Focșani, 18-24 aprilie 2003

Olimpiada NAȚIONALĂ de informatică 2003

Vă prezentăm în continuare enunțurile celor 36 de probleme propuse spre rezolvare la ediția din acest an a Olimpiadei Naționale de Informatică. Articolul cuprinde problemele pentru gimnaziu, cele pentru liceu, precum și cele pentru selectarea membrilor lotului național.

Clasa a V-a

P050301: Săritura cangurului

A fost odată ca niciodată un cangur care creștea, ca *Făt-Frumos* din poveste, într-un an precum alții în zece. Într-o zi a început să facă sărituri. Și a sărit pentru început șapte metri. A doua zi a sărit, în plus față de ziua precedentă, de zece ori mai mult. În a treia zi a reușit să sară, în plus față de prima zi, de zece ori mai mult decât în ziua a doua. În a patra zi a sărit, în plus față de prima zi, de zece ori mai mult decât în ziua a treia. Și așa mai departe.

Scrieți un program care calculează câți metri a sărit cangurul, în total, în n zile.

Date de intrare
Se citește de la tastatură valoarea n.

Date de ieșire
Se va afișa pe ecran câți metri a sărit cangurul, în total, după cele n zile.

Restricție
- n este un număr natural, strict mai mic decât 12.

Exemplu
Pentru n = 3 se va afișa: 861 metri.

P050302: Numere prime

Se numește număr prim, un număr care este divizibil doar cu 1 și cu el însuși. Astfel în intervalul [1, 30] numerele prime vor fi: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 (în total sunt zece numere prime).

Dându-se două numere n și k să se determine $2 \cdot k$ numere prime situate în centrul listei numerelor prime din intervalul [1, n], în cazul în care intervalul conține un număr par de numere prime, respectiv $2 \cdot k - 1$ numere din centrul listei de numere prime, în cazul în care numărul de numere prime din interval este impar.

Dacă numărul $2 \cdot k$ (respectiv $2 \cdot k - 1$) este mai mare decât numărul de numere prime din intervalul considerat, atunci se vor afișa toate numerele prime din interval.

Date de intrare
De la tastatură se vor citi două numere n și k:
- n reprezintă marginea superioară a intervalului din care se determină numerele prime;
- k are semnificația din enunț.

Date de ieșire
Se vor afișa pe ecran numerele cerute, separate prin spațiu.

Restricții și precizări
- $1 \le n \le 10000$;
- $1 \le k \le 30$;
- $k \le n$;
- 1 nu se consideră a fi număr prim.

Exemple

Date de intrare	*Date de ieșire*
21 2	5 7 11 13
18 2	5 7 11
18 18	2 3 5 7 11 13 17
100 7	17 19 23 29 31 37 41 43 47 53 59 61 67

P050303: Poarta Orintiei

Copa bate la poarta *Orintiei*, dar poarta este programată să nu se deschidă decât după ce se introduc, într-o casetă cu s spații ($3 \le s \le 10$), s cifre strigate de portar. Portarul a strigat: "1", *Copa* a butonat 1, în primul spațiu de la stânga la dreapta. Portarul a strigat: "0", și, în timp ce *Copa* butona 0 în spațiul al doilea, 1 a devenit 2 în spațiul anterior. Portarul a strigat: "7". *Copa* scria 7 în spațiul al treilea, iar în primul spațiu, 2 devine 3, iar în al doilea spațiu, 0 devine 1. Și tot așa, până la al s-lea spațiu, când *Copa* reușește să scrie toate cifrele și apare tot codul. Și poarta se deschide,

dar… surpriză, mai era o poartă, iar codul acesteia, *N*, era cel mai mic număr format din cât mai multe dintre cifrele codului anterior, astfel încât nici o cifră să nu se repete.

Disperat de atâta informatizare, *Copa*, umil cetățean al *Orintiei* vă cere sprijinul să calculați cel de-al doilea cod *N*.

Date de intrare
De la tastatură se introduc numărul *s* de spații ale casetei și cele *s* cifre pe care le strigă portarul.

Date de ieșire
Se vor afișa pe ecran cifrele codului *N* neseparate prin spații.

Precizare
În cazul în care într-un spațiu al casetei se află cifra 9, după ce *Copa* va introduce următoarea cifră, cifra 9 va deveni 0.

Exemplu

Date de intrare	*Date de ieșire*
10 1 0 7 9 7 3 6 9 4 6	102456789

Timp de execuție: 1 secundă/test

Clasa a VI-a

P050304: Ora de sport

Profesorul de sport al clasei a *VI*-a *B* de la o școală din *Focșani* vrea la începutul orei să așeze elevii pe terenul de sport, la raport, într-o anumită ordine. Pentru acest lucru, elevii sunt bine instruiți, astfel încât, așezând pe ultimul rând *n* elevi, celelalte rânduri de elevi să se creeze singure după regula că pe poziția *i* a unui rând se va așeza un elev, după cum urmează: dacă pe rândul din spate, pe pozițiile *i* și *i* + 1 stau fie numai băieți, fie numai fete, atunci se va așeza o fată, iar dacă pe aceste poziții stau elevi de sexe diferite, se va așeza un băiat.

Conform acestei reguli, pe rândul cu numărul de ordine *i* ($i \in \{1, 2, \ldots, n\}$) se vor așeza *i* elevi.

Numărul de elevi din clasă este $n \cdot (n + 1) / 2$.

Pentru *n* dat și un șir de *n* numere `0` și `1` (`0` reprezintă codificarea pentru o fată, iar `1` pentru un băiat), care reprezintă șirul de elevi de pe ultimul rând, se cere să se determine numărul de băieți din clasă.

Date de intrare
De la tastatură se citesc datele de pe două linii:
- de pe prima linie se citește numărul *n*;
- de pe linia a doua se citește un șir de *n* numere `0` sau `1`, separate printr-un spațiu care reprezintă șirul de elevi de pe ultimul rând.

Date de ieșire
Pe ecran se va afișa numărul de băieți din clasă.

Restricție
- $1 \le n \le 20$.

Exemplu

Date de intrare	*Date de ieșire*
5 1 0 0 1 1	8

Explicație
Pentru exemplu avem așezarea:

```
1   0   0   1   1    ultimul rând
  1   0   1   0      rândul 4
    1   1   1        rândul 3
      0   0          rândul 2
        0            rândul 1
```

Timp de execuție: 1 secundă/test

P050305: Număr

Gigel are de rezolvat următoarea problemă: se consideră numărul natural *N* format din maxim nouă cifre, distincte două câte două și în care nu există cifra `0`. *Gigel* va trebui să facă bilețele pentru fiecare cifră din număr, bilețele pe care le va pune într-o căciulă, conform următorului algoritm: inițial pornește de la ultima cifră a numărului (cifra unităților) și pune în căciulă bilețelul pe care este scrisă această cifră.

Dacă aceasta este o valoare pară, începe parcurgerea numărului spre dreapta, în caz contrar spre stânga, parcurgerea făcându-se cu un număr de pași egal cu cifra respectivă.

