MINISTERUL EDUCAŢIEI AL REPUBLICII MOLODOVA

Academia de Studii Economice din Moldova

Olimpiada Republicană la Informatică. Ediţia 2014

Chişinău 2014

Olimpiada Republicană la Informatică. Ediţia 2014. Chişinău, ASEM, 2014. – 75 p.

Lucrarea conţine problemele propuse la Olimpiada Republicană la Informatică a elevilor, ediţia 2014. Pentru fiecare problemă sunt descrise algoritmul şi soluţia respectivă în limbajul de programare PASCAL.

Materialele Olimpiadei pot fi de un real folos la studierea Informaticii atât elevilor, cât şi profesorilor de Informatică.

La elaborarea ediţiei au contribuit:

Bolun Ion	prof.univ., dr.hab., ASEM
Prisăcaru Angela	consultant, Ministerul Educaţiei
Bercu Igor	lector sup. univ., ASEM
Croitor Mihai	lector univ., USM
Dolghier Constantin	informatician
Globa Angela	lector sup. univ., UST (Chişinău)
Hîncu Boris	conf. univ., dr., USM
Negară Corina	lector sup. univ., dr. US Bălţi

CUPRINS

Dragi elevi şi stimaţi profesori,
care vă dăruiţi acestui domeniu de vârf
al ştiinţei şi tehnologiei moderne pe nume
Informatica

Orice competiţie este o experienţă pentru participanţi. Cu cât mai înalt este ratingul competiţiei, cu atât mai bogată şi fructuoasă este experienţa obţinută. Olimpiada Republicană la Informatică întruneşte cei mai buni elevi în domeniu din republică şi are ca scop atât încurajarea şi susţinerea interesului elevilor către informatică, cât şi evaluarea gradului de pregătire a lor în domeniu, elucidarea lacunelor în instruire şi definirea unor căi de îmbunătăţire continuă a acesteia.

Ea permite, de asemenea, stabilirea de noi cunoştinţe între elevi şi profesori, dar şi reevaluarea valorilor. Rezultatele acesteia sunt folosite la selectarea candidaţilor pentru echipa de elevi ce vor participa la competiţii internaţionale la Informatică.

Printre olimpiadele republicane ale elevilor, cea la Informatică are, totuşi, un rol aparte, determinat de rolul deosebit al informaticii în societatea modernă. *Informaţia* este deja un neofactor de producţie, deseori mult mai important decât materiile prime şi energia: "*Cine deţine informaţia – stăpâneşte situaţia*". *Informatica*, la rândul său, facilitează considerabil valorificarea eficientă a informaţiilor deja cunoscute şi, de asemenea, procesele de obţinere a unor noi informaţii, folosind mijloacele informatice, îndeosebi calculatoarele.

Calculatoarele – cea mai complexă operă a raţiunii umane, completează şi fortifică considerabil, uneori greu de imaginat, capacităţile intelectuale umane. Astfel, în timpul parcurgerii de către o rază de lumină în vid a distanţei de un singur milimetru, supercalculatorul Tianhe-2 (MilkyWay-2) al Universităţii NUDT (China) execută cca. 113000 de operaţii în virgulă mobilă. Efectul aplicării unor asemenea capacităţi de procesare a informaţiei este considerabil. Se rezolvă probleme care era imposibil de rezolvat, creşte semnificativ productivitatea şi se îmbunătăţeşte calitatea muncii.

Avantajele folosirii mijloacelor informatice au condus la implementarea lor tot mai largă. Sectorul informaticii a devenit un sector strategic de susţinere a creşterii economice şi de prosperare a societăţii. Cercetări recente arată că până la 30-50% din creşterea economică a unei ţări economic dezvoltate se datorează sectorului Tehnologiilor informaţionale şi de telecomunicaţii.

Anume impactul benefic considerabil al informaticii asupra societăţii a condus la concluzia despre evoluţia spre societatea informaţională-societatea cunoaşterii. Paşi concreţi în edificarea Societăţii informaţionale sunt

întreprinşi şi în Republica Moldova, inclusiv agenţii economici, care activează în domeniul TI, se bucură de anumite facilităţi fiscale. Solicitarea informaticienilor pe piaţa muncii este atât de înaltă că chiar şi în perioade de criză economică în multe ţări se simte un deficit de specialişti în domeniu.

Tinerii sunt viitorul societăţii, iar Dvs., ca tineri informaticieni, vă revine şi rolul onorabil de avangardă în edificarea societăţii viitorului – a societăţii informaţionale. Ne mândrim cu acest rol deosebit de important al informaticii în societate, dar şi responsabilitatea ce ne revine este pe măsură. Pentru a face faţă sarcinilor respective, trebuie să ne pregătim bine din timp, orientat şi sistematic. Conform raportului grupului Bangemann către Consiliul Europei: ţările, care nu vor întreprinde măsuri radicale orientate la edificarea intensă a societăţii informaţionale, vor rămâne economic în urmă pentru totdeauna. Nu trebuie să ne liniştească, în această privinţă, „legea vaselor comunicante" în era Internetului.

În avangardă întotdeauna se află cei mai buni. De ei depinde, în primul rând, bunăstarea zilei de mâine pentru toţi. Cu alte cuvinte, mâine voi, tinerii informaticieni de azi, veţi fi aceia, de care va depinde substanţial prosperarea acestei palme de pământ – Republica Moldova. Sarcinile sunt diverse şi fiecare din noi la fel. Astăzi unul din noi găseşte o soluţie originală, iar mâine – un altul şi în rezultat avem de câştigat cu toţii.

Lumea aparţine celor capabili şi energici. În primii cinci, din cei mai bogaţi oameni ai lumii la ora actuală, trei sunt din informatică şi telecomunicaţii: pe primul loc este William (Bill) Gates (Microsoft) – informatică, pe al doilea – Carlos Slim - telecomunicaţii şi pe al cincilea – Larry Ellison (Oracle) - informatică. Nu departe în această listă, cu zeci de mlrd de dolari SUA la activ, sunt Larry Page şi Sergey Brin (Google), Mark Zuckenberg (Facebook) şi Steve Ballmer (Microsoft). Există şi multe alte exemple demne de urmat.

Din 2007 la 17 mai se sărbătoreşte Ziua Mondială a Telecomunicaţiilor şi Societăţii informaţionale. Felicitări călduroase tuturor cu această sărbătoare – este sărbătoarea noastră a informaticienilor, şi sperăm ca cele două zile de competiţii în informatică ce urmează vă vor mobiliza la noi realizări în domeniu.

În numele Consiliului Olimpic la Informatică

Ion Bolun
prof.univ.,dr.hab. ASEM

Ministerul Educaţiei al Republicii Moldova
Olimpiada Republicană la Informatică, Ediţia 2014

Instrucţiune

I. Fişiere şi directoare

I.1. În fiecare zi de concurs, competitorul va rezolva probleme de natură algoritmică, iar fişierele sursă ale programelor elaborate, câte unul pentru fiecare problemă, le va trimite, prin reţea, pe serverul pus la dispoziţie de juriu.

I.2. Fişierele sursă vor avea aceleaşi nume ca şi fişierele de intrare/ieşire. De exemplu, în cazul fişierelor de intrare/ieşire numite `SORTARE.IN` şi `SORTARE.OUT`, fişierul sursă se va numi `SORTARE.PAS`, `SORTARE.C` sau `SORTARE.CPP`.

I.3. Fişierele de intrare/ieşire vor fi citite/scrise în catalogul curent, evitându-se introducerea căilor absolute sau relative de acces. De exemplu, în cazul limbajului PASCAL şi a fişierului `SORTARE.IN`, se va utiliza apelul assign(f,'SORTARE.IN'). Pentru un program în limbajul C, se va utiliza apelul `f=fopen("SORTARE.IN","rt")`. Orice fişiere temporare, utilizate de program în cursul execuţiei, vor fi create numai în directorul curent.

I.4. Programul nu va scrie nimic pe ecran, nu va citi nimic de la tastatură, nu va lucra cu alte fişiere decât cel de intrare şi cel de ieşire, cu excepţia cazurilor în care se indică altfel în enunţul problemei.

II. Limbaje de programare

II.1. Limbajele de programare acceptate la Olimpiadă sunt FreePascal si GnuCC, cu versiunile şi opţiunile de compilare specificate în Regulamentul Olimpiadei.

II.2. La dorinţă, competitorii pot lucra în Turbo PASCAL sau Turbo C, însă, la evaluare, compilarea programelor sursă va fi făcută cu FreePascal şi GCC.

II.3. Programul competitorului poate utiliza toate mijloace oferite de compilatorul ales, cu excepţia celor care intra în contradicţie cu Regulamentul Olimpiadei. Printre acţiunile interzise se numără: apelul de întreruperi, scrierea în porturi, resetarea timpului curent, lansarea altor programe şi/sau procese, accesarea reţelei.

II.4. Programul competitorului trebuie sa returneze sistemului de operare codul de ieşire nul. Astfel, în cazul ieşirilor forţate din program, programatorii PASCAL vor scrie halt(0), iar programatorii C vor scrie return 0 la ieşirea din `main` şi `exit(0)` în celelalte cazuri.

II.5. Competitorii vor avea la dispoziţie fişierele `C_PAS.BAT` şi `C_CPP.BAT`, care compilează programele-sursă cu exact aceleaşi opţiuni ca şi evaluatorul. Competitorii care doresc ca programele lor să fie compilate cu alte opţiuni decât cele folosite implicit de sistemul de evaluare, o pot face prin includerea opţiunilor dorite în programul-sursă, folosind mijloacele prevăzute de limbajul respectiv: `{$R-, B- ...}` în PASCAL si `#pragma` în C/C++.

III. Alte informaţii

III.1. Competitorii vor lucra în sistemul de operare Windows.

III.2. Competitorii vor avea dreptul să utilizeze orice programe instalate pe calculator, în afară de cele de configurare şi/sau monitorizare a sistemului de operare sau a reţelei. Competitorii nu au dreptul să instaleze sau să dezinstaleze programe, să şteargă şi/sau să modifice alte fişiere, decât doar cele create de ei.

Problemele pentru elevii claselor 7 – 9

Denumirea problemei	Numărul de puncte alocat problemei	Denumirea fişierului sursă	Denumirea fişierului de intrare	Denumirea fişierului de ieşire
Numere de înmatriculare	100	AUTO.PAS AUTO.C AUTO.CPP	AUTO.IN	AUTO.OUT
Regele	100	REGE.PAS REGE.C REGE.CPP	REGE.IN	REGE.OUT
Cenuşăreasa	100	CENUSRSA.PAS CENUSRSA.C CENUSRSA.CPP	CENUSRSA.IN	CENUSRSA.OUT
Sortare	100	SORTARE.PAS SORTARE.C SORTARE.CPP	SORTARE.IN	SORTARE.OUT
Focuri de artificii	100	FOCURI.PAS FOCURI.C FOCURI.CPP	FOCURI.IN	FOCURI.OUT
Bacterii	100	BACTERII.PAS BACTERII.C BACTERII.CPP	BACTERII.IN	BACTERII.OUT
Total	**600**	-	-	-

Numere de înmatriculare

Ionel este pasionat de automobile şi numerologie. Pentru fiecare automobil pe care îl vede, el calculează cifra unică a numărului de înmatriculare a acestuia şi îi dă şoferului unul din calificativele: `norocos`, `vorbaret`, `inteligent` sau `obisnuit`. Dacă cifra unică este egală cu:

- 7 – şoferul este `norocos`;
- 8 – şoferul este `vorbaret`;
- 9 – şoferul este `inteligent`;
- 0, 1, 2, 3, 4, 5 sau 6 – şoferul este `obisnuit`.

Cifra unică se calculează în modul următor. Se adună cifrele numărului de înmatriculare; dacă numărul obţinut este din două cifre, atunci acestea iarăşi se adună ş.a.m.d., până se obţine un număr dintr-o singură cifră, care şi este cifra unică.

Sarcină. Fie Ionel a întâlnit *N* automobile cu numere de înmatriculare de 3 cifre. Alcătuiţi un program, care ar determina calificativul fiecărui şofer.

Date de intrare. Fişierul text de intrare `AUTO.IN` conţine pe prima linie numărul întreg *N*, iar pe fiecare din următoarele *N* linii – câte un număr de 3 cifre, care poate avea cifre de zero şi la început.

Date de ieşire. Fişierul text de ieşire `AUTO.OUT` va conţine *N* linii, corespondente ultimelor *N* linii ale fişierului `AUTO.IN`, dar fiecare din care va specifica calificativul şoferului automobilului cu numărul de înmatriculare respectiv.

Restricţii. $1 \leq N \leq 100000$. Timpul de execuţie nu va depăşi 1 secundă. Programul va folosi cel mult 32 Megaocteţí de memorie operativă. Fişierul sursă va avea denumirea `AUTO.PAS`, `AUTO.C` sau `AUTO.CPP`.

Exemplu

AUTO.IN

```
5
123
556
044
216
999
```

AUTO.OUT

```
obisnuit
norocos
vorbaret
inteligent
inteligent
```

Rezolvare

Pentru calcularea sumei cifrelor numărului $x = \overline{abc}$ se vor utiliza formulele:

$a = x$ div 100; $b = (x$ div 10) mod 10; $c = x$ mod 10; $S = a + b + c$.

De exemplu, pentru numărul de înmatriculare 999, obţinem S = 27. Deoarece S este un număr de două cifre, atunci, conform enunţului, se va calcula suma cifrelor acestui număr, obţinând numărul 9; acesta este deja un număr dintr-o singură cifră, care şi reprezintă cifra unică, iar şoferului i se va atribui calificativul – `inteligent`. Este uşor de detereminat că, pentru fiecare număr de înmatriculare, nu vor fi mai mult de trei iteraţii de calcule.

