CONCURSUL de programare AGORA

Cea de-a patra rundă a Concursului de Programare Agora s-a încheiat. În acest număr vă vom prezenta soluțiile oficiale ale celor trei probleme, precum și rezultatele obținute de concurenți la această etapă. De asemenea, veți putea vedea care este clasamentul intermediar, după patru etape.

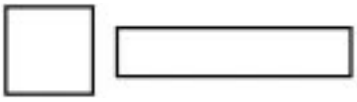
Cele trei probleme ale rundei a patra au fost relativ simple dar, surprinzător, doar șase concurenți au reușit să obțină toate cele 300 de puncte puse în joc.

Capcana

După cum se poate vedea în clasamentul etapei, există o mulțime de concurenți care au acumulat 297 puncte, adică au pierdut doar punctele corespunzătoare unui test. Este vorba despre un test de la problema *Asemănare* care i-a pus în dificultate pe mulți participanți.

Această problemă cerea verificarea asemănării a două poligoane. Generalizând unul din cazurile de asemănare pentru triunghi, un număr foarte mare de concurenți au considerat că *două poligoane sunt asemenea dacă au unghiurile respectiv congruente*.

Totuși, în realitate, această afirmație nu este adevărată, după cum se poate vedea în figura de mai jos.



Este evident că toate unghiurile au 90°, deci unghiurile sunt respectiv congruente. Totuși, este la fel de evident că cele două poligoane nu sunt asemenea.

Aceasta a fost cea mai "frumoasă" capcană în care au căzut concurenții la această rundă. Cei care nu au fost atenți la acest detaliu ar trebui să fie foarte bucuroși deoarece a fost ales un singur test de acest tip.

Liste de discuții

Existența diferitelor liste de dialog unde se discută despre *Concursul de Programare Agora* nu poate decât să ne bucure. Există însă concurenți care nu sunt de acord cu faptul că pe aceste liste apar și idei de rezolvare pentru probleme. Ei sunt nemulțumiți că se chinuie să rezolve anumite probleme, în timp ce alții așteaptă ca altcineva să le trimită o idee de rezolvare.

Noi considerăm că un concurent care nu rezolvă singur nici o problemă, nu are șanse prea mari să se claseze între primii zece. De aceea, vă asigurăm că un astfel de concurent nu va participa la etapa finală.

Totuși, principalul scop al *Concursului de Programare Agora* este acela de a vă ajuta să vă pregătiți pentru concursurile de programare "reale" la care doriți să participați. Din acest motiv, credem că listele care au ca subiect problemele de la concursul nostru, constituie un element pozitiv.

Cu toate acestea, după părearea noastră, cei care preiau o rezolvare a unui alt concurent și o trimit ca fiind rezolvare proprie nu procedează corect. În plus, există riscul ca acel concurent să fie descalificat, deoarece soluția este mult prea asemănătoare cu cea a unui alt concurent. Situația este diferită în cazul în care este preluată doar ideea de rezolvare și concurentul implementează un algoritm propriu.

În concluzie, considerăm că aceste liste de discuții au un rol foarte important, dar nu ar trebui să se exagereze cu prezentarea soluțiilor.

De exemplu, nu ne-am bucurat atunci când pe una dintre listele de discuții a apărut exact formula care trebuia folosită pentru rezolvare. Din nefericire pentru alții, pe o altă listă s-a ales o formulă incorectă.

Descalificări

Anul trecut, obișnuiam să publicăm numele celor descalificați în paginile revistei. Am decis ca de data aceasta numele acestora să nu mai fie publice, concurenții descalificați fiind înștiințați prin e-mail asupra deciziei redacției. Până acum au fost descalificați 27 de concurenți.

Runda 4 - Top 50

Csaba Andras, Oradea	**300**
Ciprian Baicu, Drobeta Turnu Severin	**300**
Vencel Bors, Oradea	**300**
Cosmin-Silvestru Negruseri, Bistrița	**300**
Csaba Patcas, Oradea	**300**
Radu Ștefan, Brașov	**300**

Mugurel Ionuț Andreica, București	**297**
Aurelian Ghiță, Buzău	**297**
Mihai Hărănguș, Cluj-Napoca	**297**
Maximilian Machedon, București	**297**
Liviu Păunescu, Constanța	**297**