În timpul parcurgerii unui număr spre dreapta se consideră că după ultima cifră urmează prima (cea mai semnificativă cifră a numărului), după aceasta urmează a doua, etc., iar în deplasarea spre stânga după prima cifră (cea mai semnificativă cifră a numărului) urmează ultima cifră (cifra unităților), apoi penultima, etc., iar parcurgerea începe cu cifra din număr imediat de lângă cifra scrisă pe ultimul bilețel introdus în căciulă, respectând sensul parcurgerii.

De exemplu, dacă numărul nostru este `1346`, *Gigel* pornește de la cifra `6`, iar biletul pe care s-a scris această cifră îl pune în căciulă. El parcurge numărul spre dreapta, efectuând șase pași; el trece de cifrele: `1`, `3`, `4`, `6`, `1` și se oprește la cifra `3`. Deci, în căciulă va pune bilețelul pe care este scrisă cifra `3`.

Algoritmul continuă până când se termină toate bilețelele sau când ajunge la o cifră pentru care bilețelul cu valoarea respectivă a fost introdus deja în căciulă.

În cazul în care algoritmul se încheie întrucât *Gigel* a pus toate biletelele în căciulă, se va afișa cifra de pe ultimul bilețel introdus în căciulă, iar în cazul în care *Gigel* ajunge în timpul parcurgerii la o cifră pentru care bilețelul corespunzător a fost introdus deja în căciulă, se va afișa valoarea acestei cifre.

Date de intrare
Se va citi de la tastatură numărul natural *N* format din cel mult nouă cifre distincte între ele și care nu conține cifra 0.

Date de ieșire
Se va afișa pe ecran cifra la care *Gigel* a ajuns în momentul opririi algoritmului.

Restricție
• $0 < N < 1.000.000.000$.

Exemple
Pentru *N* = `412`, *Gigel* începe cu cifra 2 (bilețelul cu cifra `2` este pus de *Gigel* în căciulă); fiind valoare pară, parcurge spre dreapta și se oprește la cifra `1`, bilețelul cu această cifră fiind pus în căciulă. Cifra `1` fiind impară, continuă parcurgerea spre stânga și se oprește la cifra `4` și pune astfel și ultimul bilețel în căciulă. În acest moment nu mai există bilețele și se va afișa cifra `4`.

Pentru *N* = `1243`, *Gigel* începe cu cifra `3` (bilețelul cu cifra `3` este pus de *Gigel* în căciulă); fiind valoare impară, parcurge spre stânga și se oprește la cifra `1`, bilețelul cu această cifră fiind pus în căciulă. Cifra `1` fiind impară, continuă parcurgerea spre stânga și se oprește tot la cifra `3`, dar nu mai există bilețelul cu cifra `3` pentru a putea fi pus în căciulă. Se va afișa deci, cifra `3`.

Timp de execuție: 1 secundă/test

P050306: "Reorganizarea" numerelor

Ministerul numerelor are de câteva zile un nou șef. Acesta a dorit să facă o serie de schimbări în ministerul pe care îl conduce și a început "reorganizarea" cu mulțimea numerelor naturale în două etape: mai întâi toate numerele naturale au fost așezate fără spațiu (sau alt separator) între ele. După această primă etapă, mulțimea numerelor naturale arăta astfel:
`12345678910111213141516171819202122232425262 7...`

A doua etapă a "reorganizării" a constat în formarea unor noi "grupe": o grupă de o cifră, o grupă de două cifre, o grupă de trei cifre și așa mai departe. Astfel, "grupele reorganizate" sunt: `1`, `23`, `456`, `7891`, `01112`, `131415`, `1617181`, `92021222`, `324252627`...

Pentru un număr natural *N* dat, să se afișeze prima și ultima cifră din cea de-a *N*-a grupă de cifre obținută după "reorganizare", valori separate printr-un spațiu.

Date de intrare
De la tastatură se citește valoarea numărului natural *N*.

Date de ieșire
Pe ecran se va afișa prima și ultima cifră din cea de-a *N*-a grupă de cifre obținută după "reorganizare", valori separate printr-un spațiu.

Restricție
• $1 \le n \le 250$.

Exemplu
Pentru *N* = `8` se va afișa: `9 2` (deoarece `9` și `2` sunt prima, respectiv ultima cifră din grupa a opta care este `92021222`)

Timp de execuție: 1 secundă/test

Clasa a VII-a

P050307: Portocal

Oaza *Lacotrop* din deșertul *Etpas* este înconjurată de *n* portocali care conțin toți fructe, dispuși sub formă de cerc și numerotați de la 1 la *n*, în sensul acelor de ceasornic. Maimuța *Gino* pornește de la un portocal *m* și numără întotdeauna, în sensul acelor de ceasornic, *k* portocali care conțin fructe. Culege toate fructele din portocalul de pe poziția *k*. Continuă numărătoarea începând cu portocalul următor celui din care a cules, dar care conține fructe. În final rămâne un singur portocal *p* necules, în care *Gino* își face adăpost.

Cu ce portocal *m* trebuie să înceapă *Gino* numărătoarea pentru ca acesta să-și facă adăpostul exact în portocalul *p*?

Date de intrare
Fișierul de intrare **`portocal.in`** conține pe o singură linie, numerele *n*, *k* și *p*, separate printr-un singur spațiu.

Date de ieșire
Fișierul **`portocal.out`** conține pe prima linie numărul natural *m*, reprezentând portocalul cu care *Gino* începe numărătoarea.

Restricții
• $2 \le n \le 1.000$;
• $1 \le k \le 10.000$;
• $1 \le p \le 1.000$.

Exemplu

`portocal.in`	**`portocal.out`**
`6 8 5`	`3`

Timp de execuție: 1 secundă/test

P050308: Debarcare

În toiul pregătirilor debarcării din *Normandia* (din al doilea Război Mondial) ofițerii de contrainformații germani au observat că prin punctele de frontieră au început să circule fel de fel de indivizi care au pe centură gravate litere și cifre. După ce au reușit să prindă câțiva dintre ei și le-au

confiscat centurile au putut constata că întotdeauna pe centură se află un număr n de litere și cifre. După îndelungi "interviuri" au aflat că pe centură sunt codificate sub formă numerică, în baza 16, liniile și modurile de atac. Pentru a decodifica mesajul centura era tăiată în $\sqrt{n}$ bucăți care erau așezate una sub alta, după care se citeau caracterele de pe fiecare coloană, de sus în jos, iar numărul format pe o coloană era transformat în baza 10. Dacă numărul rezultat ar avea cifrele în ordine strict crescătoare atunci va ataca mai întâi infanteria, dacă era strict descrescător atacă prima aviația, altfel va fi un atac combinat (mixt). Numărul de linii de atac este egal cu $\sqrt{n}$.

Scrieți un program care, citind informațiile de pe o centură, să determine numărul x al liniilor de atac și modul în care se va desfășura atacul.

Date de intrare
De pe prima linie a fișierului **mesaj.in** se citește mesajul. Literele din mesaj vor fi doar litere mari.

Date de ieșire
Pe prima linie a fișierului **mesaj.out** se va scrie numărul x, iar pe următoarele x rânduri câte unul dintre cuvintele `infanterie`, `aviatie`, `mixt` în funcție de tipul de atac.

Precizare
- n este pătrat perfect, strict mai mic ca 100. Numerele formate în baza 10 au cel puțin două cifre. Literele care pot apărea sunt *A*, *B*, *C*, *D*, *E*, *F*.

