Craiova, 26-28 martie 2004

Concursul de programare ȘTEFAN ODOBLEJA

În perioada 26-28 martie 2004 a avut loc la Craiova ediția a treia a Concursului Interjudețean de Programare "Ștefan Odobleja". Datorită unor probleme organizatorice concursul pe echipe nu a mai avut loc, concurenții având posibilitatea să-și arate cunostințele de algoritmică și să-și dovedească abilitățile de programatori în cadrul concursului individual.

Clasa a IX-a

P040519: Loto

Văzând *Aurel* cum toată lumea se îmbulzește să joace la loto 6 din 49, se hotărăște să își încerce și el norocul. Un vecin îi explică regulile acestui joc. Există un bilet care trebuie completat. Pe el se află un careu cu numere de la 1 la 49, deci careul conține 49 de pătrățele numerotate. Dacă el marchează 6 numere, se consideră că a ales o variantă; aceasta costă 30 000 de lei. Jocul constă în extragerea a 6 numere. Dacă toate numerele extrase se regăsesc printre cele marcate de el în careu, atunci biletul este câștigător.

Dacă el marchează M numere ($M > 6$), se consideră că a ales toate variantele de câte 6 numere dintre cele M completate. Deci, dacă va marca 7 numere, va avea 7 variante posibile, fiecare variantă reprezentând câte 6 numere din cele 7.

De exemplu, dacă *Aurel* marchează primele numere de la 1 la 7, următoarele variante se obțin:

```
1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 6 7
1 2 3 5 6 7
1 2 4 5 6 7
1 3 4 5 6 7
2 3 4 5 6 7
```

Dacă va marca 8 numere, va avea 28 de variante de câte 6 numere care se pot crea. Costul unui bilet este egal cu numărul de variante care se pot forma înmulțite cu costul unei variante.

Aurel are o sumă de N lei. Ajutați-l să cumpere bilete de loto astfel: pe primul bilet să marcheze cât mai multe numere, dar costul biletului să nu depășească suma de N lei. Din suma de bani eventual rămasă trebuie să cumpăre alte bilete, respectând același principiu.

Date de intrare

Fișierul **LOTO.IN** conține un singur număr natural N care reprezintă suma de bani pe care vrea *Aurel* să o joace la loto.

Date de ieșire

Fișierul de ieșire **LOTO.OUT** trebuie să conțină pe fiecare linie câte două numere.

Primul număr de pe o linie reprezintă numărul de numere marcate M_1 pe un bilet, iar al doilea număr reprezintă numărul de bilete cu M_1 numere marcate. Următoarea linie, dacă există, are numărul M_2 ($M_2 < M_1$) și numărul de bilete în care se pot marca cele M_2 numere ș.a.m.d.

Restricție

- $N \leq 15\,000\,001$.

Exemplu

LOTO.IN

```
1300000
```

LOTO.OUT

```
8 1
7 2
6 1
```

Explicație

Cu suma de 1.300.000 lei, *Aurel* poate cumpăra un bilet cu 8 numere marcate, două bilete cu 7 numere marcate și un bilet cu 6 numere marcate.

Timp de execuție: 1 secundă/test

P040520: Premii

Compania *C* dorește să premieze o parte dintre clienții săi la sfârșitul zilei. Numărul premiilor fiind mic, se stabilește următoarea strategie:

- toți clienții se vor înscrie pe o listă, fiecare primind un număr de ordine; primul client va primi numărul de ordine 2, al doilea client va primi numărul de ordine 3, al treilea client va primi numărul de ordine 4 ș.a.m.d.;
- premiile se vor acorda astfel: primul client, cu numărul de ordine 2, este declarat câștigător, iar clienții cu numerele de ordine 4, 6, 8, 10 etc. sunt necâștigători și sunt excluși din listă; următorul client din listă, cel cu numărul de ordine 3, este declarat câștigător, iar clienții cu numerele de ordine 9, 15, 21, 27 etc. sunt necâștigători și sunt excluși din listă; următorul client din listă, cel cu numărul de ordine 5, este declarat câștigător, iar clienții cu numerele de ordine 19, 35, 49, 65 etc. sunt necâștigători și sunt excluși din listă;
- procedeul continuă până când nu mai rămâne nici un client în listă.