12. George Drumea, București	294
13. Bogdan Stan, Câmpina	293
14. Alexandru Mosoi, Bacău	292
15. Adrian Cârcu, Bistrița	290
16. Lucian-Daniel Stanciu, Brăila	286
17. Mihai-Silviu Chiru, Brăila	281
18. Artur Kuczapski, Oradea	280
19. Ciprian Cană, Vaslui	275
20. Lucian-Raul Silistru, Vaslui	272
21. Cristian Alexandrescu, Botoșani	262
22. Ștefan Bucur, București	261
23. Petru-Simon Moț, Brăila	250
24. Dan Ghinea, București	247
25. Cătălin Kesa, Bușteni	238
26. Alexandra-Elena Constantin, Ploiești	237
27. Adrian Cearnău, București	235
27. Andrei Csibi, Cluj-Napoca	235
27. Mihail Pit-Rada, Drobeta Turnu Severin	235
27. Mihai-Vlad Pantiș, Cluj-Napoca	235
31. Horea Coroiu, Cluj-Napoca	233
31. Claudiu Gruia, București	233
33. Cristian Băicoianu, Ploiești	232
34. Mihail Popa, București	228
35. Octavian-Daniel Dumitran, București	226
35. Valentin Pop, Carei	226
35. Mihai Stroe, București	226
38. Victor Asavei, Drobeta Turnu Severin	223
38. Gabriel Pârvan, Râmnicu Sărat	223
40. Bogdan Dumitru, Ploiești	221
41. Paul Marinescu, Buzău	218
42. Bogdan Nicolae, Sibiu	215
43. Andrei-Liviu Dumbrava, Iași	210
44. Daniel Crișan, Râmnicu Vâlcea	206
45. Andrei Giurgiu, București	203
46. Elvis Asaftei, Vaslui	200
46. Horia Ciurdar, Timișoara	200
46. Vlad Dascălu, Bacău	200
46. Costin Speciac, Buzău	200
50. Cristian Toth, Lugoj	199
50. Vlad Vâlceanu, Arad	199

Sfere

Această problemă a fost rezolvată corect de către 50 dintre cei care au participat la a patra rundă a ***Concursului de Programare Agora***. O metodă de rezolvarea a problemelor de acest tip a fost prezentată în articolul *Geometrie 3D* publicat în numărul 11/2 (februarie 2001) al ***GInfo***.

Să presupunem că am ales k sfere; vom încerca să determinăm cu cât crește numărul de zone în care este împărțit spațiul după alegerea celei de-a (k+1)-a sfere.

Numărul maxim de zone se obține dacă oricare două sfere se intersectează, adică nu există nici o pereche de sfere tangente, nu există nici o sferă care să se afle complet în interiorul unei alte sfere, nu există trei sfere care să aibă un cerc comun și oricare patru sfere nu au nici un punct comun.

Sfera considerată se va intersecta cu celelalte k sfere după k cercuri. Vom încerca să determinăm acum numărul maxim de zone în care aceste k cercuri pot împărți suprafața sferei.

Deoarece dimensiunile sferelor nu au nici o importanță, putem considera că noua sferă are o rază mult mai mare decât cea a celorlalte k sfere. La limită, vom considera că raza acestei sfere este infinită. Datorită faptului că, în acest caz, sfera degenerează într-un plan (un plan poate fi considerat ca fiind o sferă de rază infinită) problema în spațiu se reduce la o problemă în plan.

Enunțul acestei noi probleme este următorul: *determinați numărul maxim de zone în care poate fi împărțit planul prin intermediul a n cercuri*.

Cercurile împart planul într-un număr maxim de zone dacă oricare două cercuri se intersectează, adică nu există nici o pereche de cercuri tangente, nu există nici un cerc care să se afle complet în interiorul unui alt cerc și nu există nici un grup de trei cercuri care să aibă un punct comun.

Considerăm că am ales k cercuri și încercăm să determinăm numărul de zone suplimentare care apar dacă alegem cel de-al (k+1)-lea cerc. Deoarece cercul se intersectează cu fiecare dintre celelalte k cercuri în câte două puncte, rezultă că circumferința sa va fi împărțită în $2 \cdot k$ regiuni. Ca urmare, numărul de zone suplimentare care apar după considerarea unui nou cerc este egal cu dublul numărului cercurilor considerate anterior.

Așadar, relația de recurență, cu ajutorul căreia să se poată determina numărul de zone în care poate fi împărțit un plan, este $c_{k+1} = c_k + 2 \cdot k$. Deoarece un cerc împarte planul în două zone (interiorul cercului și exteriorul său), avem $c_1 = 2$.