Exemplu

mesaj.in	**mesaj.out**
01C7A8BAA	3 infanterie mixt aviatie

Explicație
Dacă tăiem centura și punem bucățile una sub alta obținem:

```
01C
7A8
BAA.
```

Așadar, numerele în baza 16 vor fi: `07B`, `1AA`, `C8A` a căror valori în baza 10 vor fi : `123`, `426`, `3210`.

Timp de execuție: 1 secundă/test

P050309: Domino
Indiferent de codificarea datelor necesare pentru memorarea coordonatei, N piese de domino de diferite înălțimi sunt așezate în dreptul unor coordonate, marcate de-a lungul unei linii drepte trasată pe o suprafață plană.

Acționând asupra unei piese, aceasta poate antrena în cădere la stânga sau la dreapta, un șir de alte piese, piese care vor cădea unele peste altele. O piesă este doborâtă de o altă piesă dacă este atinsă în cădere de aceasta, chiar și numai la baza ei, fiind astfel posibil ca și o piesă mai mică să doboare o alta mai mare, dacă cea mare este atinsă de cea mică în cădere.

Determinați lungimea maximă a unui șir de piese căzute care se obține prin acționarea unei piese într-un anumit sens. Aceasta este cea mai mare lungime pe orizontală acoperită integral de piese căzute.

Date de intrare
Fișierul de intrare **domino.in** conține pe prima linie N, numărul de piese, iar pe fiecare din următoarele N linii, coordonata la care se află piesa și înălțimea acesteia, despărțite printr-un spațiu.

Date de ieșire
Fișierul de ieșire **domino.out** va conține pe prima linie un număr natural reprezentând lungimea maximă a șirului de piese căzute.

Restricții și precizări
- suprafața este suficient de mare pentru ca piesele să nu cadă în afara ei;
- numărul de piese N este un număr natural din intervalul [1, 100];
- înălțimea unei piese este un număr natural din intervalul [1, 1000].

Exemple

domino.in
```
5
40 1
8 2
30 4
1 6
4 11
```

domino.out
```
14
```

domino.in
```
6
53 3
62 4
15 5
20 6
60 8
75 9
```

domino.out
```
12
```

Explicație
Pentru primul exemplu, lungimea celui mai lung șir de piese căzute este egală cu 14 și se obține prin acționarea spre dreapta a piesei așezate la coordonata 1, care va antrena în cădere piesa de la coordonata 4, iar aceasta la rândul ei pe cea de la coordonata 8.

Pentru cel de-al doilea exemplu, lungimea celui mai lung șir de piese căzute este egală cu 12 și se obține prin acționarea spre stânga a piesei așezate la coordonata 62, care va antrena în cădere piesa de la coordonata 60, iar aceasta la rândul ei pe cea de la coordonata 53.

Timp de execuție: 1 secundă/test

Clasa a VIII-a

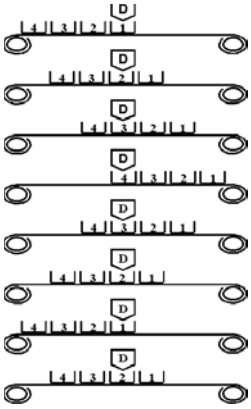
P050310: Cutii de bomboane

Pe o bandă care desfășoară o mișcare "du-te-vino" se află n cutii, inițial goale. Un dispozitiv fix suspendat deasupra benzii eliberează câte o bomboană din când în când plasând-o în cutia aflată în momentul respectiv în dreptul său.

Banda se deplasează constant astfel încât la fiecare secundă se află o altă cutie în dreptul dispozitivului D (aceasta este cutia vecină celei care s-a aflat anterior în dreptul acestuia).

Dacă $n = 4$, atunci inițial cutia având numărul 1 se află în dreptul dispozitivului, în secunda imediat următoare banda se va deplasa astfel încât cutia 2 se va afla în dreptul dispozitivului. În secundele următoare (3, 4, 5, 6, 7, 8 etc.) cutiile 3, 4, 3, 2, 1, 2 etc. se vor afla succesiv în dreptul dispozitivului.

Lungimea totală a benzii este de $2 \cdot n - 1$ ori mai mare decât lungimea unei cutii, astfel încât în timpul mișcării, va exista în permanență o cutie sub dispozitivul care eliberează bomboane.



În figura alăturată sunt reprezentate primele opt secunde de funcționare a ansamblului format din banda rulantă cu patru cutii și dispozitivul D. Se știe că, în prima secundă de funcționare a ansamblului, dispozitivul eliberează o bomboană în cutia numărul 1.

Cunoscându-se numărul n al cutiilor, durata de timp t dintre două eliberări succesive de bomboane și numărul de bomboane b eliberate în total de dispozitiv, să se determine numărul de cutii care rămân goale și numărul maxim de bomboane existente într-o cutie la sfârșitul procesului.

De exemplu, dacă $n = 6$, $t = 4$ și $b = 10$, atunci, deoarece în secundele 1, 5, 9, 13, 17, 21, 25, 29, 33, 37 dispozitivul eliberează câte o bomboană în cutiile 1, 5, 3, 3, 5, 1, 5, 3, 3 și respectiv 5, înseamnă că au rămas trei cutii goale (cutiile 2, 4 și 6) și numărul maxim de bomboane dintr-o cutie este 4.

Date de intrare

Fișierul **cutii.in** conține pe o singură linie numerele n, t și b separate prin spațiu.

Date de ieșire

Fișierul **cutii.out** va trebui să conțină pe o linie numerele c și m separate printr-un singur spațiu.

Restricții

- $1 < n < 1.000$;
- $0 < t < 1.000.000$;
- $0 < b < 1.000.000.000$.

Exemplu

cutii.in	cutii.out
4 17 5	0 2

Timp de execuție: 1 secundă/test

P050311: Matrice monoton maximală

O matrice *monotonă* este o matrice care, dacă este citită pe linii de la stânga spre dreapta sau pe coloane de sus în jos, valorile parcurse sunt crescătoare.

O submatrice este o regiune dreptunghiulară dintr-o matrice cu proprietatea că este formată din linii și coloane consecutive.

Se cere să se determine o submatrice maximală monotonă dintr-o matrice dată. Dacă există mai multe astfel de submatrice, va trebui determinată doar una dintre ele.

Date de intrare

Fișierul de intrare **matrice.in** conține pe prima linie numerele n și m, separate printr-un singur spațiu, care reprezintă numărul de linii, respectiv coloane ale matricei.

Pe fiecare a i-a linie din următoarele n linii ale fișierului se află m numere, separate printr-un singur spațiu, care reprezintă a i-a linie din matrice.

Date de ieșire

Fișierul de ieșire **matrice.out** va trebui să conțină pe prima linie două numere k și p, separate printr-un singur spațiu, care reprezintă numărul de linii și coloane ale matricei monoton maximale determinate.

Pe fiecare dintre următoarele k linii se vor afla p numere separate printr-un singur spațiu reprezentând o linie din matricea determinată.

Restricții

- $1 \le n, m \le 100$;
- elementele matricei sunt numere întregi cuprinse între 0 și 99.