Clienții câștigători vor fi cei cu numerele de ordine: 2, 3, 5, 7, 11, 13, 17 etc.

Dându-se un număr *I*, să se determine numărul de ordine al celui de al *I*-lea client câștigător.

Date de intrare

Fișierul de intrare **C.IN** conține pe prima linie un număr întreg pozitiv *N* care reprezintă numărul de linii care se află în continuare. Fiecare dintre următoarele *N* linii, va conține un număr intreg *I* care reprezintă un câștigător.

Date de ieșire

Fișierul de ieșire **C.OUT** trebuie să conțină *N* linii. Fiecare linie a fișierului de ieșire va trebui să conțină un singur număr care reprezintă numărul de ordine al unui câștigător corespunzător unei linii din fișierul de intrare.

Datele din fișierul de ieșire trebuie scrise în ordinea celor din fișierul de intrare.

Restricție

- $1 \le I \le 3000$.

Exemplu

C.IN

```
5
1
2
3
20
73
```

C.OUT

```
2
3
5
83
437
```

Timp de execuție: 1 secundă/test

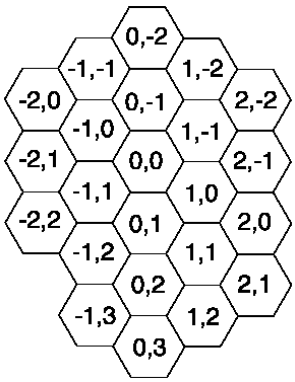
Clasa a X-a

P040521: Albinuța Zora

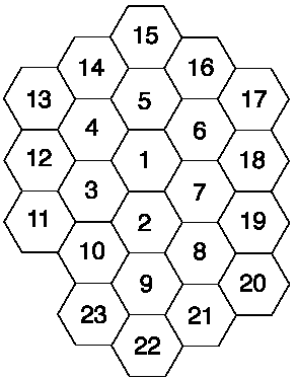
Zora este o albinuță. Ea trăiește într-un stup împreuna cu alte albine. Acest stup este alcătuit din mai multe celule hexagonale în care se stochează miere.

Zora are o problemă. *Somnorilă* i-a spus unde poate să-l întâlnească, dar, pentru că *Somnorilă* este un bondar, iar *Zora* este o albină lucratoare, ei utilizează sisteme de coordonate diferite.

Zora care, de regulă, zboară direct către o celulă anume, și-a dezvoltat un sistem de coordonate bidimensional peste întregul stup, iar celula din mijlocul stupului are coordonatele (0, 0).



Somnorilă, mult mai leneș, este obișnuit să meargă de jur împrejur, numărând celulele în sensul acelor de ceasornic, începând cu 1 în mijlocul stupului.



Ajutați-o pe *Zora* sa obțină conversia din sistemul lui *Somnorilă* în sistemul ei.

Date de intrare

Fișierul de intrare **ZORA.IN** conține unul sau mai multe numere întregi reprezentând numerele lui *Somnorilă*. Fiecare număr este situat pe câte o linie și este urmat de marcajul de sfârșit de linie (<ENTER>).

Ultima linie din fișier conține valoarea 0. Nu trebuie produsă nici o ieșire pentru 0.

Date de ieșire

Fișierul de ieșire **ZORA.OUT** trebuie să conțină coordonatele sistemului lui *Zora* corespunzătoare numerelor lui *Somnorilă*. Fiecare pereche de coordonate se va scrie pe câte o linie

și este urmată de marcajul de sfârșit de linie (<ENTER>), iar cele două coordonate corespunzătoare unei perechi trebuie separate între ele printr-un spațiu.

Restricție
• numărul de celule din stup este mai mic decât 100000.