Astfel obținem:

$$\begin{aligned} c_n &= 2+2+4+6+8+\ldots+2\cdot(n-1) \\ &= 2+2\cdot(1+2+3+\ldots+n-1) \\ &= 2+2\cdot\frac{n\cdot(n-1)}{2} \\ &= n^2-n+2. \end{aligned}$$

Așadar, numărul maxim de zone în care k cercuri pot împărți suprafața unei sfere este de $k^2 - k + 2$. Ca urmare, prin considerarea celei de-a (k+1)-a sfere numărul de zone este mărit cu $k^2 - k + 2$.

Așadar, relația de recurență, cu ajutorul căreia se poate determina numărul de zone în care poate fi împărțit spațiul, este $s_{k+1} = s_k + k^2 - k + 2$.

Deoarece o sferă împarte spațiul în două zone (interiorul și exteriorul ei), $c_1 = 2$.

Efectuând calculele obținem:

$$\begin{aligned} s_n &= 2+(1^2-1+2)+(2^2-2+2)+(3^2-3+3)+\ldots+[(n-1)^2-(n-1)+2] \\ &= 2+[1^2+2^2+3^2+\ldots+(n-1)^2]-[1+2+3+\ldots+(n-1)]+2\cdot(n-1) \\ &= 2+\frac{n\cdot(n-1)\cdot(2\cdot n-1)}{6}-\frac{n\cdot(n-1)}{2}+2\cdot(n-1) \\ &= \frac{12+2\cdot n^3-3\cdot n^2+n-3\cdot n^2+3\cdot n+12\cdot n-12}{6} \\ &= \frac{2\cdot n^3-6\cdot n^2+16\cdot n}{6} \\ &= \frac{n\cdot(n^2-3\cdot n+8)}{3}. \end{aligned}$$

Așadar, numărul maxim de zone în care poate fi împărțit spațiul cu ajutorul a n sfere este $\frac{n\cdot(n^2-3\cdot n+8)}{3}$.

Versiunea oficială a rezolvării acestei probleme este următoarea:

Listing SPHERE.CPP

```
#include <stdio.h>

void main(void){
  long n;
  FILE *f=fopen("SPHERE.IN","rt");
  fscanf(f,"%ld",&n);
  fclose(f);
  f=fopen("SPHERE.OUT","wt");
  fprintf(f,"%ld",n*(n*n-3*n+8)/3);
  fclose(f);
}
```

Au rezolvat corect:

Cristian Alexandrescu, Botoșani
Csaba Andras, Oradea
Mugurel Ionuț Andreica, București
Liviu-Cosmin Andreicuț, București
Elvis Asaftei, Vaslui
Victor Asavei, Drobeta Turnu Severin
Cristian Băicoianu, Ploiești
Ciprian Baicu, Drobeta Turnu Severin
Livia Bana, Drobeta Turnu Severin
Vencel Bors, Oradea
Ștefan Bucur, București
Ciprian Cană, Vaslui
Adrian Cârcu, Bistrița
Adrian Cearnău, București
Mihai-Silviu Chiru, Brăila
Alexandra-Elena Constantin, Ploiești
Horea Coroiu, Cluj-Napoca
Andrei Csibi, Cluj-Napoca
George Drumea, București
Andrei-Liviu Dumbrava, Iași
Octavian-Daniel Dumitran, București
Bogdan Dumitru, Ploiești
Aurel-Mihai Fulger, Constanța
Dan Ghinea, București
Aurelian Ghiță, Buzău
Vincent Groenhuis, Olanda
Claudiu Gruia, București
Mihai Hărănguș, Cluj-Napoca
Cătălin Kesa, Bușteni
Artur Kuczapski, Oradea
Maximilian Machedon, București
Paul Marinescu, Buzău
Mihail Piț-Rada, Drobeta Turnu Severin
Alexandru Mosoi, Bacău
Petru-Simon Moț, Brăila
Cosmin-Silvestru Negruseri, Bistrița
Bogdan Nicolae, Sibiu
Mihai-Vlad Pantiș, Cluj-Napoca
Gabriel Pârvan, Râmnicu Sărat
Csaba Patcas, Oradea
Liviu Păunescu, Constanța
Valentin Pop, Carei
Mihail Popa, București
Lucian-Raul Silistru, Vaslui
Johannes Slotta, Germania
Costin Speciac, Buzău
Bogdan Stan, Câmpina
Lucian-Daniel Stanciu, Brăila
Mihai Stroe, București
Radu Ștefan, Brașov

Asemănare

Această problemă a fost rezolvată corect doar de către 8 dintre cei care au participat la a patra rundă a *Concursului de Programare Agora*. Problema a fost propusă spre rezolvare membrilor lotului național de informatică în tabăra de pregătire și selecție de la *Cluj*, din anul 1998. O rezolvare a acestei probleme a fost publicată și în numărul 8/6 (toamna 1998) al *GInfo*.