Exemple

matrice.in	matrice.out
4 5 2 4 4 8 8 1 4 5 10 9 7 9 8 13 17 10 11 14 15 16	4 2 4 8 510 8 13 1415

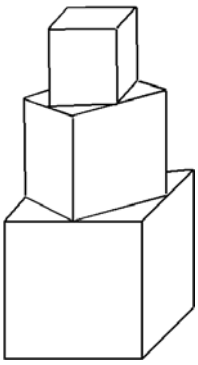
matrice.in	matrice.out
5 6 2 0 5 4 8 7 1 2 4 6 8 14 0 4 7 8 10 12 4 8 8 10 13 15 6 6 10 12 11 16	3 4 2 4 6 8 4 7 8 10 8 8 10 13

Timp de execuție: 2 secunde/test

P050312: Cuburi

Fie un turn format din n cuburi: primul (cel de jos) are latura L. Fiecare cub care se așează peste altul are latura mai mică decât acesta, astfel încât este poziționat cu vârfurile bazei sale exact în mijloacele laturilor bazei superioare a cubului precedent.

În figura alăturată este prezentat un turn format din trei cuburi care respectă condițiile precizate.



Dându-se numerele L, n cu semnificația descrisă mai sus și h un număr real strict pozitiv, se cere să se calculeze:
- volumul corpului obținut;
- aria totală a corpului astfel obținut (inclusiv baza inferioară a cubului de jos și cea superioară a cubului de sus);
- înălțimea turnului;
- numărul minim de cuburi care formează un turn de înălțime cel puțin h, plecând de la cubul de latură L în prima poziție (cea de jos).

Date de intrare

Fișierul de intrare **cuburi.in** conține o singură linie pe care se află numerele L, n și h separate printr-un spațiu.

Date de ieșire

Fișierul **cuburi.out** va conține pe prima linie volumul obținut, pe cea de-a doua linie aria totală a corpului, pe cea de-a treia linie înălțimea turnului și pe cea de-a patra linie numărul de cuburi necesare pentru a atinge înălțimea h sau `-1` dacă nu sunt suficiente `30.000` de cuburi.

Restricții și precizări

- L este număr real strict pozitiv;
- n este număr natural cuprins între 1 și 10.000;
- h este număr real strict pozitiv;
- calculele se vor efectua cu numere reale, iar afișarea se va face cu 5 zecimale exacte.

Exemple

cuburi.in
```
10.00 2 20
```

cuburi.out
```
1353.55339
800.00000
17.07107
3
```

cuburi.in
```
10.00 3 40
```

cuburi.out
```
1478.55339
900.00000
22.07107
-1
```

Timp de execuție: 1 secundă/test

Clasa a IX-a

P050313: Seti

Cercetătorii care lucrează la programul *SETI* au recepționat două transmisii de date foarte ciudate, date care ar putea veni din partea unor civilizații extraterestre. Primul set de date este format din zece caractere distincte, date în ordinea lor lexicografică, ce formează alfabetul extraterestru. A doua transmisie conține cuvinte din exact patru caractere.

Cercetătorii trebuie să ordoneze lexicografic cuvintele primite în a doua transmisie (conform alfabetului extraterestru).

Date de intrare

Fișierul de intrare **seti.in** conține pe prima linie cele zece caractere ale alfabetului, iar pe fiecare din următoarele linii câte un cuvânt.

Date de ieșire

Fișierul de ieșire **seti.out** va conține cuvintele ordonate, câte unul pe linie.

Restricții și precizări

- în fișierul de intrare nu sunt mai mult de 200.000 de cuvinte, iar caracterele sunt literele mici ale alfabetului englez;
- datele de intrare se presupun a fi corecte.

Exemplu

seti.in
```
abcdefghij
aaaa
fgaa
aabc
iihf
```

seti.out
```
aaaa
aabc
fgaa
iihf
```

Timp de execuție: 1 secundă/test

P050314: Circular

Unele numere naturale sunt formate doar din cifre distincte nenule. Dintre acestea, unele, numite ***numere circulare***, au următoarea proprietate: pornind de la prima cifră și numărând spre dreapta, după cifră, atâtea cifre cât indică aceasta, se determină o nouă cifră. Procedând la fel și pentru aceasta și pentru toate cele care urmează se va ajunge din nou la prima cifră. Dacă toate cifrele au fost vizitate exact o dată, numărul se numește circular. De exemplu numărul `1894256` este număr circular deoarece:
- are numai cifre distincte;
- nu conține cifra `0`;
- pornind de la `1` obținem, pe rând: `8`, `9`, `2`, `6`, `5`, `4`, `1`.

Scrieți un program care, pentru un N dat, determină câte numere circulare sunt mai mici sau egale cu N, precum și cel mai mare număr circular mai mic sau egal cu N.

Date de intrare

Pe prima linie a fișierului de intrare **circular.in** se află numărul natural N.

Date de ieșire

Fișierul de ieșire **circular.out** conține o singură linie, pe care se află numărul de numere circulare mai mici ca N,

precum și numărul circular maxim cerut, separate printr-un spațiu. Dacă nu există nici un număr circular mai mic ca N, în fișierul de ieșire se vor afișa două valori `0` separate printr-un spațiu.

Restricție
- $10 \le N < 10000000$.

Exemple

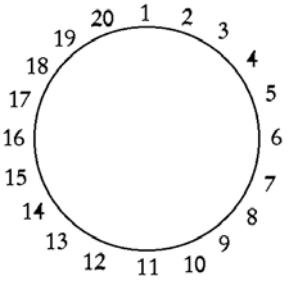
circular.in	**circular.out**
1894250	347 1849625

Explicație
Există `347` numere circulare mai mici decât `1894250`, cel mai mare dintre acestea fiind `1849625`.

Timp de execuție: 1 secundă/test

P050315: Scaune

Se consideră `ns` scaune numerotate de la `1` la `ns`, aranjate în cerc. Ca exemplu, pentru `ns` = 20 așezarea scaunelor este dată în figura alăturată.



Pe fiecare din aceste scaune este așezat un copil. Primul copil stă pe scaunul `1`, iar ultimul pe scaunul `ns`.

Pe lângă cele `ns` scaune deja ocupate, alți `nc` copii ($1 \le nc \le ns$) așteaptă să se elibereze un loc.

La un moment dat un singur copil se ridică de pe scaun și pleacă. Atunci, cât timp în dreptul scaunului liber nu există un copil, toți copiii aflați în așteptare se mișcă în sens invers mișcării acelor ceasornicului, câte o poziție, până când unul din ei ocupă locul liber.

Sunt respectate următoarele condiții:
- la început toate scaunele sunt ocupate;
- fiecare copil aflat în așteptare se află inițial în dreptul unui scaun ocupat;
- când un copil avansează cu n poziții spre un loc pe scaun, toți cei care așteaptă avansează tot cu n poziții; datorită faptului că mișcarea este circulară, avansarea cu `4` poziții de la poziția `18`, semnifică o deplasare în dreptul poziției `2`.

Se dă o secvență a numerelor de ordine a copiilor care pleacă la fiecare moment; să se scrie un program care afișează numărul scaunului pe care s-a așezat fiecare copil care așteaptă, dacă acest lucru este posibil.