Exemplu
ZORA.IN
```
1
2
3
4
5
0
```

ZORA.OUT
```
0 0
0 1
-1 1
-1 0
0 -1
```

Timpi de execuție: 1 secundă/test

P040522: Dreptunghiuri
Pe ecranul calculatorului sunt dispuse dreptunghiuri care au laturile paralele cu axele de coordonate (marginile ecranului).

Sarcina voastră este să scrieți un program care acceptă la intrare astfel de dreptunghiuri, date prin coordonatele pe ecran a două vârfuri opuse. Programul va trebui să determine perimetrul și aria reuniunii de dreptunghiuri de pe ecran.

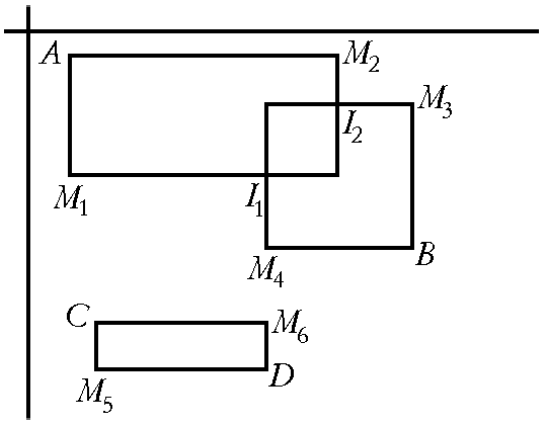


Figura 1: Exemplu de dreptunghiuri. Reuniunea lor este: $(A, M_1, I_1, M_4, B, M_3, I_2, M_2) \cup (C, M_5, D, M_6)$

Date de intrare
Fișierul de intrare **DREPT.IN** va conține pe fiecare linie coordonatele de pe ecran (numere întregi) a două vârfuri opuse ale unui dreptunghi cu laturile paralele cu axele de coordonate.

Dacă două vârfuri opuse ale unui dreptungi sunt $M_1(x_1, y_1)$ și $M_2(x_2, y_2)$, atunci o linie din fișierul de intrare va conține numerele x_1, y_1, x_2 și y_2, separate între ele printr-un singur spațiu.

Date de ieșire
Fișierul de ieșire **DREPT.OUT** va conține pe prima linie perimetrul, iar pe cea de-a doua linie aria reuniunii de dreptunghiuri.

Restricții și precizări
• numărul maxim de dreptunghiuri din fișierul de intrare este 200;
• în fișierul de intrare nu există două dreptunghiuri suprapuse în totalitate.

Exemplu

DREPT.IN	DREPT.OUT
0 20 100 100	680
60 0 120 90	21300
190 10 110 110	
150 50 20 120	

Timpi de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P040523: Rumbu
Gigel se duce în țara *Rumbu*, undeva pe planeta *Ajelbodo*, și nu cunoaște limba băștinașilor.

După mai multe peripeții reușește să fure un papirus în care sunt descrise regulile de formare a cuvintelor din limba *rumbu*. Cuvintele în limba *rumbu* sunt formate numai din literele mici ale alfabetului latin Σ = {'*a*', ..., '*z*'}.

Un cuvânt se construiește în urma aplicării unor reguli de forma: $M_1 \to M_2M_3$ sau $M_4 \to \alpha$, unde M_1, M_2, M_3 și M_4 sunt metasimboluri oarecare (nu aparțin alfabetului Σ al limbii), iar $\alpha \in \Sigma$. Mulțimea metasimbolurilor este formată din litere mari din mulțimea {'A', ..., 'Z'}.

Orice cuvânt aparținând limbii este obținut pornind de la o regulă care are în partea stângă metasimbolul *S*. Un cuvânt este considerat corect dacă:
• s-a obținut prin aplicarea repetată a regulilor;
• nu conține nici un metasimbol.

Dacă pe papirus se află următoarele reguli:
R1) $S \to AB$
R2) $A \to a$
R3) $B \to b$
R4) $A \to AA$
atunci cuvântul '*aaab*' este corect deoarece se poate obține astfel:

$$S \xrightarrow{R_1} AB \xrightarrow{R_4} AAB \xrightarrow{R_2} aAB \xrightarrow{R_4}$$
$$aAAB \xrightarrow{R_3} aAAb \xrightarrow{R_2} aaAb \xrightarrow{R_2} aaab$$

Ajutați-l pe *Gigel* să dezlege misterele limbii *rumbu.*

Dându-se un set de reguli de pe un papirus și mai multe șiruri de caractere, verificați dacă acestea aparțin limbajului *rumbu*.