Potrivit definiției asemănării poligoanelor, două poligoane se vor numi ***asemenea*** dacă și numai dacă au unghiurile respectiv congruente și laturile respectiv proporționale.

Știm că în cazul particular al triunghiurilor, pentru a demonstra asemănarea este suficient să arătăm că una dintre următoarele afirmații este adevărată:

- triunghiurile au unghiurile respectiv congruente;
- triunghiurile au laturile respectiv proporționale;
- triunghiurile au două laturi respectiv proporționale și unghiurile formate de aceste laturi în cele două triunghiuri sunt congruente.

Un poligon cu n laturi poate fi triangularizat în n - 2 triunghiuri. Dacă pentru două poligoane cele n - 2 triunghiuri sunt respectiv asemenea, atunci putem afirma că poligoanele sunt asemenea.

De aici putem deduce că două poligoane sunt asemenea dacă au laturile și diagonalele care pleacă dintr-un anumit vârf respectiv proporționale.

Vom rezolva problema folosind această proprietate. Vom considera vârfurile primului poligon în ordinea dată și vom încerca să verificăm dacă putem determina o ordine a vârfurilor celui de-al doilea poligon pentru care să obținem asemănarea.

Vom începe pe rând cu fiecare dintre vârfurile 1, ..., n. Pentru fiecare vârf k, vom considera pentru început ordinea k, k + 1, ..., n, 1, 2, ... k - 1 și apoi ordinea k, k - 1, ..., 1, n, n - 1, ..., k + 1.

Pentru a reduce puțin numărul calculelor vom lucra cu pătratele distanțelor în loc de distanțe propriu-zise. Acest lucru este posibil deoarece următoarea relație este întotdeauna adevărată:

$$\frac{a}{b}=\frac{c}{d}\Leftrightarrow\frac{a^2}{b^2}=\frac{c^2}{d^2}.$$

De asemenea, în loc de a testa o relație de tipul $a / b = c / d$, vom testa relația echivalentă $a \cdot d = b \cdot c$.

Vă prezentăm în continuare versiunea oficială a soluției pentru această problemă:

Au rezolvat corect:

Csaba Andras, Oradea
Ciprian Baicu, Dr. Turnu Severin
Vencel Bors, Oradea
Daniel Crișan, Râmnicu Vâlcea
Silvestru Negruseri, Bistrița
Csaba Patcas, Oradea
Dan Popovici, Arad
Radu Ștefan, Brașov

Listing POLY.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define epsilon 0.001

double x1[100],y1[100], // primul poligon
       x2[100],y2[100]; // al doilea poligon
int n, // numarul de varfuri
    c[100]; // ordinea considerata

void ReadData(void){
  FILE *f=fopen("POLY.IN","rt");
  fscanf(f,"%d",&n);
  for (int i=0;i<n;i++)
    fscanf(f,"%lf%lf",&x1[i],&y1[i]);
  for (i=0;i<n;i++)
    fscanf(f,"%lf%lf",&x2[i],&y2[i]);
}

inline double length1(int i,int j){
// lungimea unei laturi sau a unei diagonale
// in primul poligon
  return (x1[i]-x1[j])*(x1[i]-x1[j])+
         (y1[i]-y1[j])*(y1[i]-y1[j]);
}
```

```
inline double length2(int i,int j){
// lungimea unei laturi sau a unei diagonale
// in primul poligon
  return (x2[i]-x2[j])*(x2[i]-x2[j])+
         (y2[i]-y2[j])*(y2[i]-y2[j]);
}

int Ok(void){
// verificarea pastrarii raportului intre
// lungimile laturilor si diagonalelor
// corespunzatoare
  double denominator=length1(0,n-1),
         nominator=length2(c[0],c[n-1]),
         denominator2,
         nominator2;
  // s-a determinat raportul dintre
  // lungimile laturilor care unesc primul si
  // ultimul varf in cele doua poligoane

  // se verifica pastrarea raportului pentru
  // celelalte laturi
  for (int i=0;i<n-1;i++){
    denominator2=length1(i,i+1);
    nominator2=length2(c[i],c[i+1]);
    // se verifica pastrarea raportului
    if (fabs(denominator*nominator2-
        denominator2*nominator)>epsilon)
      // daca raportul nu se pastreaza
      // atunci nu am gasit ordinea pentru
      // care cele doua poligoane sunt
      // asemenea
      return 0;
  }