Date de intrare
Pe prima linie a fișierului de intrare **`scaune.in`** se află două numere, separate prin spațiu, reprezentând numărul de scaune (`ns`) și numărul copiilor care stau în așteptare (`nc`). Pe următoarele `nc` linii vor fi date pozițiile copiilor aflați în așteptare. În continuare, până la sfârșitul fișierului, sunt linii care descriu numerele de ordine ale copiilor care se ridică unul câte unul de pe scaune și părăsesc jocul.

Date de ieșire
Fișierul de ieșire **`scaune.out`** va conține `nc` linii, fiecare linie conținând poziția inițială de așteptare a copilului și poziția ocupată, separate printr-un spațiu. Liniile de ieșire trebuie să fie în aceeași ordine ca cele din fișierul de intrare. În cazul în care un copil nu are nici o posibilitate să se așeze, în dreptul său se va scrie `0` în fișierul de ieșire.

Restricții
- $1 \le ns \le 200$;
- $1 \le nc \le ns$.

Exemplu

scaune.in	**scaune.out**
20 5 6 19 17 13 1 1 3 20 16	6 16 19 3 17 0 13 20 1 1

Timp de execuție: 1 secundă/test

P050316: Criptare

Mircea și *Vasilică* vor să-și trimită mesaje pe care alții să nu le înțeleagă. Au citit ei despre spioni și modalități de a scrie mesaje și, în final, au imaginat un mod de criptare a unui mesaj care folosește *cuvânt cheie* (le-a plăcut lor denumirea asta).

Alegându-și un cuvânt cheie format numai din litere distincte, ei numără literele acestuia și împart mesajul în grupe de lungime egală cu numărul de litere ale cuvântului cheie și le așează una sub alta. Desigur, se poate întâmpla ca ultima grupă să fie incompletă, așa că o completează cu spații. Apoi numerotează literele cuvântului cheie în ordinea apariției lor în alfabetul englez. În final, rescriu mesajul astfel: coloana de sub litera numerotată cu `1`, urmată de coloana de sub litera numerotată cu `2`, etc. De asemenea, spațiile sunt înlocuite de caracterul `'*'`.

De exemplu, fie cuvântul cheie `criptam` și mesajul care trebuie criptat `Incercam sa lucram cu coduri si criptari.` Cuvântul cheie are șapte litere numerotate cu `2`, `6`, `3`, `5`, `7`, `1`, respectiv `4`, deoarece aceasta este ordinea apriției caracterelor cuvântului în alfabetul englez. În continuare se împarte mesajul în grupuri de câte șapte litere fiecare:

```
Incerca|m sa lu|cram cu| coduri| si cri|ptari.
```

Codificăm acum mesajul:

`2635714`
```
Incerca
m*sa*lu
cram*cu
*coduri
*si*cri
ptari.*
```

După codificare mesajul devine:
`clcrr.Imc**pcsaoiaauuii*eamd*rn*rcstr**uci.`

Fiind date un cuvânt cheie și un mesaj criptat, decodificați mesajul trimis de *Mircea* pentru *Vasilică*.

Date de intrare
Fișierul de intrare **criptare.in** conține pe prima linie mesajul criptat, iar pe linia a doua cuvântul cheie.

Date de ieșire
Fișierul de intrare **criptare.out** conține pe prima linie mesajul decriptat.

Restricții și precizări
- lungimea mesajului este de minim 20 și maxim 1000 caractere;
- cuvântul cheie are minim 5 și maxim 20 de caractere;
- cuvântul cheie conține numai litere mici ale alfabetului.

Exemplu
criptare.in
```
clcrr.Imc**pcsaoiaauuii*eamd*rn*rcstr**uci
criptam
```

criptare.out
```
Incercam sa lucram cu coduri si criptari.
```

Timp de execuție: 1 secundă/test

P050317: Mașină
O țară are $3 \le N \le 30\,000$ orașe, numerotate de la 1 la N, dispuse pe un cerc. *PAM* tocmai și-a luat carnet de conducere și vrea să viziteze toate orașele țării. Lui *PAM* îi este frică să conducă prin locuri aglomerate, așa că ea și-a propus să meargă numai pe șoselele unde traficul este mai redus. Există șosele de legătură între oricare două orașe alăturate: între orașul 1 și orașul 2, … , între orașul i și orașul $i + 1$, iar orașul N este legat de orașul 1. Ca să nu se rătăcească, *PAM* și-a propus să-și aleagă un oraș de început și să meargă pe șoselele respective în sens trigonometric până ajunge înapoi în orașul de unde a plecat. Dacă *PAM* pleacă din orașul K, atunci traseul ei va fi: $K, K + 1, ..., N, 1, 2, ..., K$.

Mașina lui *PAM* are un rezervor foarte mare (în care poate pune oricât de multă benzină).

În fiecare oraș, *PAM* ia toată cantitatea de benzină existentă în oraș, iar parcurgerea fiecărei șosele necesită o anumită cantitate de benzină.

Știind că *PAM* are, la începutul călătoriei, doar benzina existentă în orașul de plecare, și că, atunci când ajunge într-un oraș, ea va lua toată cantitatea de benzină disponibilă în acel oraș, să se găsească un oraș din care *PAM* își poate începe excursia, astfel încât să nu rămână fără benzină.

Se consideră că *PAM* a rămas fără benzină dacă în momentul plecării dintr-un oraș, nu are suficientă benzină pentru a parcurge șoseaua care duce la orașul următor. Dacă benzina îi ajunge "la fix" (adică la plecare are tot atâta benzină câtă îi trebuie) se consideră că *PAM* poate ajunge până în orașul următor.

Date de intrare
Fișierul de intrare **masina.in** conține pe prima linie numărul N. Pe cea de-a doua linie se găsesc N numere naturale $a_1, a_2, \ldots, a_N$, separate prin câte un spațiu, unde a_i reprezintă cantitatea de benzină disponibilă în orașul i. Linia a treia conține un șir de N numere naturale $b_1, b_2, \ldots, b_N$, separate prin câte un spațiu, unde b_i reprezintă cantitatea de benzină necesară străbaterii șoselei dintre orașele i și $i + 1$ (sau N și 1, dacă $i = N$).

Date de ieșire
Fișierul de ieșire **masina.out** va conține un singur număr s care reprezintă un oraș din care, dacă *PAM* își începe călătoria, poate completa turul țării fără a face pana prostului.

Restricții și precizări
- dacă există mai multe soluții, se va determina una singură;
- $0 \le a_i \le 30.000$;
- $1 \le b_i \le 30.000$;
- $a_1 + a_2 + \ldots + a_N \le 2.000.000.000$.

Exemplu
masina.in
```
6
0 3 2 5 10 5
7 8 3 2 1 4
```

masina.out
```
4
```

Timp de execuție: 0,3 secunde/test

P050318: Operații
Notăm cu c și r câtul și respectiv restul împărțirii unui număr nr la 2^k, unde k este un număr natural nenul.

Asupra numărului putem efectua succesiv una dintre următoarele operații:
- $O_1(nr, k)$ reprezintă transformarea numărului nr în numărul $2^k \cdot (2 \cdot c + 1) + r$ pentru orice rest r;
- $O_2(nr, k)$ reprezintă transformarea numărului nr în numărul $2^{k-1} \cdot c + r$ doar dacă $r < 2^{k-1}$,

Se consideră două numere naturale nenule m și n. Efectuați asupra numerelor m și n operații succesive, O_1 sau O_2, pentru valori alese ale lui k, astfel încât după un număr finit de operații cele două numere să devină egale, iar valoarea astfel obținută să fie minimă.