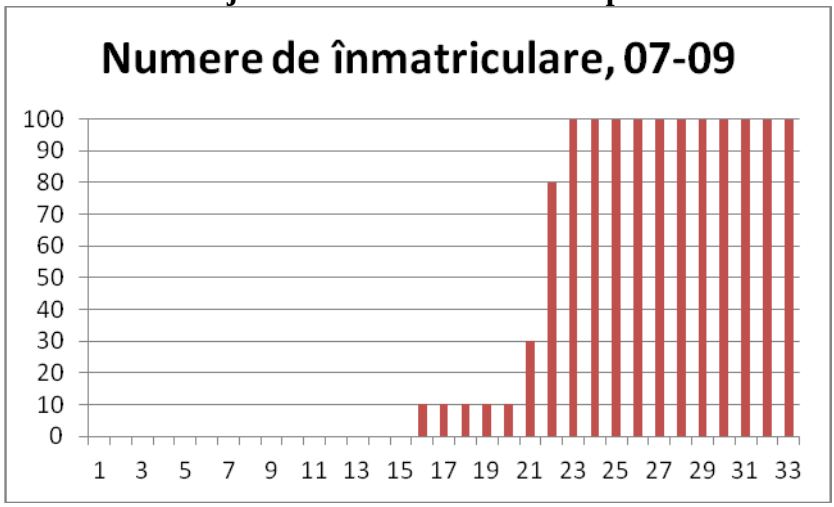
Conform restricţiilor $1 \le N \le 100000$, deci N va fi declarat de tip `longint`.

```
Program Soferii;
{clasele 07-09}

var s:integer;
    n,j:longint;
    unica:byte;
    f,g:text;

begin
  assign(f,'AUTO.IN'); reset(f);
  readln(f,n);
  assign(g,'AUTO.OUT'); rewrite(g);
  for j:=1 to n do
    begin
      readln(f,s);
      unica:=0;
      unica:=s mod 10 + (s div 10 ) mod 10 + s div 100;
```

```
      if unica>=10 then
        begin s:=unica; unica:=s mod 10 + s div 10; end;
      if unica>=10 then
        begin s:=unica; unica:=s mod 10 + s div 10; end;
      case unica of
        7:writeln(g,'norocos');
        8:writeln(g,'vorbaret');
        9:writeln(g,'inteligent')
          else writeln(g,'obisnuit')
      end;
    end;
  close(f); close(g);
end.
```

Ciclul `for` se repetă de cel mult 10^5 ori, iar numărul de operaţii în cadrul lui nu depăşeşte 25. Prin urmare timpul de execuţie a programului nu va depăşi 0,1 secunde.

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Regele

Pe o planetă îndepărtată era o împărăţie mare şi bogată Roşlandia, învecinată cu alte N ţări. Într-un vis, regele Roşu al Roşlandiei a hotărât să acapareze cât mai multe pământuri ale statelor vecine. Pentru aceasta el a adunat o armată foarte mare din S soldaţi. Fiecare stat i din cele N dispunea de o armată din A_i soldaţi – număr direct proporţional cu suprafaţa statului. Nu existau state cu acelaşi număr de soldaţi.

Lăudăros din fire, regele Roşu doreşte să supună statele după următorul principiu: statul următor de cotropit trebuie să aibă o suprafaţă mai mare, decât suma suprafeţelor tuturor statelor deja acaparate. La acapararea statului i regele pierde A_i soldaţi. Pentru a cuceri un teritoriu străin cât mai mare, regele este dispus să piardă toată armata sa.

Sarcină. Elaboraţi un program, care ar calcula pentru regele Roşu ordinea cotropirii statelor vecine, astfel ca acesta să acapareze un teritoriu de o suprafaţă cât mai mare.

Date de intrare. Fişierul text de intrare `REGE.IN` va conţine pe prima linie (linia 0) numerele N şi S, iar pe fiecare linie i, la i = 1, 2, 3, …, N, – numărul A_i.

Date de ieşire. Fişierul text de ieşire `REGE.OUT` va conţine pe prima linie două numere, separate prin spaţiu: numărul de state cucerite şi numărul de soldaţi rămaşi. Următoarele linii vor conţine numărul de soldaţi pierduţi pentru fiecare stat cucerit, câte unul pe linie, în ordinea cuceririi statelor.

Restricţii. $1 \le N \le 30$; $1 \le S$, $A_i \le 2 \cdot 10^8$. Setul de date iniţiale admite o singură soluţie a problemei. Timpul de execuţie a programului nu va depăşi 0,1 secunde. Programul va folosi cel mult 16 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `REGE.PAS`, `REGE.C` sau `REGE.CPP`.

Exemplu

REGE.IN

```
6 50
8
5
2
36
70
17
```

REGE.OUT

```
3 1
5
8
36
```

Rezolvare

Ţinând cont de restricţii ($1 \le S, A_i \le 2 \cdot 10^8$), mărimea *S*, elementele vectorului ***A*** şi alte mărimi aferente se vor declara de tip `longint`.

Deoarece statul de cotropit trebuie să aibă o suprafaţă mai mare, decât suma suprafeţelor tuturor statelor deja acaparate, soluţia B_j, $j = 1, 2, \ldots, m$, unde B_j este numărul de soldaţi ai *j*-lui stat cotropit, va fi un vector cu creştere mare, adică

$$B_j \ge B_1 + B_2 + \ldots + B_{j-1}, j = 2, 3, \ldots, m.$$

Pentru obţinerea soluţiei, elementele vectorului ***A*** se vor aranja în ordine crescătoare

$$A_1 < A_2 < A_3 < \ldots < A_N.$$

Apoi se va determina elementul A_q din condiţia

$$A_q = \max\{ A_i \le S\}.$$

Ulterior, se vor analiza elementele vectorului ***A*** de la dreapta spre stânga, începând cu cel A_q (de la A_q spre A_1). În acest proces, fiecărui asemenea element A_i i se va pune pune în corespondenţă un număr $R_i = \max\{A_i + R_{i+k} \le S, k = 1..q\text{-}i\}$, cu excepţia că $R_q = A_q$, şi, totodată, elementul A_i va forma cu cele constituente ale $R_{i+k} = R_i - A_i$ (elementele vectorului ***A***, din adunarea cărora este obţinută valoarea R_{i+k}) un vector cu creştere mare.

În sfârşit, se determină

$$R_p = \max\{R_i, i = 1..q\},$$

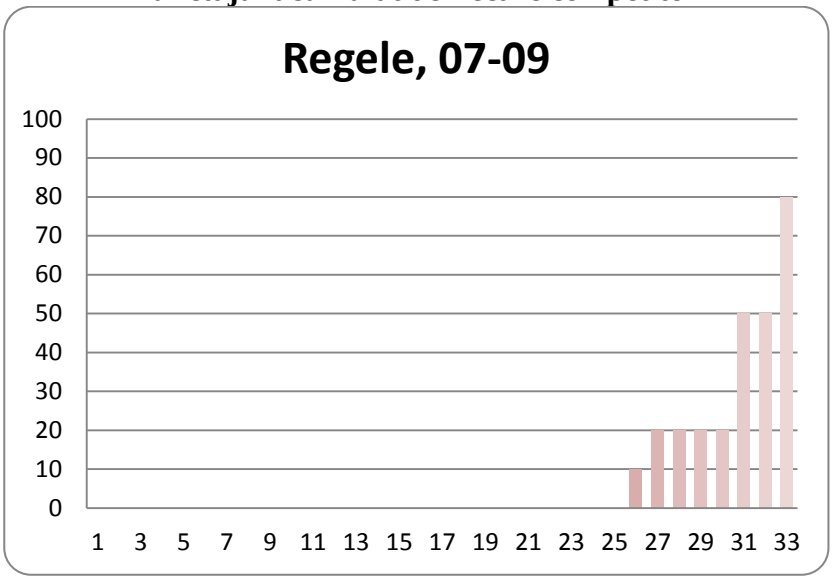
elementele constituente ale căruia şi formează soluţia scontată – numerele de soldaţi ai armatelor ţărilor cotropite de Roşlandia. Conform restrcţiilor enunţului, valoarea R_p este unică printre cele q valori ale mărimilor R_i, $i = 1..q$.

```
Program Regele;
{clasele 07-09}

type multime=set of byte;
var A,sum,pred:array[1..30] of longint;
    ind:array[1..30] of multime;
    n,i,j,k,delta:byte;
    max,s,mm:longint;
    f,g:text;

begin
  assign(f,'REGE.IN');assign(g,'REGE.OUT');
  reset(f);rewrite(g);
  readln(f,n,s);
  for i:=1 to n do
  begin
    readln(f,A[i]);
  end;
  close(f);

  for i:=1 to n do ind[n]:=[];

  for i:=1 to n-1 do
    for j:=i+1 to n do
      if A[i]>A[j] then
        begin
          mm:=A[i];
          A[i]:=A[j];
          A[j]:=mm;
        end;

  if A[n]<=s then
    begin
      sum[n]:=A[n];
      ind[n]:=[n];
      pred[n]:=A[n];
    end else pred[n]:=s;
```

```
  for k:=n-1 downto 1 do
    begin
      max:=A[k];delta:=k;
      if A[k]<=s then
      begin
        for i:=k+1 to n do
          if A[k]<pred[i] then
            begin
              mm:=A[k]+sum[i];
              if (mm<=s) and (mm>max) then
              begin
                max:=mm;delta:=i;
              end;
            end;
          sum[k]:=max;
          ind[k]:=ind[delta]+[k];
          pred[k]:=abs(pred[delta]-A[k]);
          if pred[k]>A[k] then pred[k]:=A[k];
      end;
    end;

  max:=sum[1];j:=1;
  for i:=2 to n do
    if sum[i]>max then begin max:=sum[i];j:=i;end;
  k:=0;
  for i:=1 to n do
    if i in ind[j] then inc(k);
  writeln(g,k,' ',s-max);

  for i:=1 to n do
    if i in ind[j] then writeln(g,A[i]);

  close(g);
end.
```

Complexitatea temporală a programului este de ordinul $O(n^2)$. Deoarece N =30, timpul de execuţie a programului va fi mult mai mic decât 0,1 secunde.

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Cenuşăreasa

În realitate, în povestea despre Cenuşăreasă, mama vitregă, răsturnând sacul cu boabe de mazăre pe podea, i-a poruncit Cenuşăresei să determine distanţa dintre cele mai îndepărtate una de alta boabe de mazăre. Acum Cenuşăreasă trebuie cu rigla să măsoare distanţa dintre toate boabele de mazăre înşirate pe podea.

Sarcină. Elaboraţi un program, care i-ar ajuta Cenuşăresei să reuşească la bal.

Date de intrare. Fişierul text de intrare`CENUSRSA.IN` conţine pe prima linie (linia 0) numărul total de boabe de mazăre *N*. Fiecare din următoarele *N* linii conţine câte două numere întregi x_i şi y_i, separate prin spaţiu. Numerele x_i şi y_i din linia i reprezintă coordonatele pe podea ale bobului de mazăre i.

Date de ieşire. Fişierul text de ieşire `CENUSRSA.OUT` va conţine pe prima linie un număr egal cu distanţa dintre cele mai îndepărtate una de alta boabe de mazăre cu exactitatea de până la patru cifre zecimale după virgulă.

Restricţii. $2 \le N \le 1000$; $|x_i| \le 1000$, $|y_i| \le 1000$, $i = 1, 2, 3, \ldots, N$. Timpul de execuţie nu va depăşi 0,1 secunde. Programul va folosi cel mult 32 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `CENUSRSA.PAS`, `CENUSRSA.C` sau `CENUSRSA.CPP`.

Exemplul 1

`CENUSRSA.IN`

```
3
0 1
0 0
1 1
```

`CENUSRSA.OUT`

```
1.4142
```

Exemplul 2

CENUSRSA.IN

```
4
1 1
1 -2
-3 1
0 0
```

CENUSRSA.OUT

```
5.0000
```

Rezolvare

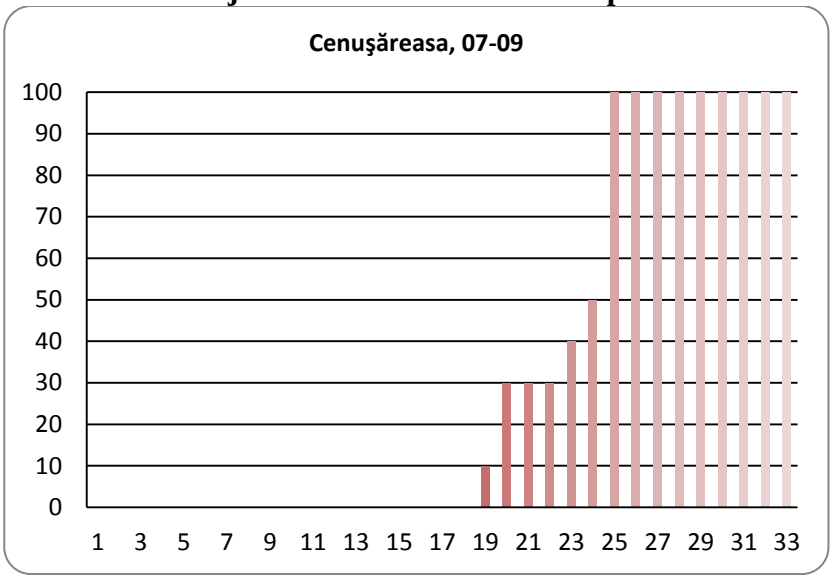
La restricțiile propuse, problema poate fi rezolvată direct, calculând distanța dintre fiecare două puncte. Pentru a nu avea probleme cu compararea numerelor zecimale, se compară pătratele distanțelor dintre perechile de puncte.

```
Program Cenusareasa;
{clasele 07-09}

const MAX_BEANS = 1000;

var x, y: array [1 .. MAX_BEANS] of point;
      num_beans, i, j: integer;
    distance: real;
    distance2, tmp: longint;
      fin, fout: textfile;
begin
    assign(fin, 'CENUSRSA.IN');
    reset(fin);
    assign(fout, 'CENUSRSA.OUT');
    rewrite(fout);

readln(fin, num_beans);
for i := 1 to num_beans do
      readln(fin, x[i], y[i]);
    close(fin);

    distance2 := 0;
```

```
    for i := 1 to num_beans-1 do
        for j := i+1 to num_beans do
        begin
          tmp := (x[i] - x[j])*( x[i] - x[j]) +
(y[i] - y[j])*(y[i] - y[j]);
          if tmp > distance2 then distance2 := tmp;
        end;

    distance := sqrt(distance2);
    write(fout, distance:4:4);
    close(fout);
end.
```

Numărul de segmente ce unesc N puncte direct este egal cu $N(N-1)/2$. La $N = 1000$, obţinem 499500 segmente.

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Sortare

Fie un tablou din *N* numere întregi, ce nu depăşesc 100000 după modul.