  // se verifica pastrarea raportului pentru
  // diagonalele care pleaca din primul varf
  // al celor doua poligoane
  for (i=0;i<n-1;i++){
    denominator2=length1(0,i);
    nominator2=length2(c[0],c[i]);
    // se verifica pastrarea raportului
    if (fabs(denominator*nominator2-
        denominator2*nominator)>epsilon)
      // daca raportul nu se pastreaza
      // atunci nu am gasit ordinea pentru
      // care cele doua poligoane sunt
      // asemenea
      return 0;
  }
  // raportul s-a pastrat, deci aceasta este
  // ordinea pentru care cele doua
  // poligoane sunt asemenea si constituie
  // solutia problemei
  return 1;
}
```

```
void WriteSolution(void){
  FILE *f=fopen("POLY.OUT","wt");
  for (int i=0;i<n;i++)
    // se aduna 1 deoarece am considerat ca
    // indicii incep de la valoarea
    // implicita (zero)
    fprintf(f,"%d\n",c[i]+1);
  fclose(f);
  // se intrerupe executia programului
  // pentru ca nu mai este nevoie sa gasim
  // alte solutii
  exit(1);
}

void SameOrder(int k){
// cele doua poligoane sunt parcurse in
// acelasi sens
  for (int i=k;i<n;i++)
    c[i-k]=i;
  for (i=0;i<k;i++)
    c[n-k+i]=i;
  if (Ok())
    WriteSolution();
}

void ReverseOrder(int k){
  for (int i=k;i>=0;i--)
    c[k-i]=i;
  for (i=n-1;i>k;i--)
    c[n-i+k]=i;
  if (Ok())
    WriteSolution();
}

void Check(void){
  for (int i=0;i<n;i++){
    // acelasi sens de parcurgere
    SameOrder(i);
    // sensuri diferite de parcurgere
    ReverseOrder(i);
  }
}

void NoSolution(void){
  FILE *f=fopen("POLY.OUT","wt");
  fprintf(f,"0\n");
  fclose(f);
}

void main(void){
  ReadData();
  Check();
  // nu am reusit sa gasim nici o solutie
  NoSolution();
}
```

Dreptunghi

Această problemă a fost rezolvată corect de 76 dintre cei care au participat la a doua rundă a ***Concursului de Programare Agora***. Problema era foarte simplă, fiind o aplicație clasică a metodei programării dinamice.

Vom construi o matrice A de dimensiuni $m \times n$, elementul a_{ij} indicând numărul minim de pătrate în care poate fi împărțit un dreptunghi care are lungimile laturilor i și j. Este evident faptul că elementele a_{ii} au valoarea 1.

Pentru a determina valorile elementelor a_{ij} va trebui să determinăm poziția în care ar trebui să efectuăm tăietura pentru a minimiza numărul pătratelor în care va fi împărțit dreptunghiul care are laturi de lungimi i și j.

Pentru aceasta vom determina numărul de pătrate corespunzător fiecărei tăieturi verticale și fiecărei tăieturi orizontale. Formula pe care o vom folosi este:

$$a_{ij} = \min\left(\min_{k=1,i}\left(a_{kj} + a_{i-k,j}\right)\min_{k=1,j}\left(a_{ik} + a_{i,j-k}\right)\right).$$

Se observă că numărul minim de pătrate în care poate fi împărțit un dreptunghi cu laturile i și j este egal cu numărul minim de pătrate în care poate fi împărțit orice dreptunghi cu laturile $k \cdot i$ și $k \cdot j$, pentru orice număr natural $k > 0$.

De aceea, pentru fiecare element al matricei A, am putea verifica dacă nu am determinat deja valoarea corespunzătoare. Valoarea a fost deja determinată numai dacă cmmdc(i, j) > 1. În acest caz avem $a_{ij} = a_{i/\text{cmmdc}(i,j),\, j/\text{cmmdc}(i,j)}$.

Prin aceasta timpul de execuție se reduce semnficativ. Totuși, o variantă de rezolvare care nu folosește această observație se încadrează fără probleme în timpul de execuție admis.

Varianta oficială a rezolvării acestei probleme este prezentată în continuare:

Listing RECT.CPP