Date de intrare
Fișierul de intrare **operatii.in** conține pe o singură linie două numere naturale nenule m și n, separate printr-un spațiu, reprezentând cele două numere date.

Date de ieșire
Fișierul de ieșire **operatii.out** va conține:
- pe prima linie numărul natural nenul *nmin*, reprezentând valoarea minimă obținută din m și n prin aplicarea unor succesiuni de operații O_1 sau O_2;

- pe a doua linie numărul *i* al operațiilor efectuate asupra numărului *m*;
- pe fiecare dintre următoarele *i* linii o pereche formată din două numere reprezentând operația (`1` sau `2`) și respectiv valoarea lui *k* pentru operația respectivă, separate printr-un spațiu;
- pe linia *i* + 2 numărul *j* al operațiilor efectuate asupra numărului *n*;
- pe fiecare dintre următoarele *j* linii o pereche formată din două numere reprezentând operația (`1` sau `2`) și respectiv valoarea lui *k* pentru operația respectivă, separate printr-un spațiu.

Restricții

- $1 \le m, n \le 2.000.000.000$.

Exemplu

`operatii.in`
```
11 45
```

`operatii.out`
```
15
2
2 3
1 2
2
2 2
2 4
```

Timp de execuție: 1 secundă/test

Clasa a X-a

P050319: Asediu

Se aproximează o zonă de război sub forma unui cerc pe circumferința căruia se stabilesc *N* puncte, reprezentând comandamentele trupelor aliate, cu proprietatea că nu există trei corzi cu capetele în aceste *N* puncte care să fie concurente într-un punct situat în interiorul cercului. Între oricare două puncte (comandamente) există un drum sigur de acces direct. Aceste drumuri împreună cu circumferința cercului delimitează un număr de regiuni distincte. Există informații că regiunile astfel delimitate reprezintă de fapt terenuri minate de combatanții inamici. Fiecare astfel de regiune va fi cercetată amănunțit de câte un soldat aliat echipat corespunzător cu detectoare de mine.

Indicați numărul de soldați de care este nevoie pentru cercetarea tuturor regiunilor formate.

Date de intrare

Fișierul de intrare **`asediu.in`** conține pe prima linie numărul *N* al comandamentelor.

Date de ieșire

Pe prima linie a fișierului de ieșire **`asediu.out`** se va afla numărul *T* de soldați necesari pentru cercetarea tuturor regiunilor formate.

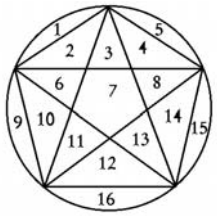
Restricție

- $2 \le N \le 2.000.000$.

Exemplu

`asediu.in`
```
5
```

`asediu.out`
```
16
```



Timp de execuție: 1 secundă/test

P050320: Muzeu

Sunteți un participant la *Olimpiada Națională de Informatică*. În programul olimpiadei intră și câteva activități de divertisment. Una dintre ele este vizitarea unui muzeu. Acesta are o structură de matrice dreptunghiulară cu *M* linii și *N* coloane.

Din orice cameră se poate ajunge în camerele vecine pe direcțiile nord, est, sud și vest (dacă aceste camere există). Pentru poziția (*i*, *j*) deplasarea spre nord presupune trecerea în poziția (*i* - 1, *j*), spre est în (*i*, *j* + 1), spre sud în (*i* + 1, *j*) și spre vest în (*i*, *j* - 1).

Acest muzeu are câteva reguli speciale. Fiecare cameră este marcată cu un număr între `0` și `10` inclusiv. Mai multe camere pot fi marcate cu același număr. Camerele marcate cu numărul `0` pot fi vizitate gratuit. Într-o cameră marcată cu numărul *i* (*i* > `0`) se poate intra gratuit, dar nu se poate ieși din ea decât dacă arătați supraveghetorului un bilet cu numărul *i*.

Din fericire, orice cameră cu numărul *i* (*i* > `0`) oferă spre vânzare un bilet cu numărul *i*; odată cumpărat acest bilet, el este valabil în toate camerele marcate cu numărul respectiv. Biletele pot avea prețuri diferite, dar un bilet cu numărul *i* va avea același preț în toate camerele în care este oferit spre vânzare.

Dumneavoastră intrați în muzeu prin colțul de nord-vest (poziția (1, 1) a matricei) și doriți să ajungeți la ieșirea situată în colțul de sud-est (poziția (*M*, *N*) a matricei). Odată ajuns acolo, primiți un bilet gratuit care vă permite să vizitați tot muzeul.

Pozițiile (1, 1) și (*M*, *N*) sunt marcate întotdeauna cu numărul `0`.

Cunoscându-se structura muzeului, determinați o strategie de parcurgere a camerelor, astfel încât să ajungeți în camera (*M*, *N*) plătind cât mai puțin. Dacă există mai multe variante, alegeți una în care este parcurs un număr minim de camere (pentru a câștiga timp și pentru a avea mai mult timp pentru vizitarea integrală a muzeului).

Date de intrare

Prima linie a fișierului de intrare **`muzeu.in`** conține două numere întregi *M* și *N* separate printr-un spațiu, numărul de linii, respectiv de coloane, al matricei care reprezintă muzeul. Următoarele *M* linii conțin structura muzeului; fiecare conține *N* numere întregi între `0` și `10` inclusiv separate prin spații. A (*M* + 2) - a linie conține `10` numere întregi între `0` și `10000` inclusiv, reprezentând costurile biletelor `1`, `2`, `3`, ..., `10` în această ordine.

Date de ieșire
În fișierul **muzeu.out** veți scrie:
- pe prima linie suma minimă necesară pentru a ajunge din poziția (1, 1) în poziția (M, N);
- pe a doua linie numărul minim de mutări L efectuate dintr-o cameră într-o cameră vecină, pentru a ajunge din poziția (1, 1) în poziția (M, N);
- pe a treia linie L caractere din multimea {N, E, S, V}, reprezentând deplasări spre nord, est, sud sau vest.

Restricție
- $2 \le N, M \le 50$.

Exemplu

muzeu.in	muzeu.out
5 6 0 0 0 0 0 2 0 1 1 1 4 3 0 1 0 0 0 0 0 1 5 1 0 0 0 0 0 1 0 0	12 9 EEEEESSSS

Timp de execuție: 1 secundă/test

P050321: Munte

Într-o zonă montană se dorește deschiderea unui lanț de telecabine. Stațiile de telecabine pot fi înființate pe oricare dintre cele N vârfuri ale zonei montane. Vârfurile sunt date în ordine de la stânga la dreapta și numerotate de la 1 la N, fiecare vârf i fiind precizat prin coordonata X_i pe axa Ox și prin înălțimea H_i.

Se vor înființa exact K stații de telecabine. Stația de telecabine i ($2 \le i \le K$) va fi conectată cu stațiile i - 1 și i + 1; stația 1 va fi conectată doar cu stația 2, iar stația K, doar cu stația K - 1. Stația 1 va fi obligatoriu amplasată în vârful 1, iar stația K în vârful N.