Sarcină. Alcătuiţi un program, care ar determina numărul minimal de permutări de numere în perechi, necesare pentru a sorta tabloul dat în ordine crescătoare. Prin permutare de numere în perechi se are în vedere schimbul cu locul în tablou al celor două numere ale perechii respective.

Date de intrare. Fişierul text `SORTARE.IN` conţine pe prima linie numărul întreg *N*, iar pe a doua linie *N* numere întregi, mai mici de 100000 după modul, separate prin spaţiu.

Date de ieşire. Fişierul text `SORTARE.OUT` va conţine pe prima linie numărul minimal de permutări de numere în perechi, necesar pentru a sorta tabloul dat în ordine crescătoare.

Restricţii. $0 \leq N \leq 1000$. Timpul de execuţie nu va depăşi 0,1 secunde. Programul va folosi cel mult 16 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `SORTARE.PAS`, `SORTARE.C` sau `SORTARE.CPP`.

Exemplul 1

`SORTARE.IN`

```
5
8 6 4 2 1
```

`SORTARE.OUT`

```
2
```

Exemplul 2

`SORTARE.IN`

```
3
-10 5 2
```

`SORTARE.OUT`

```
1
```

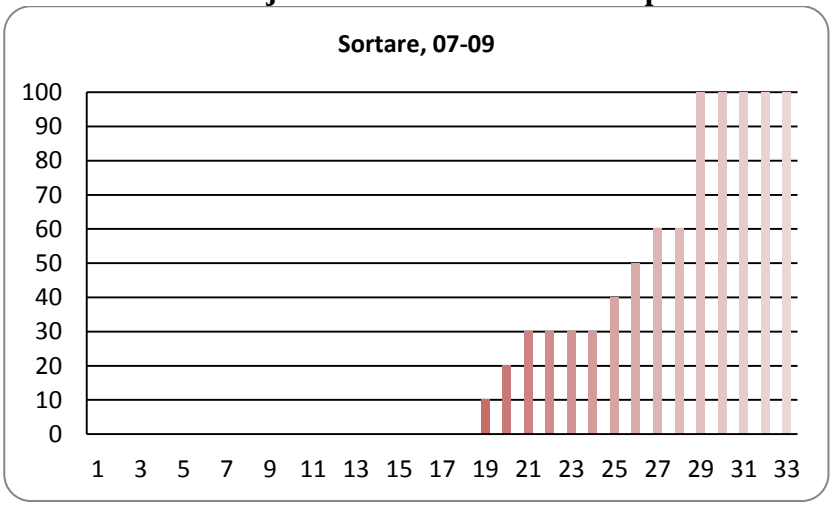
Rezolvare

Cea mai simplă cale de rezolvare a acestei probleme constă în utilizarea sortării prin selecție simplă. Conform acestui algoritm, elementele nimeresc pe locul său maximum după o permutare. Astfel, soluționarea problemei se reduce la realizarea sortării prin metoda de selecție simplă cu calcularea concomitentă a numărului de permutări.

```
Program Sortare;
{clasele 07-09}

var v: array [1..1000] of integer;
    count, i, j, min, tmp, size: integer;
    stdin, stdout: TextFile;
begin
  assign(stdin, 'SORTARE.IN');
  reset(stdin);
  assign(stdout, 'SORTARE.OUT');
  rewrite(stdout);

  count := 0;

  readln(stdin, size);
  for i := 1 to size do
  begin
    read(stdin, v[i]);
  end;
  close(stdin);

  for i := 1 to size-1 do
  begin
    min := i;
    for j := i+1 to size do
    begin
      if v[min] > v[j] then min := j;
    end;
    if min <> i then
    begin
      inc(count);
      tmp := v[min];
      v[min] := v[i];
      v[i] := tmp;
```

```
    end
  end;
  write(stdout, count);
  close(stdout);
end.
```

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Focuri de artificii

Organizatorii Jocurilor Olimpice de iarnă de la Sochi din 2014 şi-au propus realizarea unui adevărat spectacol de focuri de artificii. În acest scop se folosesc rachete speciale confecţionate în blocuri. Într-un bloc sunt N rachete. Fiecare rachetă i, $i = 1, 2, 3, \ldots, N$, se caracterizează prin următoarele proprietăţi:

- h_i – înălţimea rachetei faţă de orizont;
- $[a_i; b_i]$ – intervalul de armonie a rachetei.

Orice rachetă a blocului poate fi lansată prima. Rachetă $j < i$ poate fi lansată imediat după cea i a blocului doar dacă $a_j \le h_i \le b_j$; altfel aceasta nu poate fi lansată.

Sarcină. Elaboraţi un program, care ar ajuta organizatorii Jocurilor Olimpice să determine o astfel de ordine de lansare a rachetelor, ca numărul de rachete ale blocului dat ce pot fi lansate să fie maximal.

Date de intrare. Fişierul text de intrare `FOCURI.IN` conţine pe prima linie (linia 0) numărul натурал N de rachete în bloc. Fiecare linie i, din următoarele N linii, conţine câte trei numere naturale h_i, a_i şi b_i, separate prin spaţiu, ce caracterizează racheta i a blocului de rachete dat.

Date de ieşire. Fişierul text de ieşire `FOCURI.OUT` va conţine pe prima linie numărul natural ce caracterizează numărul maximal de rachete ale blocului dat ce pot fi lansate.

Restricţii. $0 \le N \le 3000$; $0 \le a_i, b_i, h_i \le 1000000$. Timpul de execuţie a programului nu va depăşi 0,5 secunde. Programul va folosi cel mult 32 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `FOCURI.PAS`, `FOCURI.C` sau `FOCURI.CPP`.

Exemplu

FOCURI.IN

```
5
9 0 6
5 7 8
7 0 6
2 1 5
9 3 8
```

FOCURI.OUT

```
4
```

Rezolvare

Din descrierea condiţiilor de lansare a unei rachete *j* după o rachetă *i* se pot deduce următoarele:

- ordinea apariţiei rachetelor în fişierul de intrare e semnificativă;
- pentru fiecare rachetă *j*, racheta care va fi lansată după ea apare in fişierul de intrare înainte de ea.

Dacă definim $chainLen_i$ ca fiind "gradul maxim posibil de frumuseţe a unui spectacol care începe cu racheta *i*", atunci observăm că:

$$chainLen_i = 1 + \max_{k=0..i-1} chainLen_k, \text{daca } a_i \le h_k \le b_i .$$

Odată tabelul *chainLen* (cu elementele $chainLen_i$, $i = 1, 2, \ldots, N$) construit aşa, rezultatul final (răspunsul la problemă) va fi elementul maxim din *chainLen*.

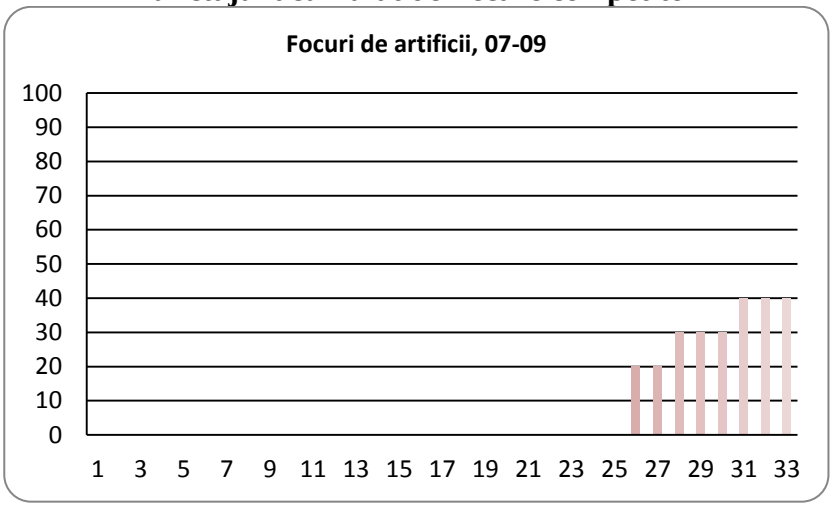
```
Program Focuri;
{clasele 07-09}

uses math;
type Racheta = record
    h: longint;
    a: longint;
    b: longint;
end;

const MAXN = 5000;
var rachete: array[1..MAXN] of Racheta;
    chainLen: array[1..MAXN] of longint;
```

```
{calculeaza daca racheta a poate fi lansata dupa
racheta b}
function poateFiLansataDupa(var a: Racheta; var b:
Racheta): boolean;
begin
  poateFiLansataDupa := (a.a <= b.h) and (b.h <=
a.b);
end;

var n, i, j: integer;
f: text;
res, maxLen: longint;
begin
  assign(f, 'FOCURI.IN');
  reset(f);
  {citim datele de intrare}
  readln(f, n);
  for i:=1 to n do
    readln(f, rachete[i].h, rachete[i].a,
rachete[i].b);
  close(f);
      {rezolvare}
  for i:=1 to n do
    begin
      maxLen:=0;
    for j:=1 to i-1 do
      begin
        if poateFiLansataDupa(rachete[j],
rachete[i]) and (chainLen[j] > maxLen) then
          begin
            maxLen := chainLen[j];
          end;
      end;
    chainLen[i] := maxLen + 1;
    end;
    res := 0;
    for i := 1 to n do
      res := max(res, chainLen[i]);
      {scriere date iesire}
    assign(f, 'FOCURI.OUT');
      rewrite(f);
      writeln(f, res);
      close(f);
end.
```

Consumul de resurse al algoritmului se deduce în mod direct din descrierea acestuia:

- consumul de timp procesor – pentru calcularea lui $chainLen_i$ se fac i-1 iteratii, deci numărul necesar de operaţii este $1+2+3+...+(n-1)=n(n-1)/2=O(n^2)$ şi se încadrează lejer în restricţia de timp impusă;
- consumul de memorie – se consumă spaţiu pentru a păstra descrierea rachetelor, precum şi tabelul *chainLen*. Deci consumul de memorie este $O(n)$.

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Bacterii

Lena este pasionată de biologie. Efectuând diverse experimente, ea a descoperit o nouă specie de bacterii cu proprietăţi interesante: în primul minut al vieţii sale bacteria nu se reproduce (o vom numi bacterie tânără), iar în fiecare din următoarele minute ea se divide (o vom numi bacterie matură), dând naştere unei bacterii tinere. Lena a hotărât să determine câte bacterii vor fi după o perioadă de timp anumită.

Sarcină. Elaboraţi un program, care i-ar ajuta Lenei să calculeze numărul de bacterii după o perioadă de timp dată, în condiţiile că în această perioadă nu moare nici o bacterie.

Date de intrare. Fişierul text de intrare `BACTERII.IN` conţine pe prima linie trei numere naturale *p*, *q* şi *N*, separate prin spaţiu, ce reprezintă numărul de bacterii mature (*p*) şi tinere (*q*) la începutul experimentului şi durata experimentului în minute (*N*).

Date de ieşire. Fişierul text de ieşire `BACTERII.OUT` va conţine pe prima linie numărul total de bacterii (tinere şi mature) în populaţie după *N* minute ale experimentului.

Restricţii. $0 \leq p, q, N \leq 1000$. Timpul de execuţie nu va depăşi 0,75 secunde. Programul va folosi cel mult 128 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `BACTERII.PAS`, `BACTERII.C` sau `BACTERII.CPP`.

Exemplu

`BACTERII.IN`	`BACTERII.OUT`
2 3 3	19

Rezolvare

Soluționarea problemei se reduce la construirea unui șir ce seamănă după proprietăți cu șirul Fibbonacci. Dacă a_i este numărul de bacterii tinere și b_i este numărul de bacterii mature în minutul i, atunci în minutul următor numărul de bacterii va fi egal cu:

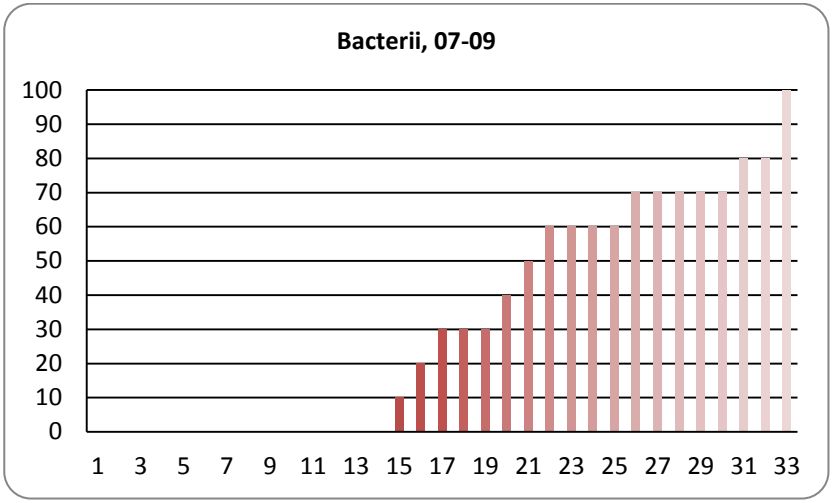
$$\begin{cases} a_{i+1} = b_i \\ b_{i+1} = b_i + a_i. \end{cases}$$

La definirea variabilelor trebuie de ţinut cont că şirurile a_i şi b_i cresc exponenţial.

```
Program Bacterii;
{clasele 07-09}

const MAX_SIZE = 600;
  DIVIDER = 10000;

var   young,   young0,   mature,   mature0:   array
[1..MAX_SIZE] of integer;
  p, q, t: longint;
  i, j, k, aux, res: integer;
  stdin, stdout: TextFile;

begin
  assign(stdin, 'BACTERII.IN');
  reset(stdin);
  read(stdin, p, q, t);
  close(stdin);

  assign(stdout, 'BACTERII.OUT');
  rewrite(stdout);

  for i:=1 to MAX_SIZE do
  begin
    young[i] := 0;
    mature[i] := 0;
  end;
  young[1] := q;
  mature[1] := p;

  while t > 0 do
  begin
    young0 := young;
    mature0 := mature;
    young := mature0;
```