```
#include <stdio.h>

int m,n,a[101][101];

void ReadData(void){
// citirea datelor de intrare
  FILE *f=fopen("RECT.IN","rt");
  fscanf(f,"%d%d",&m,&n);
  fclose(f);
}

int gcd(int m,int n){
// functie recursiva pentru determinarea
// celui mai mare divizor comun a doua
// numere naturale
  return (n)?gcd(n,m%n):m;
}

void FindSolution(void){
  int div;
  a[1][1]=1;
  // pentru lungimile laturilor 1 si 1 avem
  // un singur patrat
  for (int i=1;i<=m;i++)
    for (int j=1;j<=n;j++)
      if ((div=gcd(i,j))>1)
        a[i][j]=a[i/div][j/div];
      else{
        // numarul maxim de patrate in care
        // poate fi impartit un dreptunghi
        // cu laturile de lungimi i si j
        // este i * j
        a[i][j]=i*j;
        // taieturi orizontale
        for (int k=1;k<i;k++)
          if (a[k][j]+a[i-k][j]<a[i][j])
            a[i][j]=a[k][j]+a[i-k][j];
        // taieturi verticale
        for (k=1;k<j;k++)
          if (a[i][k]+a[i][j-k]<a[i][j])
           a[i][j]=a[i][k]+a[i][j-k];
      }
}

void WriteSolution(void){
  FILE *f=fopen("RECT.OUT","wt");
  fprintf(f,"%d",a[m][n]);
  fclose(f);
}