Se dorește ca lanțul de telecabine să asigure legătura între vârful 1 și vârful N. Mai mult, se dorește ca lungimea totală a cablurilor folosite pentru conectare să fie minimă. Lungimea cablului folosit pentru a conecta două stații este egală cu distanța dintre ele. În plus, un cablu care unește două stații consecutive nu poate avea lungimea mai mare decât o lungime fixată L.

O restricție suplimentară este introdusă de formele de relief. Astfel, vârfurile i și j ($i < j$) nu pot fi conectate direct dacă există un vârf v ($i < v < j$), astfel încât segmentul de dreapta care ar uni vârfurile i și j nu ar trece pe deasupra vârfului v. În cazul în care cele trei vârfuri sunt coliniare, se consideră toate trei ca fiind stații, chiar dacă distanța dintre vârfurile i și j este mai mică decât L.

Dându-se amplasarea celor N vârfuri ale lanțului muntos, stabiliți o modalitate de dispunere a celor K stații de telecabine, astfel încât lungimea totală a cablurilor folosite pentru conectare să fie minimă, cu restricțiile de mai sus.

Se garantează că întotdeauna conectarea va fi posibilă.

Date de intrare
Prima linie a fișierului de intrare **munte.in** conține trei numere întregi N, K și L separate prin spații, cu semnificațiile de mai sus. Următoarele N linii conțin coordonatele vârfurilor; linia i + 1 conține coordonatele vârfului i, X_i și H_i separate printr-un spațiu.

Date de ieșire
În fișierul **munte.out** veți scrie:
- pe prima linie lungimea totală minimă a cablurilor, rotunjită la cel mai apropiat numar întreg (pentru orice întreg Q, Q + 0.5 se rotunjeste la Q + 1);
- pe a doua linie K numere distincte cuprinse între 1 și N, ordonate crescător, numerele vârfurilor în care se vor înființa stații de telecabine; dacă există mai multe variante, trebuie determinată doar una dintre ele.

Restricții și precizări
- $2 \le N \le 100$;
- $2 \le K \le 30$ și $K \le N$;
- $0 \le L, X_i, H_i \le 100.000$ și $X_i < X_{i+1}$.

Exemplu

munte.in	munte.out
7 5 11 0 16 4 3 6 8 7 4 12 16 13 16 14 16	22 1 3 5 6 7

Observații
Trasarea unui cablu direct între vârfurile 1 și 5 ar fi contravenit restricției referitoare la lungimea maximă a unui cablu; în plus, s-ar fi obținut o soluție cu două stații de telecabine în loc de trei (deci soluția ar fi invalidă și pentru valori mari ale lui L);

Pentru a ilustra restricția introdusă de formele de relief, precizăm că vârfurile 1 și 4 nu au putut fi conectate direct datorită înălțimii vârfului 3. De asemenea, vârfurile 5 și 7 nu au putut fi conectate direct datorită înălțimii vârfului 6.

Timp de execuție: 1 secundă/test

P050322: Partiție

Se definește o partiție a unui număr natural n ca fiind o scriere a lui n sub forma:

$$n = n_1 + n_2 + \ldots + n_k,$$

unde $k \ge 1$ și $n_1, n_2, \ldots, n_k$ sunt numere naturale care verifică relația: $n_1 \ge n_2 \ge \ldots \ge n_i \ge \ldots \ge n_k \ge 1$.

Fiind dat un număr natural n, să se determine câte partiții ale lui se pot scrie, conform cerințelor de mai sus, știind că oricare număr n_i dintr-o partiție trebuie să fie un număr impar.

Date de intrare
Fișierul **partitie.in** conține pe prima linie numărul *n*.

Date de ieșire
Fișierul **partitie.out** va conține pe prima linie numărul de partiții ale lui *n* conform cerințelor problemei.

Restricție
• $1 \le n \le 160$.

Exemplu

partitie.in	partitie.out
7	5

Explicație
Cele cinci partiții ale numărului 7 sunt:
• 1 + 1 + 1 + 1 + 1 + 1 + 1
• 1 + 1 + 1 + 1 + 3
• 1 + 1 + 5
• 1 + 3 + 3
• 7

Timp de execuție: 3 secunde/test

P050323: Rubine

Ali-Baba și cei *N* hoți ai săi au descoperit harta unei comori pe o insulă izolată. Harta este similară unui caroiaj de $L \cdot C$ regiuni (*L* linii și *C* coloane). În fiecare regiune se află un sipet fermecat, în care se găsesc rubine (0 sau mai multe), numărul de rubine din fiecare sipet fiind păstrat în elementul corespunzător al caroiajului.

Se stabilește următorul plan de acțiune:
• deplasarea hoțului se poate face dintr-o regiune într-o altă regiune vecină (o regiune are maxim opt regiuni vecine);
• fiecare hoț pleacă dintr-o anumită regiune, de coordonate date, spre regiunea în care se află *Ali-Baba* (regiunea din colțul dreapta-jos, de coordonate *L* și *C*), pe drumul cel mai scurt (lungimea unui drum fiind egală cu numărul regiunilor parcurse);
• la trecerea printr-o regiune, hoțul pune într-un sac rubinele din sipetul existent acolo;
• dacă există mai multe drumuri de lungime minimă, un hoț îl va alege pe acela pe care parcurgându-l, reușește să strângă cât mai multe rubine;
• primul hoț care se deplasează este cel cu numărul de ordine 1, urmează apoi cel cu numărul de ordine 2, 3 etc., iar în timpul deplasării unui hoț, ceilalți stau pe loc până ce acesta ajunge la *Ali-Baba*;
• sipetul fiind fermecat, în locul rubinelor luate de hoț, apar altele identice, apariția petrecându-se înaintea plecării fiecărui hoț din punctul lui de pornire spre *Ali-Baba*.

Fiecare hoț a strâns în sacul său un număr de rubine, cunoscut doar de el și ajunge cu sacul în regiunea lui *Ali-Baba*. La împărțirea rubinelor apare și *Sindbad Marinarul* care vrea o parte din saci. *Ali-Baba*, cunoscând anumite relații care există între perechi de hoți din regiune va trebui să aleagă acele relații care să implice repartizarea unui număr minim de saci lui *Sindbad*.

Se știe că alegerea unei perechi de hoți (*i*, *j*) implică repartizarea sacului hoțului *j* din "cauza" lui *i* în visteria lui *Ali-Baba*, însă din acest moment nu se mai poate folosi nici o pereche de forma (*j*, *k*), hoțul *j* fiind eliminat.

Ajutați-l pe *Ali-Baba* să aleagă ordinea perechilor astfel încât sacii care rămân să fie într-un număr cât mai mic posibil, știind că aceștia îi revin lui *Sindbad*.

Date de intrare
Fișierul de intrare **rubine.in** conține:
• pe prima linie dimensiunile caroiajului: valorile lui *L* și *C* separate printr-un spațiu;
• pe următoarele *L* linii se află câte *C* valori separate prin spațiu, reprezentând numărul de rubine din sipetul aflat în regiunea respectivă;
• pe următoarea linie se află numărul de hoți *N*;
• pe următoarele *N* linii se află coordonatele regiunilor (valori separate printr-un spațiu) în care se află inițial hoții; pe prima linie dintre acestea pentru primul hoț, pe linia a doua pentru al doilea hoț etc;
• pe următoarele linii, până la sfârșitul fișierului, se află perechi de numere care reprezintă relațiile din problemă.