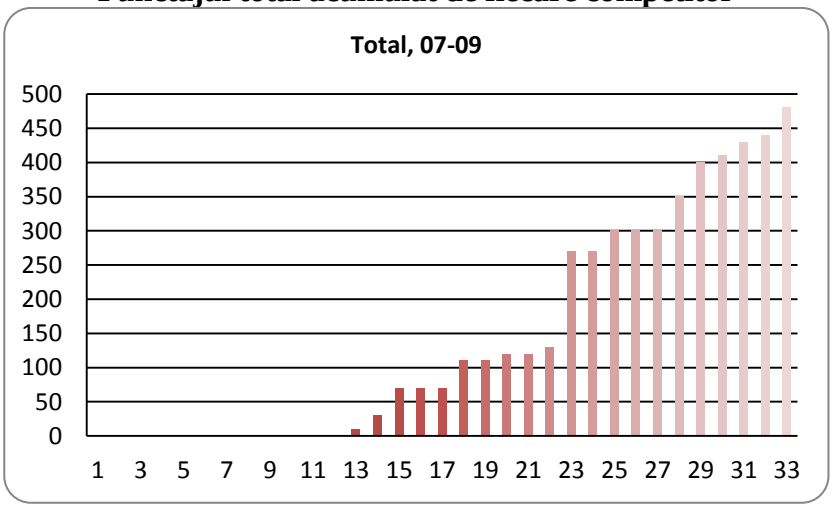
```
    for i:=1 to MAX_SIZE do
    begin
      aux := young0[i] + mature0[i] + res;
      mature[i] := aux mod DIVIDER;
      res := aux div DIVIDER;
      {mature := Add(young0, mature0);}
    end;
    dec(t);
  end;

  for i:=1 to MAX_SIZE do
  begin
    aux := young[i] + mature[i] + res;
    mature[i] := aux mod DIVIDER;
    res := aux div DIVIDER;
  end;

  j:= MAX_SIZE;
  while (j <> 1) and (mature[j] = 0) do
    dec(j);

  write (stdout, mature[j]);

  for i := j-1 downto 1 do
  begin
    k := DIVIDER;
    aux := mature[i];
    while k > 1 do
    begin
      k := k div 10;
      write (stdout, aux div k);
      aux := aux mod k;
    end;
  end;

  close(stdout);
end.
```

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Punctajul total acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Problemele pentru elevii claselor 10 – 12

Denumirea problemei	Numărul de puncte alocat problemei	Denumirea fişierului sursă	Denumirea fişierului de intrare	Denumirea fişierului de ieşire
Numere de înmatriculare	100	AUTO.PAS AUTO.C AUTO.CPP	AUTO.IN	AUTO.OUT
Regele	100	REGE.PAS REGE.C REGE.CPP	REGE.IN	REGE.OUT
Tripletele lui Pitagora	100	TRIPLETE.PAS TRIPLETE.C TRIPLETE.CPP	TRIPLETE.IN	TRIPLETE.OUT
Submulţimi	100	SMULTIMI.PAS SMULTIMI.C SMULTIMI.CPP	SMULTIMI.IN	SMULTIMI.OUT
Focuri de artificii	100	FOCURI.PAS FOCURI.C FOCURI.CPP	FOCURI.IN	FOCURI.OUT
Bacterii	100	BACTERII.PAS BACTERII.C BACTERII.CPP	BACTERII.IN	BACTERII.OUT
Total	**600**	-	-	-

Numere de înmatriculare

Ionel este pasionat de automobile şi numerologie. Pentru fiecare automobil pe care îl vede, el calculează cifra unică a numărului de înmatriculare a acestuia şi îi dă şoferului unul din calificativele: `norocos`, `vorbaret`, `inteligent` sau `obisnuit`. Dacă cifra unică este egală cu:

- 7 – şoferul este `norocos`;
- 8 – şoferul este `vorbaret`;
- 9 – şoferul este `inteligent`;
- 0, 1, 2, 3, 4, 5 sau 6 – şoferul este `obisnuit`.

Cifra unică se calculează în modul următor. Se adună cifrele numărului de înmatriculare; dacă numărul obţinut este din două cifre, atunci acestea iarăşi se adună ş.a.m.d., până se obţine un număr dintr-o singură cifră, care şi este cifra unică.

Sarcină. Fie Ionel a întâlnit *N* automobile cu numere de înmatriculare de 5 cifre. Alcătuiţi un program, care ar determina calificativul fiecărui şofer.

Date de intrare. Fişierul text de intrare `AUTO.IN` conţine pe prima linie numărul întreg *N*, iar pe fiecare din următoarele *N* linii – câte un număr de 5 cifre, care poate avea cifre de zero şi la început.

Date de ieşire. Fişierul text de ieşire `AUTO.OUT` va conţine *N* linii, corespondente ultimelor *N* linii ale fişierului `AUTO.IN`, dar fiecare din care va specifica calificativul şoferului automobilului cu numărul de înmatriculare respectiv.

Restricţii. $1 \leq N \leq 1000000$. Timpul de execuţie nu va depăşi 2 secunde. Programul va folosi cel mult 32 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `AUTO.PAS`, `AUTO.C` sau `AUTO.CPP`.

Exemplu

AUTO.IN

```
5
12300
05056
00404
20106
90909
```

AUTO.OUT

```
obisnuit
norocos
vorbaret
inteligent
inteligent
```

Rezolvare

Pentru calcularea sumei cifrelor numărului $x = \overline{abcde}$ se vor utiliza formulele:

$a = x$ div 10000

$b = (x$ div 1000) mod 10

$c = (x$ div 100) mod 10

$d = (x$ div 10) mod 10

$e = x$ mod 10

$S = a + b + c + d + e$.

De exemplu, pentru numărul de înmatriculare 99999, obţinem S = 45. Deoarece S este un număr de două cifre, atunci, conform enunţului, se va calcula suma cifrelor acestui număr, obţinând numărul 9; acesta este deja un număr dintr-o singură cifră, care şi reprezintă cifra unică, iar şoferului i se va atribui calificativul – `inteligent`. Este uşor de detereminat că, pentru fiecare număr de înmatriculare, nu vor fi mai mult de trei iterţaţii de calcule.

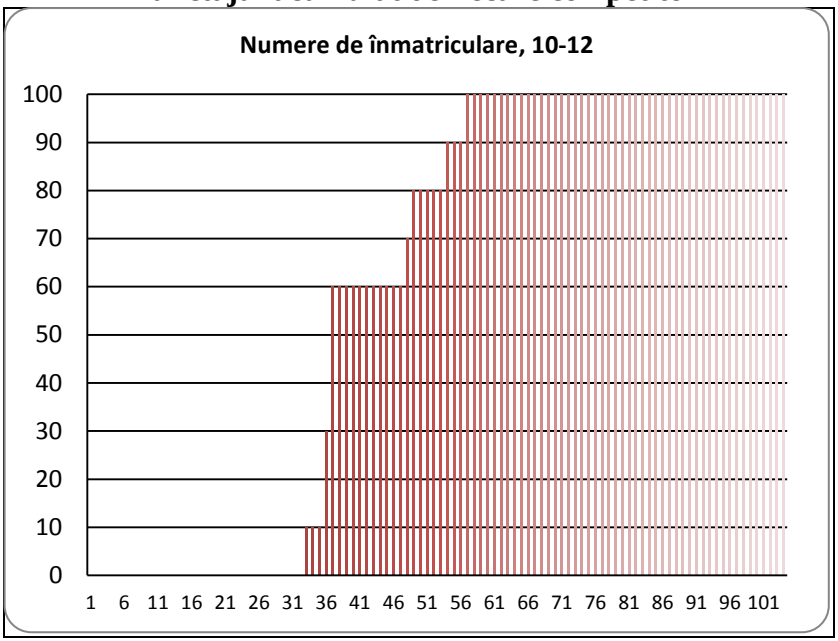
Conform restricţiilor $1 \leq N \leq 1000000$, deci N va fi declarat de tip `longint`.

```
Program Soferii;
{clasele 10-12}

var s,n,j:longint;
    unica:byte;
    f,g:text;

Begin
  assign(f,'AUTO.IN');
```

```
  reset(f);
  readln(f,n);
  assign(g,'AUTO.OUT');
  rewrite(g);
  for j:=1 to n do
    begin
      readln(f,s);
      unica:=0;
      unica:=s mod 10 + (s div 10 ) mod 10 +
      (s div 100) mod 10 + (s div 1000) mod 10 +
       s div 10000;
      if unica>=10 then
        begin s:=unica; unica:=s mod 10 + s div 10;
        end;
      if unica>=10 then
        begin
          s:=unica; unica:=s mod 10 + s div 10;
        end;
      case unica of
        7:writeln(g,'norocos');
        8:writeln(g,'vorbaret');
        9:writeln(g,'inteligent')
          else writeln(g,'obisnuit')
      end;
    end;
  close(f);
  close(g);
end.
```

Ciclul `for` se repetă de cel mult 10^6 ori, iar numărul de operaţii în cadrul lui nu depăşeşte 100. Prin urmare timpul de execuţie a programului nu va depăşi 2 secunde.

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Regele

Pe o planetă îndepărtată era o împărăţie mare şi bogată Roşlandia, învecinată cu alte N ţări. Într-un vis, regele Roşu al Roşlandiei a hotărât să acapareze cât mai multe pământuri ale statelor vecine. Pentru aceasta el a adunat o armată foarte mare din S soldaţi. Fiecare stat i din cele N dispunea de o armată din A_i soldaţi – număr direct proporţional cu suprafaţa statului. Nu existau state cu acelaşi număr de soldaţi.

Lăudăros din fire, regele Roşu doreşte să cucerească statele după următorul principiu: statul următor de cotropit trebuie să aibă o suprafaţă mai mare decât suma suprafeţelor tuturor statelor deja acaparate. La cotropirea statului i regele pierde A_i soldaţi. Pentru a cuceri un teritoriu străin cât mai mare, regele este dispus să piardă toată armata sa.

Sarcină. Alcătuiţi un program, care ar calcula pentru regele Roşu ordinea cotropirii statelor vecine, astfel ca acesta să acapareze un teritoriu de o suprafaţă cât mai mare.

Date de intrare. Fişierul text de intrare `REGE.IN` va conţine pe prima linie (linia 0) numerele N şi S, iar pe fiecare linie i, la i = 1, 2, 3, …, N, – numărul A_i.

Date de ieşire. Fişierul text de ieşire `REGE.OUT` va conţine pe prima linie două numere întregi, separate prin spaţiu: numărul de state cucerite şi numărul de soldaţi rămaşi ai Roşlandiei. Următoarele linii vor conţine numărul de soldaţi pierduţi pentru fiecare stat cucerit, câte unul pe linie, în ordinea cuceririi statelor.

Restricţii. $1 \le N \le 255$; $1 \le S \le 10^{255}$; $1 \le A_i \le 10^{255}$. Setul de date iniţiale admite o singură soluţie a problemei. Timpul de execuţie a programului nu va depăşi 2 secunde. Programul va folosi cel mult 16 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `REGE.PAS`, `REGE.C` sau `REGE.CPP`.

Exemplu

REGE.IN

```
6 50
8
5
2
36
70
17
```

REGE.OUT

```
3 1
5
8
36
```

Rezolvare

Ţinând cont de restricţii ($1 \le S$, $A_i \le 10^{255}$), mărimea S şi elementele vectorului ***A*** se vor declara de tip `string`. Pentru elementele vectorului ***A*** se vor utiliza structuri dinamice de date şi anume – liste dublu înlănţuite de structura:

```
type ref=^Structura;
     Structura=record
       nr:byte;
       v,sum,pred:string;  {în v se reţin valorile Ai}
       anti,next:ref;
     end;
```

Deoarece statul de cotropit trebuie să aibă o suprafaţă mai mare, decât suma suprafeţelor tuturor statelor deja acaparate, soluţia B_j, j = 1, 2, …, m, unde B_j este numărul de soldaţi ai j-lui stat cotropit, va fi un vector cu creştere mare, adică

$$B_j \ge B_1 + B_2 + \ldots , + B_{j-1}, j = 2, 3, \ldots, m.$$

Pentru obţinerea soluţiei, elementele vectorului ***A*** se vor aranja în ordine crescătoare

$$A_1 < A_2 < A_3 < \ldots < A_N.$$

Apoi se va determina elementul A_q din condiţia

$$A_q = \max\{ A_i \le S\}.$$

Ulterior, se vor analiza elementele vectorului ***A*** de la dreapta spre stânga, începând cu cel A_q (de la A_q spre A_1). În acest proces, fiecărui

asemenea element A_i i se va pune pune în corespondenţă un număr $R_i = \max\{A_i + R_{i+k} \le S,\ k = 1..q\text{-}i\}$, cu excepţia că $R_q = A_q$, şi, totodată, elementul A_i va forma cu cele constituente ale $R_{i+k} = R_i - A_i$ (elementele vectorului ***A***, din adunarea cărora este obţinută valoarea R_{i+k}) un vector cu creştere mare.

În sfârşit, se determină

$$R_p = \max\{R_i,\ i = 1..q\},$$

elementele constituente ale căruia şi formează soluţia scontată – numerele de soldaţi ai armatelor ţărilor cotropite de Roşlandia. Conform restrcţiilor enunţului, valoarea R_p este unică printre cele q valori ale mărimilor R_i, $i = 1..q$.

