talerele sunt echilibrate, -1 dacă talerul din stânga este mai ușor și 1, dacă talerul din stânga este mai greu.

Procedura/funcția `rezultat` va fi apelată la sfârșit pentru a anunța moneda falsă. Primul parametru este numărul monedei false, iar al doilea trebuie să fie -1 dacă moneda falsă este mai ușoară decât celelalte, respectiv 1 dacă moneda falsă este mai grea decât celelalte.

Nu vor fi realizate operații de intrare/ieșire cu fișiere.

Restricție
• $3 \le N \le 1000$.

Timp de execuție: 1 secundă/test

P050224: Banana

(prof. Roxana Tâmplaru și Mihai Pătrașcu, Craiova)

Se consideră o pădure tropicală, reprezentată sub forma unui caroiaj dreptunghiular. Celula din colțul stânga sus al caroiajului are coordonatele (1, 1), iar coordonatele celorlalte celule sunt determinate de linia și coloana pe care se află. În anumite celule ale caroiajului sunt plasați bananieri; o celulă conține cel mult un bananier. Mai mulți bananieri care se învecinează pe orizontală sau verticală formează o zonă de bananieri. Într-o astfel de zonă, *Cekili* se deplasează ușor, cu agilitatea-i cunoscută, de la un bananier la altul.

Maimuța *Cekili* este lacomă și nu îi ajung bananele dintr-o singură zonă. *Tarzan* vrea să-și ajute prietena. Pentru aceasta, el ar putea conecta exact *K* zone de bananieri înnodând mai multe liane și astfel *Cekili* s-ar putea deplasa de la o zonă la alta utilizând lianele.

Evident, *Tarzan* trebuie să aleagă zonele astfel încât numărul total de bananieri din cele *K* zone să fie cât mai mare.

Determinați numărul maxim de bananieri care se poate obține prin conectarea a exact *K* zone.

Date de intrare
Pe prima linie a fișierului de intrare **BANANA.IN** se află numărul *Nr* al bananierilor și numărul *K* al zonelor care pot fi conectate, despărțite printr-un spațiu. Pe fiecare dintre următoarele *Nr* linii se află câte două numere naturale, despărțite printr-un spațiu, care reprezintă coordonatele unuia dintre bananieri.

Date de ieșire
Fișierul de ieșire **BANANA.OUT** va conține o singură linie pe care se va afla numărul maxim de bananieri care se poate obține prin conectarea zonelor.

Restricții și precizări
• $1 \le Nr \le 16000$;
• coordonatele turnurilor sunt numere întregi cuprinse între 1 și 10000;
• numărul de zone este cel puțin egal cu *K*;
• două poziții se învecinează pe orizontală dacă sunt pe aceeași linie și pe coloane consecutive, respectiv pe verticală dacă sunt pe aceeași coloană și pe linii consecutive.

Exemplu

BANANA.IN	BANANA.OUT
10 3 7 10 1 1 101 1 2 2 102 1 7 11 200 202 2 1 3 2 103 1	9

Timp de execuție: 1 secundă/test

Premianții

Clasa a IX-a
Leonard Crestez, Brăila - premiul I
Iolanda Popa, Iași - premiul II
Ionuț Costică, Iași - premiul III

Clasa a X-a
George-Dan Ghinea, București - premiul I
Andrei Homescu, Gorj - premiul II
Cristian Cutocheraș, Dâmbovița - premiul III

Clasa a XI-a
Radu Berinde, București - premiul I
Victor Costan, București - premiul II
Cosmin Răianu, Constanța - premiul III

Clasa a XII-a
Daniel Dumitran, București - premiul I
Vlad Dascălu, Bacău - premiul II
Ștefăniță Fechete, Suceava - premiul III

Membrii lotului național
Andrei Benea, Vrancea
Radu Berinde, București
Ștefan Ciobâcă, Suceava
Victor Costan, București
Vlad Dascălu, Bacău
Alexandru Dumitrache, Argeș
Daniel Dumitran, București
George-Dan Ghinea, București
Claudiu Gruia, București
Bogdan Hârjoc, Alba
Andrei Markovits, Satu Mare
Paul Marinescu, Buzău
Cosmin Răianu, Constanța
Cristian-Francisc Toth, Timiș
Victor Vernescu, Constanța