Date de intrare

Pe prima linie a fișierului de intrare **PROB1.IN** se află un număr natural *N* care reprezintă numărul regulilor din mulțimea *P* (de pe papirus). Pe următoarele *N* linii se află regulile, câte una pe linie. O regulă poate avea forma:

`A->BC` sau `A->a`.

Ultima linie a fișierului de intrare va conține un cuvânt, format numai din simboluri din Σ, pentru care trebuie determinată apartenența la limbajul rumbu.

Date de ieșire

Fișierul de ieșire **PROB1.OUT** va conține pe o linie răspunsul `DA` în cazul în care cuvântul din fișierul de intrare aparține limbajului *rumbu* și `NU` în caz contrar.

Restricție

- $N \leq 200$;
- numărul de caractere ale unui cuvânt pentru care trebuie determinată apartenența la limbajul *rumbu* este cel mult egal cu 250.

Exemplu

PROB1.IN

```
2
S->SS
S->s
ssss
```

PROB1.OUT

```
DA
```

Timpi de execuție: 3 secunde/test

P040524: Multiprocesor

O companie a achiziționat un calculator multiprocesor care dispune de *N* procesoare. Aceste procesoare sunt așezate liniar și sunt etichetate cu numere cuprinse între 1 și *N*. Ele sunt utilizate pentru a procesa sarcini independente. Fiecare sarcină nou sosită este distribuită, în mod arbitrar, unuia dintre cele *N* procesoare. Uneori este posibil ca un procesor să aibă atașate prea multe sarcini, iar alte procesoare să aibă prea puține (posibil chiar nici una). În această situație calculatorul realizează o realocare a sarcinilor procesoarelor pentru a îmbunătăți performanțele.

Procesul de realocare se realizează în runde. În fiecare rundă, fiecare procesor poate transfera cel mult o sarcină fiecăruia dintre procesoarele vecine lui. Vecinii unui procesor *i* sunt procesoarele *i* - 1 și *i* + 1 (procesoarele 1 și *N* au numai câte un vecin, pe 2, respectiv pe *N* - 1). Scopul acestei realocări este ca toate procesoarele să aibă de executat același număr de sarcini.

Dându-se un număr de procesoare și numărul de sarcini distribuite fiecărui procesor se cere să se determine numărul minim de runde necesar pentru a se ajunge la starea în care fiecare procesor să aibă de executat același număr de sarcini.

Date de intrare

Fișierul de intrare **PROB2.IN** conține mai multe seturi de date. Fiecare set începe cu o linie care conține un singur număr *N* - numărul de procesoare. În continuare, avem *N* numere separate prin spații și/sau caractere de sfârșit de linie, reprezentând numărul de sarcini distribuite fiecărui procesor. Seturile de date sunt separate printr-o linie vidă. Ultimul set de date este urmat de numărul -1 pe o singură linie (număr care nu va fi procesat).

Date de ieșire

Fișierul de ieșire **PROB2.OUT** va conține numărul minim de runde necesare pentru a echilibra sistemul pentru fiecare set în parte. Dacă pentru un set nu este posibil ca fiecare procesor să aiba același număr de sarcini de executat se va afișa ca rezultat -1.

Restricții și precizări

- $1 \leq N, M \leq 10000$;
- numărul de sarcini atașate fiecărui procesor este cuprins între 0 și 32767, iar numărul total de runde nu va depăși un *longint*.

Exemplu

PROB2.IN

```
2
49 50

3
1 10 1

3
0 66 3

8
16 16 15 0 20 2 1 2

10
0 0 200 0 0 0 0 0 0 0

-1
```

PROB2.OUT

```
-1
3
23
22
140
```

Timpi de execuție: 1 secundă/test