Pentru efectuarea operaţiilor aritmetice cu numere mari, se definesc operaţiile de adunate şi scădere utilizînd string-uri. Funcţiile `Less` şi `Less1` permit compararea numerelor mari.

```
Program regele;
{clasele 10-12}

uses crt;
type ref=^Structura;
     Structura=record
      nr:byte;
      v,sum,pred:string;
      prev,next:ref;
     end;
     multime=set of byte;

var prim,ultim,curent,tempo:ref;
    ind:array[1..255] of multime;
    n,i,j,k,delta:byte;
    max,s,mm:string;
    f:text;

Function Less(var p,q:string):boolean;
var ff:boolean;
begin
  ff:=false;
  if length(p)<length(q) then ff:=true;
  if (length(p)=length(q)) and (p<q) then ff:=true;
  Less:=ff;
```

```
end;

Function Less1(var p,q:string):boolean;
var ff:boolean;
begin
  ff:=false;
  if length(p)<length(q) then ff:=true;
  if (length(p)=length(q)) and (p<=q) then
ff:=true;
  Less1:=ff;
end;

Procedure Sortare(var First,Last:ref);
var q,temp:ref;
    j:byte;
begin
 new(First);
 readln(f,first^.v);
 first^.prev:=nil;
 first^.next:=nil;
 for j:=2 to n do begin
   new(q);
   readln(f,q^.v);
   if Less(q^.v,first^.v)
   then begin
       q^.next := first;
       first^.prev := q;
       q^.prev := nil;
       first := q;
     end
   else begin
       temp := first;
       while (temp^.next<>nil) and
         Less(temp^.next^.v,q^.v) do
         temp := temp^.next;
       q^.next := temp^.next;
       q^.prev := temp;
       temp^.next := q;
     end;
  end;
  temp := first;
  first^.prev := nil;
```

```
  while temp <> nil do begin
    last := temp;
    q := temp;
    temp := temp^.next;
    if (temp <> nil) then temp^.prev := q;
   end;
end;

Function Cautare(dd:byte):ref;
var temp:ref;
begin
  temp := prim;
  while temp<>nil do begin
    if temp^.nr=dd then begin Cautare:=temp;
                              break; end;
    temp:=temp^.next;
  end;
end;

Function cif(car:char):byte;
begin cif := ord(car) - ord('0'); end;

Function lit(q:byte):char;
begin lit := chr(ord('0') + q); end;

Function adunare(a1, a2:string):string;
var res,aux:string;
    l1:char;
    i,j,zero,rest,c:integer;
begin
 res := '';
 rest := 0;
 if length(a1)>length(a2) then begin
        aux := a1;
        a1 := a2;
        a2 := aux;
      end;
  zero := length(a2) - length(a1);
  for j := 1 to zero do insert('0',a1,1);
  for i := length(a1) downto 1 do begin
   c := cif(a1[i]) + cif(a2[i]) + rest;
   rest := c div 10;
```

```
   c := c mod 10;
   insert(lit(c), res, 1);
  end;
  if rest <> 0 then insert(lit(rest), res, 1);
  adunare := res;
end;

Function scad(a1,a2:string):string;
var res:string;
    l1:char;
    i,zero,c:integer;
begin
  res := '';
  zero := length(a1) - length(a2);
  for i := 1 to zero do insert('0',a2,1);
  zero := 1;
  for i := length(a1) downto 1 do begin
   c := cif(a1[i]) - 1 + zero + 10 - cif(a2[i]);
   l1 := lit(c mod 10);
   zero := c div 10;
   insert(l1, res, 1);
  end;
 while (res[1]='0') and (res <> '') do
delete(res,1,1);
 if (res = '') then res := '0';
 scad := res;
end;

Begin
  clrscr;
  assign(f,'REGE.IN');reset(f);
  readln(f,n,s);
  while s[1]=' ' do delete(s,1,1);
  sortare(prim,ultim);
  close(f);

  curent:=prim;i:=1;
  while curent<>nil do
   begin
    ind[i]:=[];
    curent^.nr:=i;
    curent^.pred:='0';
```

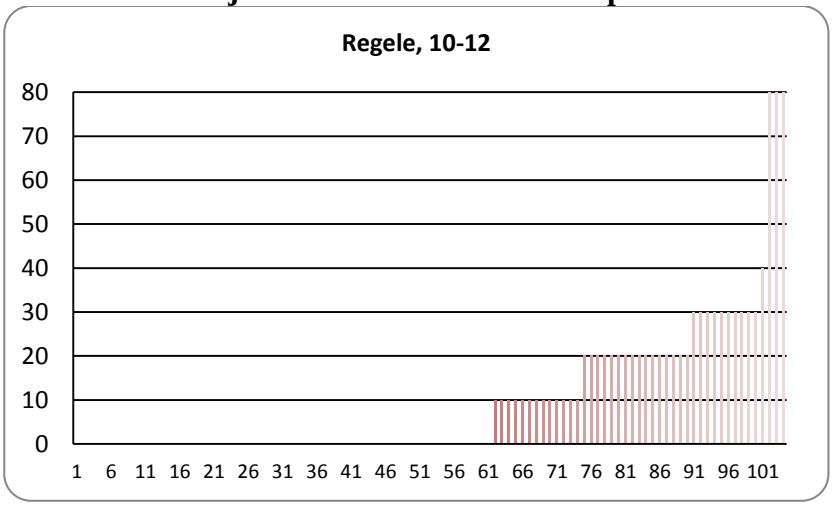
```
    curent^.sum:='0';
    curent:=curent^.next;
    inc(i);
   end;

 curent:=ultim;
 if Less1(curent^.v,s)
  then
    begin
     curent^.sum:=curent^.v;
     ind[n]:=[n];
     curent^.pred:=curent^.v;
    end
  else
    begin
     curent^.pred:=s;
     curent^.sum:='0';
    end;

 curent:=curent^.prev;k:=n-1;
 while curent<>nil do {ciclul dupa k - inainte}
  begin
   max:=curent^.v;delta:=k;

   if Less1(curent^.v,s) then
    begin
     tempo:=curent^.next;
     i:=k+1; {ciclul dupa i}
     while tempo<>nil do
      begin
       if Less(curent^.v,tempo^.pred) then
        begin
         mm:=adunare(curent^.v,tempo^.sum);
         if Less1(mm,s) and Less(max,mm) then
           begin max:=mm;delta:=i;end;
         end;
         tempo:=tempo^.next;inc(i);
      end; {while tempo}
     curent^.sum:=max;
     ind[k]:=ind[delta]+[k];
     if delta=k then
curent^.pred:=scad(curent^.v,cautare(delta)^.pred)
```

```
      else
curent^.pred:=scad(cautare(delta)^.pred,curent^.v);
     if Less(curent^.v,curent^.pred) then
curent^.pred:=curent^.v;
    end {if}
   else
    begin
     curent^.pred:=s;
     curent^.sum:='0';
    end;
   dec(k);
   curent:=curent^.prev;
  end; {while curent}

 tempo:=prim;
 max:=tempo^.sum;j:=1;
 while tempo<>nil do
  begin
   if Less(max,tempo^.sum) then begin
         max:=tempo^.sum;j:=tempo^.nr; end;
   tempo:=tempo^.next;
  end;

 assign(f,'REGE.OUT');
 rewrite(f);

 for i:=1 to n do
  if i in ind[j] then inc(k);
 mm:=scad(s,max);
 writeln(f,k,' ',mm);
 if mm='' then mm:='0';
 for i:=1 to n do
  if i in ind[j] then writeln(f,cautare(i)^.v);
 close(f);

End.
```

Complexitatea temporala a programului este de ordinul $O(n^2)$. Deoarece n=255, numărul de operaţii efectuate în program este destul de mic în comparaţie cu capacitatea de prelucrare a

calculatoarelor personale din laboratorul de Informatică. Timpul de calcul cerut de program nu va depăşi 2 secunde.

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Tripletele lui Pitagora

Toţi cunosc teorema lui Pitagora, care afirmă ca suma pătratelor catetelor este egală cu pătratul ipotenuzei unui triunghi dreptunghiular. Dar nu toţi cunosc faptul, că numerele întregi *a*, *b*şi *c*, prime între ele, care satisfac ecuaţia $a^2 + b^2 = c^2$, se numesc triplete pitagoreice. Cel mai cunoscut triplet al lui Pitagora este {3, 4, 5}, pentru care are loc $3^2 + 4^2 = 5^2$.

Sarcină. Alcătuiţi un program, care ar determina toate tripletele lui Pitagora cu numărul natural dat *c*.

Date de intrare. Fişierul text de intrare `TRIPLETE.IN` conţine pe prima linie numărul natural *c*.

Date de ieşire. Fişierul text de ieşire `TRIPLETE.OUT` va conţine toate tripletele pitagoreice cu numărul *c* respectiv, câte un triplet pe linie, sortate în ordinea crescătoare după cel mai mic număr din triplet. Dacă triplete nu există – se va afişa cuvântul `NOPE`.

Restricţii. $0 < c \leq 1000000$. Timpul de execuţie a programului nu va depăşi 0,1 secunde. Programul va folosi cel mult 16 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `TRIPLETE.PAS`, `TRIPLETE.C` sau `TRIPLETE.CPP`.

Exemplul 1

TRIPLETE.IN	TRIPLETE.OUT
2	NOPE

Exemplul 2

TRIPLETE.IN	TRIPLETE.OUT
5	3 4 5

Exemplul 3

TRIPLETE.IN	TRIPLETE.OUT
3445	83 3444 3445 1044 3283 3445 1596 3053 3445 2387 2484 3445

Rezolvare

Dacă s-ar încerca rezolvarea directă a problemei, atunci obţinerea soluţiei nu s-ar încadra în limitele de timp, deoarece parcurgerea ciclurilor pentru *a* şi *b* necesită $10^6 \cdot 10^6 = 10^{12}$ operaţii. Reducerea numărului de operaţii este posibilă în baza analizei formulei $a^2 + b^2 = c^2$. Numerele *a* şi *b* nu pot fi ambele pare, deoarece atunci ele nu ar fi prime între ele (ar avea divizorul comun 2). Totodată, *a* şi *b* nu pot fi ambele impare, fiindcă atunci suma pătratelor lor s-ar împărţi la 2, dar nu s-ar împărţi la 4; deci nu ar putea fi pătratul unui număr întreg.

Fie *a* este par, iar *b* este impar. Atunci avem

$$a^2 = c^2 - b^2 = (c - b)(c + b) = 2u \cdot 2v,$$

unde $2u = c - b$ şi $2v = c + b$, iar numerele *u* şi *v* sunt prime între ele, de unde reiese că ele sunt pătratele unor numere întregi. Atunci obţinem următoarele formule

$$a^2 + b^2 = c^2 \Rightarrow \begin{cases} a^2 = 4m^2n^2 \\ c - b = 2n^2 \\ c + b = 2m^2 \end{cases} \Rightarrow \begin{cases} a = 2mn \\ b = m^2 - n^2, m > n \\ c = m^2 + n^2 \end{cases}$$

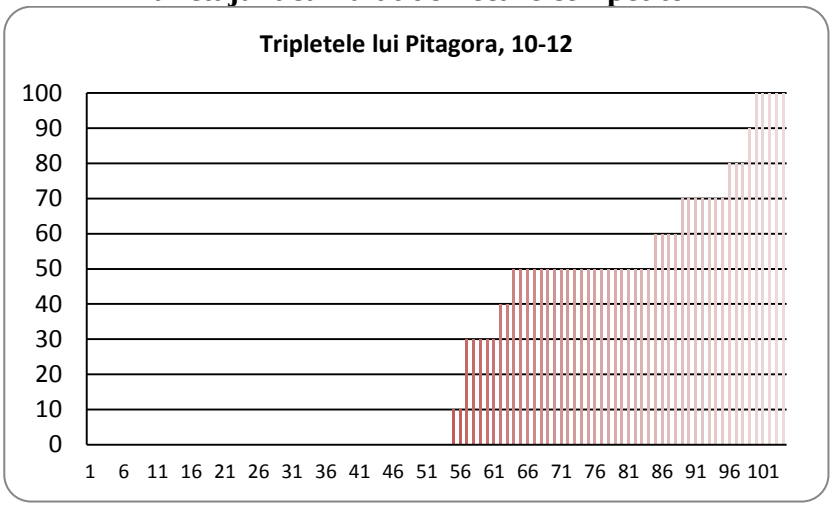
Utilizarea acestor formule permite reducerea parcurgerii ciclurilor pentru *a* şi *b* până la $10^3 \cdot 10^3 = 10^6$ operaţii. După generarea unui triplet, este necesară verificarea numerelor *a*, *b* şi *c* la lipsa divizorilor comuni. Dacă acestea sunt prime între ele (nu au divizori comuni), atunci tripletul în cauză este unul pitagoreic.

```
Program Triplete;
{clasele 10-12}

const MAX = 40000;
type triple = record
     a, b, c: LongInt;
end;

function gcd(a, b: LongInt): longInt;
begin
  gcd := a;
  if b > 0 then gcd := gcd(b, a mod b);
end;
```

```
var  m, n, size, i, aux: longint;
     tmp:triple;
     result: array[1..100] of triple;
     stdin, stdout: Text;
     q: array [1..MAX] of longint;

begin
  assign(stdin, 'TRIPLETE.IN');
  reset(stdin);
  assign(stdout, 'TRIPLETE.OUT');
  rewrite(stdout);
  size := 0;
  readln (stdin, tmp.c);

  for i:=1 to MAX do
    q[i] := i*i;

  m:=1;
  while q[m] < tmp.c do
  begin
    n:=1;
    while n <> m do
    begin
      if tmp.c = q[m] + q[n] then
      begin
        tmp.a := q[m] - q[n];
        tmp.b := 2*m*n;
        if gcd(tmp.a, tmp.b) > 1 then
        begin
          n := n+1;
          continue;
        end;
        if tmp.a > tmp.b then
        begin
          aux := tmp.a;
          tmp.a := tmp.b;
          tmp.b := aux;
        end;
        size := size + 1;
        result[size]:=tmp;
      end;
      n := n+1;
```

```
    end;

    m := m+1;
  end;
  if size = 0 then
  begin
    write (stdout, 'NOPE');
  end
  else
  begin
    {sort array}
    for m := 1 to size-1 do
      for n := m+1 to size do
        begin
          if result[m].a > result[n].a then
          begin
            tmp := result[m];
            result[m] := result[n];
            result[n] := tmp;
          end;
        end;
      for i:=1 to size do
        writeln(stdout, result[i].a, ' ',
               result[i].b, ' ', result[i].c);
  end;
  close(stdout);
end.
```

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Submulţimi

Elementele mulţimii $X = \{x_1, x_2, \ldots, x_n\}$ sunt amplasate pe o circumferinţă astfel, încât perechile de elemente x_1, x_2; x_2, x_3; …; x_{n-1}, x_n; x_n, x_1 sunt vecine.

Sarcină. Elaboraţi un program care ar determina numărul submulţimilor diferite de k elemente ale mulţimii X care nu conţin elemente vecine.

Date de intrare. Fişierul text de intrare `SMULTIMI.IN` conţine pe prima linie două numere naturale n şi k, separate prin spaţiu.

Date de ieşire. Fişierul text de ieşire `SMULTIMI.OUT` va conţine pe prima linie numărul m al submulţimilor de k elemente ale mulţimii X care nu conţin elemente vecine.

Restricţii. $1 \leq n, k \leq 444$. Timpul de execuţie a programului nu va depăşi 0,1 secunde. Programul va folosi cel mult 16 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `SMULTIMI.PAS`, `SMULTIMI.C` sau `SMULTIMI.CPP`.