void main(void){
  ReadData();
  FindSolution();
  WriteSolution();
}
```

Au rezolvat corect:

Cristian Alexandrescu, Botoșani
Csaba Andras, Oradea
Mugurel Ionuț Andreica, București
Elvis Asaftei, Vaslui
Victor Asavei, Drobeta Turnu Severin
Cristian Băicoianu, Ploiești
Ciprian Baicu, Drobeta Turnu Severin
Adrian Balasko, Zalău
Andrei Benea, Focșani
Valentin Bisa, Lugoj
Denis Bogdănaș, Iași
Daniel Bolohan, Iași
Vencel Bors, Oradea
Cătălin Bujdei, Brașov
Nadina Busuioc, Constanța
Ciprian Cană, Vaslui
Adrian Cârcu, Bistrița
Adrian Cearnău, București
Bogdan Chirilă, Botoșani
Mihai-Silviu Chiru, Brăila
Horia Ciurdar, Timișoara
Alexandra-Elena Constantin, Ploiești
Horea Coroiu, Cluj-Napoca
Daniel Crișan, Râmnicu Vâlcea
Andrei Csibi, Cluj-Napoca
Vlad Dascălu, Bacău
Andrei David, Râmnicu Vâlcea
Radu Dondera, București
George Drumea, București
Andrei-Liviu Dumbrava, Iași
Octavian-Daniel Dumitran, București
Bogdan Dumitru, Ploiești
Dan Ghinea, București
Aurelian Ghiță, Buzău
Andrei Giurgiu, București
Claudiu Gruia, București
Gabor Gyebnar, Ungaria
Mihai Hărănguș, Cluj-Napoca
Andrei Huțu, Bacău
Claudiu Ionesi, Bacău
Cătălin Kesa, Bușteni
Artur Kuczapski, Oradea
Liviu Lalescu, Craiova
Codruț-Lucian Lazăr, Câmpeni
Bogdan Lucaciu, Drobeta Turnu Severin
Maximilian Machedon, București
Paul Marinescu, Buzău
Andrei Markovits, Satu Mare
Mihail Pițˇ-Rada, Drobeta Turnu Severin
Petko Minkov, Bulgaria
Emilian Miron, Bacău
Alexandru Mosoi, Bacău
Petru-Simon Moț, Brăila
Răzvan Mușaloiu-Elefteri, București
Raluca Mușaloiu-Elefteri, București
Cosmin-Silvestru Negruseri, Bistrița
Bogdan Nicolae, Sibiu
Sorin Otescu, Brașov
Mihai-Vlad Pantiș, Cluj-Napoca
Gabriel Pârvan, Râmnicu Sărat
Flaviu Pașca, Bistrița
Csaba Patcas, Oradea
Mihai Pătrașcu, Craiova
Liviu Păunescu, Constanța
Valentin Pop, Carei
Mihail Popa, București
Lucian-Raul Silistru, Vaslui
Bogdan Stan, Câmpina
Lucian-Daniel Stanciu, Brăila
Mihai Stoicescu, Boldești-Scăeni
Mihai Stroe, București
Radu Ștefan, Brașov
Cristian Toth, Lugoj
Vlad Vâlceanu, Arad
Victor Vernescu, Constanța
Violeta Voinescu, Târgu Jiu

Runda 6

Vă prezentăm în continuare cele trei probleme de la cea de-a patra rundă a Concursului de Programare Agora. Cei care doresc să participe numai la această ultimă rundă, pot trimite mesajul de înscriere până la data de 5 Aprilie 2001.

CPAR6P1: Inversiuni

Nume fișier sursă: **INVERS.PAS**, **INVERS.C** sau **INVERS.CPP**
Nume fișier de intrare: **INVERS.IN**
Nume fișier de ieșire: **INVERS.OUT**
Descriere: Se dă un vector a care conține n ($2 \le n \le 10.000$) elemente și se cere determinarea numărului de perechi (i, j) cu proprietatea $i < j$ și $a_i > a_j$.
Date de intrare: Fișierul de intrare va conține pe prima linie numărul n al elementelor vectorului. Pe următoarea linie se află cele n elemente ale vectorului, separate prin spații.
Date de ieșire: Fișierul de ieșire va conține numărul de perechi (i, j) care au proprietatea cerută.
Exemplu:

INVERS.IN	**INVERS.OUT**
4	5
4 2 3 1	

Timp de execuție: 1 secundă/test.

CPAR6P2: Rețea

Nume fișier sursă: **NET.PAS**, **NET.C** sau **NET.CPP**
Nume fișier de intrare: **NET.IN**
Nume fișier de ieșire: **NET.OUT**
Descriere: Se consideră o rețea pătratică cu n ($1 \le n \le 100$) linii și coloane. Din fiecare punct al rețelei se poate ajunge în unul din cei patru vecini de pe orizontală sau de pe verticală. Se consideră m ($1 \le m \le n^2$) puncte ale rețelei. Să se determine dacă există trasee distincte care pornesc din cele m puncte și ajung la marginea rețelei (pe prima/ultima linie/coloană).
Date de intrare: Prima linie a fișierului de intrare conține numărul n al liniilor și coloanelor rețelei și numărul m al punctelor de plecare. Următoarele m linii conțin coordonatele punctelor de plecare.
Date de ieșire: Dacă există m drumuri distincte până la marginea rețelei atunci fișierul de ieșire va descrie fiecare dintre aceste drumuri. Prima linie corespunzătoare descrierii unui traseu conține numărul k al vârfurilor din care este format traseul, iar a doua linie conține coordonatele punctelor de pe traseu cuprinse între paranteze. Dacă nu există trasee distincte, fișierul de ieșire va conține doar cifra 0.
Exemplu:

NET.IN	**NET.OUT**
6 10	1
1 3	(1 3)
2 2	2
2 3	(2 2) (2 1)
2 4	4
4 2	(2 3) (3 3) (3 2) (3 1)
4 3	4