Notă. În condiţiile problemi, numărul m nu va avea mai mult de 93 cifre zecimale.

Exemplul 1

SMULTIMI.IN	SMULTIMI.OUT
1 1	1

Exemplul 2

SMULTIMI.IN	SMULTIMI.OUT
6 2	9

Exemplul 3

SMULTIMI.IN	SMULTIMI.OUT
64 7	296854272

Rezolvare

Vom nota prin *g(n,k)* numărul submulţimilor de *k* elemente ale mulţimii *X* care nu conţin elemente vecine. Se cunoaste că numărul *g*(*n*,*k*) verifică şi egalitatea $g(n,k) = \frac{n}{n-k} C_k^{n-k}$. Este clar ca folosirea acestei formule este dificilă pentru valori mari ale *n* si *k*. De aceea pentru determinarea *g*(*n*,*k*) se va folosi o altă relaţie.

Fie elementele mulţimii $X = \{x_1, x_2, \ldots, x_n\}$ sunt amplasate pe o dreaptă astfel, încât perechile de elemente x_1, x_2; x_2, x_3; …; x_{n-1}, x_n sunt vecine. Vom nota prin *f*(*n*,*k*) numărul submulţimilor de *k* elemente ale mulţimii *X*, amplasate pe dreaptă, care nu conţin elemente vecine. Se poate uşor demonstra veridicitatea urmatoarei afirmatii.

Afirmaţie. Este adevarata urmatoarea egalitate

$$f(n,k) = f(n-1,k) + f(n-2,k-1). \qquad (1)$$

Demonstraţie. Într-adevăr o submulţime de *k* elemente ale mulţimii *X*, care nu conţine elemente vecine şi nici elementul x_1, reprezintă totodată şi o submulţime de *k* elemente ale mulţimii $X\backslash\{x_1\}$ fără elemente vecine. Este adevărată şi afirmaţia reciprocă, uşor de probat şi ea. Deci numărul de astfel de submulţimi este *f*(*n* – 1,*k*). Pe de altă parte, numărul submulţimilor de *k* elemente ale mulţimii *X*, care nu conţin elemente vecine, dar conţin elementul x_1, este egal cu numărul submulţimilor de *k* – 1 elemente ale mulţimii $X\backslash\{x_1, x_2\}$ fără elemente vecine. Şi acest număr este egal cu *f*(*n* – 2,*k* – 1). Corespondenţa biunivocă între submulţimile sus menţionate este firească. Aşadar are loc egalitatea (1). ■

Utilizand aceasta afirmatie, este usor de observat, că *g*(*n*,*k*) = *f*(*n* – 1,*k*) + *f*(*n* – 3,*k* – 1). Mai rămâne să ţinem cont şi de relaţiile *f*(1,1) = 1, *f*(2,1) = 2, *f*(1,*k*) = 0, , *f*(2,*k*) = 0, $k \geq 2$.

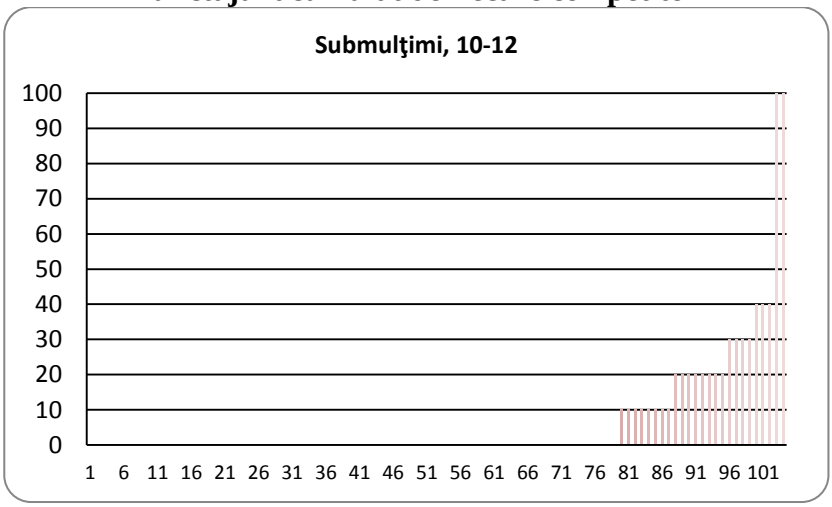
Notă. În conditiile problemei numărul căutat nu va avea mai mult de 93 cifre zecimale. Pentru a utiliza numere atat de mari în program, numărul determinat al submulţimilor se va defini ca variabila de tip `string[93]`. La fel se utilizeaza vectori de lungimea 222, elementele carora sunt de tipul `string[93]`. Afirmatia, demonstrată mai sus ne permite utilizarea variabilelor de tip `string` pentru determinarea numărului submulţimilor în cauză.

```
Program Submultimi;
{clasele 10-12}

const nt = 444;
      kt = nt div 2;
type strb=string[93];
var i,j,l,n,k,code:integer;
    a,b,c:array[1..kt] of strb;
    s:strb;
    f:text;

procedure ssum(s1,s2:strb; var s3:strb);
var code: integer;
    i, j, l, q, r: byte;
    s: string[1];
begin
  if length(s1)<length(s2) then
  begin
    s3:=s2;
    s2:=s1;
    s1:=s3
  end;
    s3 := '';
    q := 0;
  for i:=0 to length(s2)-1 do
    begin
      j:=0;
      val( s1[length(s1)-i], l, code);
        j := j + l;
      val( s2[length(s2)-i], l, code);
        j := j + l + q;
      q := j div 10;
      r:=j mod 10;
        str(r,s);
      s3 := s + s3
    end;
  for i := length(s1) - length(s2) downto 1 do
    begin
      val(s1[i], l, code);
        j := l + q;
      q := j div 10;
        r := j mod 10;
```

```
      str(r,s);
        s3 := s + s3;
      if q=0 then
      begin
        s3 := copy(s1, 1, i-1) + s3;
        break
      end
      end;
      if q=1 then s3 := '1'+s3
end;

begin
  assign(f,'SMULTIMI.IN');
    reset(f);
    read(f,s);
  close(f);

    l:=pos(' ',s);
    val(copy(s,1,l-1),n,code);
    delete(s,1,l);
    val(s,k,code);
    if k=0 then
    begin
      s:='1';
        assign(f,'SMULTIMI.OUT');
        rewrite(f);
        writeln(f,s);
        close(f);
      exit;
    end;
    if k=1 then
    begin
      str(n,s);
        assign(f,'SMULTIMI.OUT');
        rewrite(f);
        writeln(f,s);
        close(f);
      exit;
    end;
    if n<2*k then
    begin
      s:='0';
```

```
        assign(f,'SMULTIMI.OUT');
        rewrite(f);
        writeln(f,s);
        close(f);
      exit;
    end;

    a[1]:='1';
    b[1]:='2';
      for j:=2 to k do
      begin
        a[j]:='0';
        b[j]:='0'
      end;
      l:=n-2*(k-1);
      for i:=3 to l do
      begin
      str(i,c[1]);
          for j:=2 to k do ssum(b[j],a[j-1],c[j]);
      a:=b;
          b:=c;
      end;
      for i:=1 to k-2 do
        for l:=1 to 2 do
        begin
          for j:=i+1 to k do
          ssum(b[j],a[j-1],c[j]);
          a:=b;
          b:=c;
        end;
    ssum(b[k],a[k-1],c[k]);
      ssum(c[k],a[k-1],s);

    assign(f,'SMULTIMI.OUT');
      rewrite(f);
      writeln(f,s);
      close(f);
end.
```

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Focuri de artificii

Organizatorii Jocurilor Olimpice de iarnă de la Sochi din 2014 şi-au propus realizarea unui adevărat spectacol de focuri de artificii. În acest scop se folosesc rachete speciale confecţionate în blocuri. Într-un bloc sunt N rachete. Fiecare rachetă i, $i = 1, 2, 3, \ldots, N$, se caracterizează prin următoarele proprietăţi:

- h_i – înălţimea rachetei faţă de orizont;
- $[a_i; b_i]$ – intervalul de armonie a rachetei;
- f_i – spectaculozitatea (frumuseţea) rachetei lansate.

Orice rachetă a blocului poate fi lansată prima. Rachetă $j < i$ poate fi lansată imediat după cea i a blocului doar dacă $a_j \le h_i \le b_j$; altfel aceasta nu poate fi lansată. Spectaculozitatea rachetelor nelansate este zero. Spectaculozitatea, obţinută la folosirea unui bloc de rachete, se determină ca suma spectaculozităţilor rachetelor lansate ale blocului.

Sarcină. Elaboraţi un program, care ar ajuta organizatorii Jocurilor Olimpice să determine spectaculozitatea maximală a unui bloc de rachete dat.

Date de intrare. Fişierul text de intrare `FOCURI.IN` conţine pe prima linie (linia 0) numărul natural N de rachete în bloc. Fiecare linie i, din următoarele N linii, conţine câte patru numere naturale h_i, a_i, b_i şi f_i, separate prin spaţiu, ce caracterizează racheta i a blocului de rachete dat.

Date de ieşire. Fişierul text de ieşire `FOCURI.OUT` va conţine pe prima linie numărul natural ce caracterizează spectaculozitatea maximală a blocului de rachete dat.

Restricţii. $0 \le N \le 3000$; $0 \le a_i, b_i, h_i \le 1000000$. Timpul de execuţie nu va depăşi 0,5 secunde. Programul va folosi cel mult 32 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `FOCURI.PAS`, `FOCURI.C` sau `FOCURI.CPP`.

Exemplu

FOCURI.IN

```
5
9 0 6 4
5 7 8 1
7 0 6 2
2 1 5 3
9 3 8 2
```

FOCURI.OUT

```
10
```

Rezolvare

Din descrierea condiţiilor de lansare a unei rachete *j* după o rachetă *i*, se pot deduce următoarele:

- ordinea apariţiei rachetelor în fişierul de intrare e semnificativă;
- pentru fiecare rachetă *j*, racheta care va fi lansată după ea apare in fişierul de intrare înainte de ea.

Dacă definim $chainLen_i$ ca fiind "gradul maxim posibil de frumuseţe al unui spectacol care începe cu racheta *i*", atunci observăm că:

$$chainLen_i = 1 + \max_{k=0..i-1} chainLen_k, \text{daca } a_i \le h_k \le b_i .$$

Odată tabelul *chainLen* (cu elementele $chainLen_i$, $i = 1, 2, \ldots, N$) construit aşa, rezultatul final (răspunsul la problemă) va fi elementul maxim din *chainLen*.

```
Program Focuri;
{clasele 10-12}

uses math;
type Racheta = record
    h: longint;
    a: longint;
    b: longint;
    f: longint;
end;

const MAXN = 5000;
```

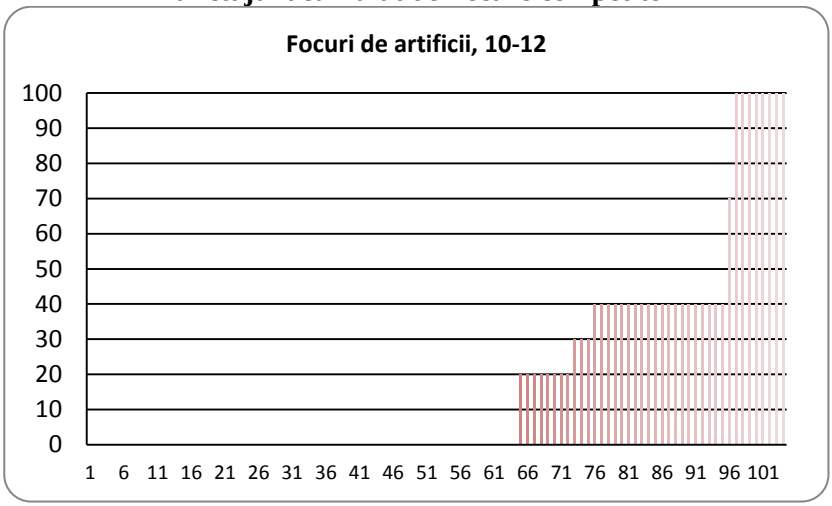
```
var rachete: array[1..MAXN] of Racheta;
    chainLen: array[1..MAXN] of longint;

{calculeaza daca racheta a poate fi lansata dupa
racheta b}
function poateFiLansataDupa(var a: Racheta; var b:
Racheta): boolean;
begin
  poateFiLansataDupa := (a.a <= b.h) and (b.h <=
a.b);
end;

var n, i, j: integer;
f: text;
res, maxLen: longint;
begin
  assign(f, 'FOCURI.IN');
  reset(f);
  {citim datele de intrare}
  readln(f, n);
  for i:=1 to n do
    readln(f, rachete[i].h, rachete[i].a,
rachete[i].b, rachete[i].f);
  close(f);
    {rezolvare}
  for i:=1 to n do
    begin
      maxLen:=0;
    for j:=1 to i-1 do
      begin
        if poateFiLansataDupa(rachete[j],
rachete[i]) and (chainLen[j] > maxLen) then
          begin
            maxLen := chainLen[j];
          end;
      end;
          chainLen[i] := maxLen + rachete[i].f;
    end;
    res := 0;
    for i := 1 to n do
      res := max(res, chainLen[i]);
```

```
      {scriere date iesire}
    assign(f, 'FOCURI.OUT');
      rewrite(f);
      writeln(f, res);
      close(f);
end.
```

Consumul de resurse al algoritmului se deduce în mod direct din descrierea acestuia:

- consumul de timp procesor – pentru calcularea lui $chainLen_i$ se fac i-1 iteratii, deci numărul necesar de operaţii este $1+2+3+...+(n-1)=n(n-1)/2=O(n^2)$ şi se încadrează lejer în restricţia de timp impusă;
- consumul de memorie – se consumă spaţiu pentru a păstra descrierea rachetelor, precum şi tabelul *chainLen*. Deci consumul de memorie este $O(n)$.

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Bacterii

Lena este pasionată de biologie. Efectuând diverse experimente, ea a descoperit o nouă specie de bacterii cu proprietăţi interesante: în primul minut al vieţii sale bacteria nu se reproduce (o vom numi bacterie tânără), iar în fiecare din următoarele minute ea se divide (o vom numi bacterie matură), dând naştere unei bacterii tinere. Fiecare bacterie trăieşte un număr concret de minute, acelaşi pentru toate bacteriile. Lena a hotărât să determine câte bacterii vor fi după o perioadă de timp anumită.