4 4	(2 4) (3 4) (3 5) (3 6)
6 2	2
6 3	(4 2) (4 1)
6 4	4
	(4 3) (5 3) (5 2) (5 1)
	3
	(4 4) (4 5) (4 6)
	1
	(6 2)
	1
	(6 3)
	1
	(6 4)

Timp de execuție: 1 secundă/test.

CPAR6P3: Numere prime

Nume fișier sursă: **PRIME.PAS**, **PRIME.C** sau **PRIME.CPP**
Nume fișier de intrare: **PRIME.IN**
Nume fișier de ieșire: **PRIME.OUT**
Descriere: Se dă un număr natural n ($2 \le n \le 100$) și se cere determinarea a m ($1 \le m \le n$) numere prime care să aibă exact n cifre.
Date de intrare: Prima linie a fișierului de intrare conține numărul n al cifrelor numerelor prime care trebuie determinate și numărul m al numerelor prime care trebuie determinate.
Date de ieșire: Fișierul de ieșire va conține m numere prime care conțin exact n cifre, câte unul pe o linie.
Exemplu:

PRIME.IN	**PRIME.OUT**
2 2	23
	29

Timp de execuție: 1 secundă/test.

Ginfo Top 100

1. Maximilian Machedon, București	1171	6. Csaba Andras, Oradea	1099
2. George Drumea, București	1165	7. Mihai Stroe, București	1089
3. Ciprian Cană, Vaslui	1161	8. Bogdan Stan, Câmpina	1065
4. Radu Ștefan, Brașov	1124	9. Cristian Băicoianu, Ploiești	1062
5. Adrian Cârcu, Bistrița	1114	10. Gabriel Pârvan, Râmnicu Sărat	1034

11. Mugurel Ionuț Andreica, București	1029	41. Andrei Csibi, Cluj-Napoca	773	71. Lucian-Daniel Stanciu, Brăila	418
12. Liviu Lalescu, Craiova	1028	42. Horia Ciurdar, Timișoara	763	72. Bogdan Lucaciu, Dr. Turnu Severin	412
13. Daniel Crișan, Râmnicu Vâlcea	1015	42. Andrei David Râmnicu, Vâlcea	763	73. Paul Rusu, Cluj-Napoca	396
14. Dan Ghinea, București	1013	44. Alexandru Mosoi, Bacău	755	74. Andrei-Liviu Dumbrava, Iași	387
15. Ciprian Baicu, Drobeta Turnu Severin	1006	45. Cristian Toth, Lugoj	754	75. Ștefan Bucur, București	378
16. Denis Bogdănaș, Iași	1004	46. Valentin Bisa, Lugoj	719	76. Andrei Benea, Focșani	376
17. Cristian Alexandrescu, Botoșani	976	47. Mihai Hărănguș, Cluj-Napoca	718	77. Petko Minkov, Bulgaria	366
18. Claudiu Gruia, București	965	48. Răzvan Mușăloiu-Elefteri, București	706	78. Livia Bana, Drobeta Turnu Severin	359
19. Mihai Pătrașcu, Craiova	946	49. Cătălin Kesa, Bușteni	696	79. Aurel-Mihai Fulger, Constanța	357
20. Victor Asavei, Drobeta Turnu Severin	944	50. Paul Marinescu, Buzău	693	80. Cristian Bojinovici, București	353
21. Vlad Vâlceanu, Arad	918	51. Horea Coroiu, Cluj-Napoca	692	81. Bogdan-Vlad Țira, Cluj-Napoca	349
22. Liviu Păunescu, Constanța	916	52. Marius Andrei, București	658	82. Emilian Miron, Bacău	337
23. Octavian-Daniel Dumitran, București	914	52. Codruț-Lucian Lazăr, Câmpeni	658	83. Mihai Gojol, Cluj-Napoca	335
24. Bogdan Nicolae, Sibiu	907	54. Victor Vernescu, Constanța	655	84. Raluca Sauciuc, București	330
25. Vencel Bors, Oradea	900	55. Liviu-Cosmin Andreicuț, București	640	85. Lucian Burja, Blaj	320
25. Mihail Popa, București	900	56. Cosmin Răianu, Constanța	596	86. Costin Speciac, Buzău	316
25. Cosmin-Silvestru Negruseri, Bistrița	900	57. Radu Dondera, București	593	87. Nadina Busuioc, Constanța	315
28. M. A. Safari Ghahsareh, Iran	892	58. Artur Kuczapski, Oradea	591	88. Silviu-Gabriel Udrea, Târgu Jiu	305
29. Vincent Groenhuis, Olanda	842	59. Valentin Pop, Carei	577	89. Adrian Balasko, Zalău	304
30. Mihai-Vlad Pantiș, Cluj-Napoca	841	60. Andrei Giurgiu, București	570	90. Johannes Slotta, Germania	298
31. Radu Vătavu, Suceava	838	61. Adrian Cearnău, București	566	91. Bogdan-Nicolae Șanta, Cluj-Napoca	290
32. Vlad Dascălu, Bacău	832	62. Lucian-Raul Silistru, Vaslui	554	92. Daniel Comșa, Crăciunelu de Jos	286
33. Dan Popovici, Arad	827	63. Andrei Markovits, Satu Mare	546	93. Alina Fești, Cluj-Napoca	284
34. Csaba Patcas, Oradea	825	64. Mihail Piț-Rada, Dr. Turnu Severin	545	94. Mihai-Silviu Chiru, Brăila	281
35. Ioana-Maria Ileană, Alba Iulia	824	65. Bogdan-Milovan Piloga, Timișoara	489	95. Daniel Bolohan, Iași	266
36. Bogdan Dumitru, Ploiești	811	66. Arcadie Crăcan, Iași	456	95. Preslav Nakov, Bulgaria	266
37. Alexandra-Elena Constantin, Ploiești	808	67. Raluca Mușăloiu-Elefteri, București	443	97. Narayanan Arvind, India	264
38. Flaviu Pașca, Bistrița	802	68. Petru-Simon Moț, Brăila	437	97. Radu Cocieru, Sibiu	264
39. Aurelian Ghiță, Buzău	785	69. V. Muthuramakrishnan, India	421	99. Constantin Asofiei, Târgu Neamț	251
40. Sorin Otescu, Brașov	784	70. Laurent Demonet, Franța	420	100. Cristian Ioniță, București	243