Sarcină. Elaboraţi un program, care i-ar ajuta Lenei să calculeze numărul de bacterii după o perioadă de timp dată.

Date de intrare. Fişierul text de intrare `BACTERII.IN` conţine pe prima linie trei numere naturale p, q şi N, separate prin spaţiu, ce reprezintă numărul de bacterii tinere la începutul experimentului (p), durata vieţii unei bacterii în minute (q) şi durata experimentului în minute (N). Bacterii mature la începutul experimentului nu sunt.

Date de ieşire. Fişierul text de ieşire `BACTERII.OUT` va conţine pe prima linie numărul total de bacterii (tinere şi mature) în populaţie după N minute ale experimentului.

Restricţii. $0 \leq p, q, N \leq 10000$. Timpul de execuţie nu va depăşi 0,75 secunde. Programul va folosi cel mult 128 Megaocteţi de memorie operativă. Fişierul sursă va avea denumirea `BACTERII.PAS`, `BACTERII.C` sau `BACTERII.CPP`.

Exemplul 1

BACTERII.IN	BACTERII.OUT
1 2 2	1

Exemplul 2

BACTERII.IN	BACTERII.OUT
1 3 5	4

Rezolvare

Soluţia problemei se obţine prin construirea vectorului ***B***, elementul b_i al căruia reprezintă numărul de bacterii care au trăit i minute. Atunci, la iteraţia următoare avem:

1. $b_{i+1} = b_i$, $i = q - 1, q - 2, \ldots, 1$.
2. $b_1 = b_2 + b_3 + \ldots + b_q$.

Numerele b_i cresc exponenţial şi de aceea în program se cere realizarea aritmeticii numerelor întregi mari. Realizarea directă a algoritmului are complexitate cubică (calcularea primului element ca suma elementelor în ciclu după timp, realizarea aritmeticii numerelor mari). Dar se poate observa că suma elementelor la fiecare iteraţie se modifică doar prim eliminarea ultimului element (b_q) şi adăugarea unui prim element nou (b_1). De aceea, notând prin S suma

$$S = b_1 + b_2 + b_3 + \ldots + b_q,$$

obţinem următorii doi paşi de efectuat iterativ:

1. $b_{i+1} = b_i$, $i = q - 1, q - 2, \ldots, 1$.
2. $S := S - b_q$.
3. $b_1 = S$.
4. $S := S + b_1$.

În acest caz complexitatea algoritmului este pătratică.

O altă cale de obţinere a soluţiei constă în deducerea formulei generale pentru numărul total $F(N)$ de bacterii după N minute ale experimentului. Un exemplu este prezentat în tabelul următor:

N	b_i/p						$F(N)/p$
	$i = 1$	2	3	4	…	$q - 1$	
1	1	0	0	0	…	0	1
2	1	1	0	0	…	0	2
3	2	1	1	0	…	0	4
4	4	2	1	1	…	0	8
…	…	…	…	…	…	…	…
$q - 1$	2^{q-3}	2^{q-4}	2^{q-5}	2^{q-6}	…	1	2^{q-2}
q	2^{q-2}	2^{q-3}	2^{q-4}	2^{q-5}	…	2	$2^{q-1} - 1$
$q + 1$	$2^{q-1} - 1$	…	…	…	…	4	$2(2^{q-1} - 2)$
$q + 2$	$2(2^{q-1} - 2)$	…	…	…	…	8	$4(2^{q-1} - 3)$
$q + 3$	$4(2^{q-1} - 3)$	…	…	…	…	16	$8(2^{q-1} - 4)$
…	…	…	…	…	…	…	…
$q + k$	$2^{k-1}(2^{q-1} - k)$	…	…	…	…	…	$2^k(2^{q-1} - k - 1)$

În tabel nu sunt prezentate cazurile:
1) $q = 0$, $F(N) = 0$;
2) $q > N = 0$, $F(N) = p$.
Astfel, avem

$$F(N) = \begin{cases} 0, \ q = 0 \\ p, \ q > N = 0 \\ 2^{N-1} p, \ q > N > 0 \\ 2^{N-q}(2^{q-1} - N + q - 1)p, \ N \geq q > 0. \end{cases}$$

În programul Bacterii, codul sursă al căruia urmează, este realizată prima cale de obţinere a soluţiei scontate.

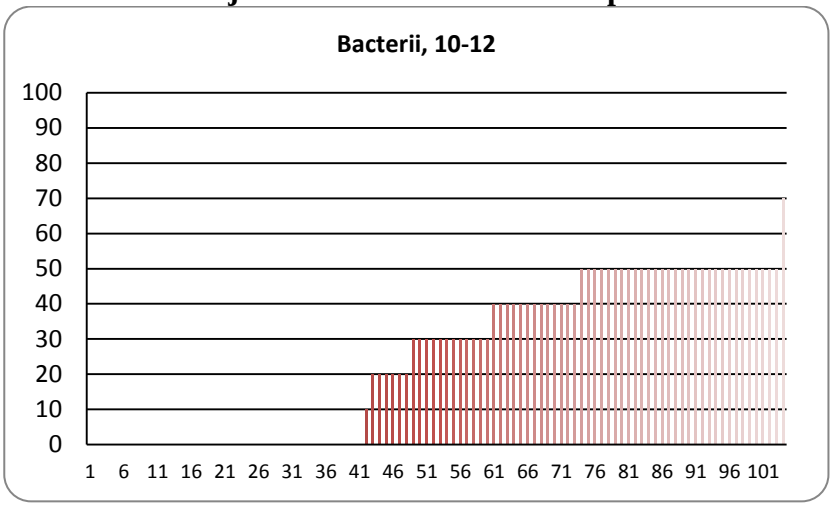
```
Program Bacterii;
{clasele 10-12}

{begin of biginteger code}
const MAX_SIZE = 800;
      DIVIDER = 10000;
      MAX_LT = 10005;
type TDigits = array[1..MAX_SIZE] of integer;
type PDigits = ^TDigits;

type BigInteger = record
    digits: PDigits;
    size: integer;
end;

var max sz:integer;

function Init(num: longint):
BigInteger;
var res: BigInteger;
      i: integer;
begin
  new(res.digits);
  for i := 1 to MAX_SIZE do
    res.digits^[i] := 0;
    res.size := 0;
```

```
  i := num;
  if i = 0 then res.size := 1;
    while i <> 0 do
    begin
      inc(res.size);
      res.digits^[res.size] := i mod DIVIDER;
      i := i div DIVIDER;
    end;
    Init := res;
end;

function Add(a, b: BigInteger):
BigInteger;
var tmp: BigInteger;
      res, i, aux: integer;
begin
  tmp := Init(0);
  res := 0;
  tmp.size := a.size+1;
  if a.size < b.size then
    tmp.size := b.size+1;

  for i := 1 to tmp.size do
  begin
    aux := a.digits^[i] + b.digits^[i] + res;
    tmp.digits^[i] := aux mod DIVIDER;
    res := aux div DIVIDER;
  end;
  if tmp.digits^[tmp.size] = 0 then
    tmp.size := tmp.size - 1;
    if (tmp.size > max_sz) then max_sz := tmp.size;
    if (tmp.size > MAX_SIZE) then
    begin
      writeln('tmp.size = ', tmp.size, ', dying');
      halt(2);
    end;
    Add := tmp;
end;

function Subst(a, b:BigInteger):BigInteger;
var tmp: BigInteger;
      res, i, aux: integer;
```

```
begin
  tmp := Init(0);
  tmp.size := a.size;
  res := 0;

  for i := 1 to tmp.size do
  begin
    aux := a.digits^[i] - b.digits^[i] - res;
    tmp.digits^[i] := (aux + DIVIDER) mod DIVIDER;
    if aux >= 0 then res := 0 else res := 1;
  end;
  Subst := tmp;
end;

procedure Print(var f: Text; a: BigInteger);
var i, k, tmp: integer;
begin
  write(f, a.digits^[a.size]);
  for i := a.size-1 downto 1 do
  begin
    k := DIVIDER;
    tmp := a.digits^[i];
    while k > 1 do
    begin
      k := k div 10;
      write (f, tmp div k);
      tmp := tmp mod k;
    end;
  end
end;

{end of biginteger code}

var accumulator: BigInteger;
      b: array [1..MAX_LT] of BigInteger;
    tmp1, tmp2: BigInteger;
      p, t, lt, i, j: longint;
    stdin, stdout: Text;

begin
  max_sz := -1;
  assign(stdin, 'BACTERII.IN');
```

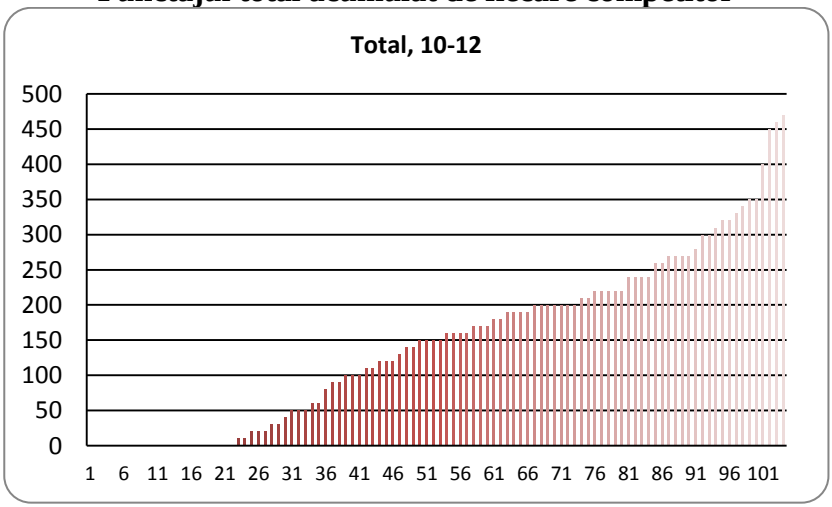
```
  reset(stdin);

  read(stdin, p, lt, t);
  close(stdin);

  assign(stdout, 'BACTERII.OUT');
  rewrite(stdout);

  if t = 0 then
  begin
    writeln(stdout, p);
    close(stdout);
    exit;
  end;
  if lt = 0 then
  begin
    writeln(stdout, 0);
    close(stdout);
    exit;
  end;

  for i:=1 to lt do
    b[i] := Init(0);
    b[1] := Init(p);
    accumulator := b[1];
  for j := 1 to t do
  begin
    tmp1 := b[1];
    tmp2 := b[lt];
    for i := lt downto 2 do
      b[i] := b[i - 1];
    b[1] := Subst(accumulator, tmp1);
    accumulator := Add(accumulator, b[1]);
    accumulator := Subst(accumulator, tmp2);
  end;

  Print(stdout, accumulator);
  close(stdout);
  writeln('max_sz = ', max_sz);
end.
```

Punctajul acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Punctajul total acumulat de fiecare competitor



Notă: Pe orizontală sunt competitorii, simbolizaţi prin numerele 1, 2, 3, ... ş.a.m.d., iar pe verticală – punctajul fiecăruia din ei.

Lista participanţilor

Elevul	Clasa	Instituţia	Localitatea	Raionul/ municipiul
Andrieş Dan	12	Liceul Teoretic "M.Eliade"	Chişinău	Chişinău
Ardeleanu Irina	10	Liceul Teoretic "M.Eminescu"	Anenii Noi	Anenii Noi
Arhiliuc Cristina	12	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Badareu Gabriela	10	Liceul Teoretic "C. Spataru"	Leova	Leova
Bahov Vladimir	11	Liceul Teoretic "M. Eminescu"	Leova	Leova
Balan Denis	11	Liceul Teoretic "I. Vatamanu"	Străşeni	Străşeni
Balta Dorin	12	Liceul Teoretic "Ştefan Vodă"	Ştefan Vodă	Ştefan Vodă
Belibov Dan	11	Liceul Teoretic Varniţa	Varniţa	Anenii Noi
Beriozchin Evghenii	9	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
Bezdrighin Marcel	10	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Bobicov Danil	12	Liceul Teoretic Tvardiţa	Tvardiţa	Taraclia
Borta Alexandru	10	Liceul Teoretic "Olimp"	Costeşti	Ialoveni
Boruţchi Vladimir	10	Liceul Teoretic "A.Puşkin"	Rezina	Rezina
Brădişteanu Elisaveta	11	Liceul Teoretic "M.Eminescu"	Căuşeni	Căuşeni
Buleza Roman	11	Liceul Teoretic "B.P.Hasdeu"	Drochia	Drochia
Burungiu Cristian	12	Liceul Teoretic "M.Eminescu"	Ungheni	Ungheni
Calistru Camelia	10	Liceul Teoretic "L.Damian"	Rîşcani	Rîşcani
Caraiman Olivia	11	Liceul Teoretic "M. Eminescu"	Bălţi	Bălţi
Casap Dumitru	12	Liceul Teoretic "M.Sadoveanu"	Ocniţa	Ocniţa
Catana Vasile	11	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
Cazacu Dana	9	Gimn. Verejeni	Verejeni	Teleneşti
Cepalîga Vladislav	8	Liceul Teoretic nr. 1	Tiraspol	Tiraspol
Cernei Eugeniu	12	Liceul Teoretic "Principesa N. Dadiani"	Chişinău	Chişinău
Cervac Petru	11	Liceul Teoretic "C.Stere"	Soroca	Soroca
Chetruş Antonina	10	Liceul Teoretic "H.Botev"	Valea Perjei	Taraclia
Chihai Mihai	11	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
Ciobanu Grigore	11	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
Ciobanu Lilian	12	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău

Elevul	Clasa	Instituţia	Localitatea	Raionul/ municipiul
Ciornei Florin	9	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Cirimpei Dumitru	12	Liceul Teoretic "M. Eminescu"	Bălţi	Bălţi
Ciuntu Victor	12	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Codiţă Mihai	9	Liceul Teoretic "M. Eminescu"	Leova	Leova
Codreanu Maria	10	Liceul Teoretic Ruseştii Noi	Ruseştii Noi	Ialoveni
Cojocari Valeriu	9	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
Cojocaru Gabriel	7	Liceul Teoretic "I.Vodă"	Cahul	Cahul
Condratiuc Mihai	10	Liceul Teoretic "V.Alecsandri"	Ungheni	Ungheni
Cotos Alexandru	11	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Cotruţă Vlad	11	Liceul Teoretic Criuleni	Criuleni	Criuleni
Covali Cristian	12	Colegiul Financiar-Bancar	Chişinău	Chişinău
Covali Victor	11	Liceul Teoretic "M.Sadoveanu"	Hînceşti	Hînceşti
Cucer Denis	12	Liceul Teoretic "M.Eminescu"	Drochia	Drochia
Cuciuc Iulian	9	Liceul Teoretic Varniţa	Varniţa	Anenii Noi
Diaconu Elena	12	Liceul Teoretic "Olimp"	Costeşti	Ialoveni
Digori Andrei	11	Liceul Teoretic "O. Ghibu"	Orhei	Orhei
Dilan Nicolae	11	Liceul Teoretic "M.Eminescu"	Căuşeni	Căuşeni
Dlujanschi Dan	12	Liceul Teoretic "L.Damian"	Rîşcani	Rîşcani
Dodon Aurel	9	Liceul Teoretic "M. Sadoveanu"	Călăraşi	Călăraşi
Dorogan Sandu	11	Liceul Teoretic "P.Rareş"	Soroca	Soroca
Draguţan Egor	12	Liceul Teoretic "N.Gheorghiu"	Chişinău	Chişinău
Drumea Vasile	10	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
Druţă Gheorghe	12	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Dumbravă George	8	Liceul Teoretic "E. Coşeriu"	Făleşti	Făleşti
Duplachi Cornel	12	Liceul Teoretic "C. Spataru"	Leova	Leova
Feraru Gabriela	10	Liceul Teoretic "M.Sadoveanu"	Hînceşti	Hînceşti
Gaidău Mihai	10	Liceul Teoretic "I. Vatamanu"	Străşeni	Străşeni
Ghileţchii Denis	9	Liceul Teoretic "G. Gaidarji"	Comrat	Comrat
Glingeanu Andrei	12	Liceul Teoretic "M.Eminescu"	Cahul	Cahul

Elevul	Clasa	Instituţia	Localitatea	Raionul/ municipiul
Gnatiuc Denis	11	Liceul Teoretic "I.Creangă"	Floreşti	Floreşti
Gorceac Vladislav	11	Liceul Teoretic "Dm.Cantemir"	Chişinău	Chişinău
Grigoraş Victoria	11	Liceul Teoretic "Ştefan cel Mare"	Rezina	Rezina
Groza Vasile	12	Liceul Teoretic "I. Creangă"	Bălţi	Bălţi
Guidea Marin	11	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
Guţu Dan	10	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
Guzovatîi Anatolii	12	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
Herţa Alina	9	Liceul Teoretic "A.Donici"	Ciuciuleni	Hînceşti
Iusiumbeli Vladislav	9	Liceul Teoretic "S.Baranovski"	Copceac	Ceadîr-Lunga
Ivanenco Valentin	11	Liceul Teoretic nr. 1	Tiraspol	Tiraspol
Jerebţov Dmitrii	10	Liceul Teoretic nr. 1	Tiraspol	Tiraspol
Lazăr Ion	9	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Leviţchi Alexandru	8	Liceul Teoretic "M.Eminescu"	Chişinău	Chişinău
Lobusov Vadim	9	Liceul Teoretic "Ştefan cel Mare"	Bălţi	Bălţi
Lungu Cătălin	10	Liceul Teoretic Criuleni	Criuleni	Criuleni
Macari Ana	10	Liceul Teoretic "Dm. Cantemir"	Mîndreşti	Teleneşti
Malai Victor	8	Liceul Teoretic "M.Eminescu"	Chişinău	Chişinău
Marambei Nicoleta	11	Liceul Teoretic "L.Blaga"	Teleneşti	Teleneşti
Marievschi Alexandru	12	Liceul Teoretic "A.Mateevici"	Pîrliţa	Ungheni
Maşurceac Serghei	9	Liceul Teoretic "I.Creangă"	Floreşti	Floreşti
Mavrodi Dmitrii	10	Liceul Teoretic "G. Gaidarji"	Comrat	Comrat
Maxian Nicu	12	Liceul Teoretic "I.Creangă"	Floreşti	Floreşti
Mihailenco Dmitrii	9	Gimnaziul umanist-matematic	Tiraspol	Tiraspol
Mihalache Andrei	10	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
Mocanu Alexandrina	12	Liceul Teoretic "Dm. Cantemir"	Cantemir	Cantemir
Moglan Mihai	7	Liceul Teoretic "I. Hasdeu"	Chişinău	Chişinău
Morgun Vadim	9	Liceul Teoretic nr. 1	Tiraspol	Tiraspol
Morozov Fiodor	12	Liceul Teoretic "I.Vazov"	Taraclia	Taraclia
Moseev Anastasia	9	Gimn. "M.Sadoveanu"	Cantemir	Cantemir

Elevul	Clasa	Instituţia	Localitatea	Raionul/ municipiul
Moşnoi Ion	12	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
Motroi Valeriu	11	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
Movila Alexandru	11	Liceul Teoretic "M.Viteazul"	Chişinău	Chişinău
Nanii Gheorghe	12	Liceul Teoretic "M. Sadoveanu"	Călăraşi	Călăraşi
Negară Taisia	12	Liceul Teoretic "O. Ghibu"	Orhei	Orhei
Oieneagă Tatiana	11	Liceul Teoretic "M.Eminescu"	Sîngerei	Sîngerei
Ou Andrei	12	Liceul Teoretic Puhoi	Puhoi	Ialoveni
Panfilii Cătălin-Ion	9	Liceul Teoretic Criuleni	Criuleni	Criuleni
Paraschiv Artur	10	Liceul Teoretic "M. Eminescu"	Floreşti	Floreşti
Patraş Cristian	10	Liceul Teoretic "O. Ghibu"	Orhei	Orhei
Pîslari Alexandru	9	Liceul Teoretic "C.Popovici"	Nihoreni	Rîşcani
Plagov Vladimir	11	Liceul Teoretic "I.Vazov"	Taraclia	Taraclia
Plamadeala Dumitru	9	Liceul Teoretic "Ştefan cel Mare"	Taraclia	Căuşeni
Plotnicu Elena	9	Liceul Teoretic "V.Alecsandri"	Sîngerei	Sîngerei
Pupeza Alexandr	11	Liceul Teoretic "N.Gogol"	Chişinău	Chişinău
Raru Dumitru	12	Liceul Teoretic "I.Vodă"	Cahul	Cahul
Rimskii Denis	11	Liceul Teoretic "Dm. Cantemir"	Chişinău	Chişinău
Rozimovschi Denis	10	Liceul Teoretic "Olimp"	Sîngerei	Sîngerei
Russu Vadim	10	Colegiul de Informatică	Chişinău	Chişinău
Russu Vicu	11	Liceul Teoretic "M.Sadoveanu"	Ocniţa	Ocniţa
Savin Vadim	10	Colegiul Financiar-Bancar	Chişinău	Chişinău
Savva Dumitru	11	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
Savva Emilia	7	Liceul Teoretic "I.Creangă"	Chişinău	Chişinău
Schidu Luca	12	Liceul Teoretic Dubăsarii Vechi	Dubăsarii Vechi	Criuleni
Sclifos Eugen	12	Liceul Teoretic Negureni	Negureni	Teleneşti
Scripchin Mihail	9	Liceul Teoretic "I.Creangă"	Cimişlia	Cimişlia
Şenişeuţchi Nicolai	11	Liceul Teoretic "A. Mateevici"	Şoldăneşti	Şoldăneşti
Şeremet Vlad	11	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
Sezghin Erol	10	Liceul Teoretic nr.2, Vulcăneşti	Vulcăneşti	UTAG

Elevul	Clasa	Instituţia	Localitatea	Raionul/ municipiul
Sîrbu Corina	11	Liceul Teoretic "M.Eminescu"	Cimişlia	Cimişlia
Străinu Dragoş	11	Colegiul Financiar-Bancar	Chişinău	Chişinău
Stratulat Ştefan	10	Liceul Teoretic "I. Vatamanu"	Străşeni	Străşeni
Ţelinschi Artiom	9	Gimnaziul Berezovca	Berezovca	Ocniţa
Terzi Andrei	11	Liceul Teoretic "I.Vodă"	Cahul	Cahul
Ţîbuleac Diana	9	Liceul Teoretic Larga	Larga	Briceni
Tidva Vladimir	10	Gimnaziul umanist-matematic	Tiraspol	Tiraspol
Timoftii Victor	12	Liceul Teoretic "M.Eminescu"	Căinari	Căuşeni
Trifan Alexandrina	10	Liceul Teoretic "M. Sadoveanu"	Călăraşi	Călăraşi
Trifan Tamara	11	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
Tuceac Ghenadii	9	Liceul Teoretic Danu	Danu	Glodeni
Umanschii Ianic	8	Liceul Teoretic "M.Eminescu"	Drochia	Drochia
Umanschii Pavel	12	Liceul Teoretic "M.Eminescu"	Drochia	Drochia
Valeanu Valentin	10	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
Vatamaniuc Dan	12	Liceul "Prometeu-Prim"	Chişinău	Chişinău
Vidraşcu Ecaterina	12	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
Vieru Andrei	11	Liceul Teoretic "V.Alecsandri"	Ungheni	Ungheni
Vîrlan Liviu	12	Liceul Teoretic "M.Sadoveanu"	Hînceşti	Hînceşti
Vovcenko Ivan	12	Liceul Teoretic "Dm.Cantemir"	Chişinău	Chişinău
Vrabie Victor	11	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
Vrabii Valeriu	12	Liceul Teoretic Visoca	Visoca	Soroca

Lista premianţilor

Locul	Elevul	Clasa	Profesorul	Instituţia	Localitatea	Municipiul/ raionul
1	Cojocari Valeriu	9	Corlat Sergiu	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
1	Ciuntu Victor	12	Adam Ecaterina	Liceul "Prometeu-Prim"	Chişinău	Chişinău
1	Savva Dumitru	11	Corlat Sergiu	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
1	Bezdrighin Marcel	10	Dragan Galina	Liceul "Prometeu-Prim"	Chişinău	Chişinău
2	Guzovatîi Anatolii	12	Miron Raisa	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
2	Lazăr Ion	9	Dragan Galina	Liceul "Prometeu-Prim"	Chişinău	Chişinău
2	Morgun Vadim	9	Şagoian Tamara	Liceul Teoretic nr. 1	Tiraspol	Tiraspol
2	Umanschii Pavel	12	Chistruga Gheorghe	Liceul Teoretic "M.Eminescu"	Drochia	Drochia
2	Vrabie Victor	11	Brodicico Valeriu	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
2	Şeremet Vlad	11	Corlat Sergiu	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
2	Russu Vadim	10	Iordăchiţă E.	Colegiul de Informatică	Chişinău	Chişinău
2	Jerebţov Dmitrii	10	Armaş Marina	Liceul Teoretic nr. 1	Tiraspol	Tiraspol
2	Valeanu Valentin	10	Brodicico Valeriu	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
3	Ciornei Florin	9	Dragan Galina	Liceul "Prometeu-Prim"	Chişinău	Chişinău
3	Umanschii Ianic	8	Chistruga Gheorghe	Liceul Teoretic "M.Eminescu"	Drochia	Drochia
3	Beriozchin Evghenii	9	Corlat Sergiu	Liceul Teoretic "Orizont", Durleşti	Chişinău	Chişinău
3	Moşnoi Ion	12	Miron Raisa	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
3	Ciobanu Lilian	12	Miron Raisa	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
3	Marievschi Alexandru	12	Ciupercă Romeo	Liceul Teoretic "A.Mateevici"	s.Pîrliţa	Ungheni

Locul	Elevul	Clasa	Profesorul	Instituţia	Localitatea	Municipiul/ raionul
3	Ciobanu Grigore	11	Brodicico Valeriu	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
3	Ivanenco Valentin	11	Şagoian Tamara	Liceul Teoretic nr. 1	Tiraspol	Tiraspol
3	Pupeza Alexandr	11	Panocec Tatiana	Liceul Teoretic "N.Gogol"	Chişinău	Chişinău
3	Mihalache Andrei	10	Miron Raisa	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
3	Savin Vadim	10	Gîncu Silviu	Colegiul Financiar-Bancar	Chişinău	Chişinău
3	Gaidău Mihai	10	Ţîrdea Lidia	Liceul Teoretic "I. Vatamanu"	Străşeni	Străşeni
m	Plamadeala Dumitru	9	Traci Pelaghia	Liceul Teoretic "Ştefan cel Mare"	s. Taraclia	Căuşeni
m	Iusiumbeli Vladislav	9	Dragan Nicolai	Liceul Teoretic "S.Baranovski"	s.Copceac	Ceadîr-Lunga
m	Moglan Mihai	7	Ţurcan Ludmila	Liceul Teoretic "I. Hasdeu"	Chişinău	Chişinău
m	Covali Cristian	12	Gîncu Silviu	Colegiul Financiar-Bancar	Chişinău	Chişinău
m	Cirimpei Dumitru	12	Curbet Gheorghe	Liceul Teoretic "M. Eminescu"	Bălţi	Bălţi
m	Vidraşcu Ecaterina	12	Brodicico Valeriu	Liceul Teoretic "B.Cazacu"	Nisporeni	Nisporeni
m	Balan Denis	11	Gaidău Maria	Liceul Teoretic "I. Vatamanu"	Străşeni	Străşeni
m	Belibov Dan	11	Druc Tatiana	Liceul Teoretic Varniţa	s.Varniţa	Anenii Noi
m	Cotos Alexandru	11	Zaim Sofia	Liceul "Prometeu-Prim"	Chişinău	Chişinău
m	Trifan Tamara	11	Miron Raisa	Liceul Academiei de Ştiinţe a Moldovei	Chişinău	Chişinău
m	Stratulat Ştefan	10	Ţîrdea Lidia	Liceul Teoretic "I. Vatamanu"	Străşeni	Străşeni
m	Trifan Alexandrina	10	Gavriliţă Alexei	Liceul Teoretic "M. Sadoveanu"	Călăraşi	Călăraşi

Premianţii Olimpiadei Republicane la Informatică, Ediţia 2014