Date de ieșire
Fișierul de ieșire **rubine.out** va trebui să conțină:
• pe prima linie numărul *T* de perechi folosite de *Ali-Baba*, respectându-se cerințele problemei;
• pe fiecare dintre următoarele *T* linii câte patru valori separate prin câte un spațiu: $i\ nr_1\ j\ nr_2$, unde *i* reprezintă prima valoare din pereche (hoțul cu numărul de ordine *i*), nr_1 reprezintă numărul de rubine adunate de hoțul *i*, *j* reprezintă al doilea hoț din pereche (hoțul cu numărul de ordine *j*), nr_2 reprezentând numărul de rubine strânse de hoțul *j*;
• pe următoarea linie numerele de ordine ale hoților ai căror saci au rămas la *Sindbad*, valori separate prin câte un spațiu;
• pe ultima linie se află numărul total de rubine din sacii rămași lui *Sindbad*.

Restricții și precizări
• $1 \le L, C, N \le 50$;
• fiecare element din caroiaj este un număr întreg cuprins între 0 și 10;
• întotdeauna există cel puțin o soluție;
• dacă există mai multe soluții, se va alege oricare dintre ele;
• în fiecare regiune, indiferent dacă este punctul de pornire al unui hoț sau regiunea în care se află *Ali-Baba* există câte un sipet cu un anumit număr de rubine. *Ali-Baba* nu cunoaște numărul de rubine din fiecare sac;
• nu există doi hoți care se află inițial în aceeași regiune;
• nu există succesiuni de relații circulare (exemplu: (*i*, *j*) (*j*, *k*) ... (*p*, *t*) (*t*, *i*));

• dacă există un hoț *i* care nu este în nici o relație, sacii săi vor fi repartizați direct lui *Sindbad*.

Exemplu

rubine.in
```
4 5
1 0 0 0 0
0 1 0 0 0
0 0 1 4 0
0 0 0 1 5
5
1 1
1 5
4 1
2 2
4 4
1 2
1 4
2 5
3 1
4 5
```

rubine.out
```
4
2 9 5 6
1 12 2 9
1 12 4 11
3 10 1 12
3
10
```

Timp de execuție: 1 secundă/test

P050324: Scufița

Majoritatea participanților la *ONI* 2003 au auzit în copilărie povestea *Scufiței Roșii*. Pentru cei care o știu, urmează partea a doua; pentru cei care nu o știu, nu vă faceți griji, cunoașterea ei nu este necesară pentru a rezolva această problemă. Povestea nu spune ce s-a întâmplat pe drumul pe care *Scufița Roșie* s-a întors de la bunicuță. Veți afla amănunte în continuare.

Pe drum, ea s-a întâlnit cu *Lupul* (fratele lupului care a părăsit povestea în prima parte). Acesta dorea să o mănânce, dar a decis să-i acorde o șansă de scăpare, provocând-o la un concurs de cules ciupercuțe.

Scufița Roșie se află în poziția (1, 1) a unei matrice cu *N* linii și *N* coloane, în fiecare poziție a matricei fiind amplasate ciupercuțe. *Lupul* se află în poziția (1, 1) a unei alte matrice similare. Pe parcursul unui minut, atât *Scufița*, cât și *Lupul* se deplasează într-una din pozițiile vecine (pe linie sau pe coloană) și culeg ciupercuțele din poziția respectivă. Dacă *Scufița Roșie* ajunge într-o poziție în care nu sunt ciupercuțe, va pierde jocul. Dacă la sfârșitul unui minut ea are mai puține ciupercuțe decât *Lupul*, ea va pierde jocul de asemenea. Jocul începe după ce amândoi participanții au cules ciupercuțele din pozițiile lor inițiale (nu contează cine are mai multe la începutul jocului, ci doar după un număr întreg strict pozitiv de minute de la început). Dacă *Scufița Roșie* pierde jocul, *Lupul* o va mânca.

Înainte de începerea jocului, *Scufița Roșie* l-a sunat pe *Vânător*, care i-a promis că va veni într-un sfert de oră (15 minute) pentru a o salva. Deci *Scufița Roșie* va fi liberă să plece dacă nu va pierde jocul după 15 minute.

Din acest moment, scopul ei este nu numai să rămână în viață, ci și să culeagă cât mai multe ciupercuțe, pentru a le duce acasă (după ce va veni, vânătorul nu o va mai lăsa să culeagă).

Lupul, cunoscut pentru lăcomia sa proverbială, va alege la fiecare minut mutarea în câmpul vecin cu cele mai multe ciupercuțe (matricea sa este dată astfel încât să nu existe mai multe posibilități de alegere la un moment dat).

Povestea spune că *Scufița Roșie* a plecat acasă cu coșulețul plin de ciupercuțe, folosind indicațiile date de un program scris de un concurent la *ONI*2003 (nu vom da detalii suplimentare despre alte aspecte, cum ar fi călătoria în timp, pentru a nu complica inutil enunțul problemei). Să fi fost acest program scris de dumneavoastră? Vom vedea...

Scrieți un program care să o ajute pe *Scufița Roșie* să rămână în joc și să culeagă cât mai multe ciupercuțe la sfârșitul celor 15 minute!

Date de intrare

Fișierul **scufita.in** are următoarea structură:

• pe prima linie se află numărul natural *N* care reprezintă numărul de linii și de coloane ale celor două matrice;
• pe următoarele *N* linii se află matricea din care culege *Scufița Roșie*; fiecare linie va conține *N* numere naturale, separate prin câte un singur spațiu; un astfel de număr arată câte ciupercuțe se află în poziția corespunzătoare;
• pe următoarele *N* linii se află matricea din care culege *Lupul*; fiecare linie va conține *N* numere naturale, separate prin câte un singur spațiu; un astfel de număr arată câte ciupercuțe se află în poziția corespunzătoare.

Date de ieșire

Fișierul **scufita.out** va trebui să conțină pe prima linie numărul total de ciupercuțe culese de *Scufița Roșie* iar pe următoarea linie o succesiune de caractere din mulțimea {N, E, S, V} care reprezintă direcțiile pe care s-a deplasat *Scufița Roșie* (N - *Nord*; E - *Est*; S - *Sud*; V - *Vest*). Aceste caractere nu vor fi separate prin spații.

Restricții și precizări

• $4 \le N \le 10$;
• valorile din cele două matrice sunt numere naturale mai mici decât 256;
• nici *Scufița Roșie* și nici *Lupul* nu vor părăsi matricele corespunzătoare;
• după ce unul din jucători culege ciupercuțele dintr-o poziție, în poziția respectivă rămân 0 ciupercuțe;
• *Scufița Roșie* va avea întotdeauna posibilitatea de a rezista 15 minute;
• poziția (1, 1) este situată în colțul de *Nord-Vest* al matricelor.

Exemplu

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

scufita.out
```
137
SSSEEENVVNEENVV
```

Explicație
Scufița Roșie a efectuat aceleași mutări cu cele efectuate de *Lup* și a avut tot timpul o ciupercuță în plus. În final ea a cules toate ciupercuțele din matrice.

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P050325: A007

Agentul 007 are de distrus o tabără de teroriști. Tabăra de teroriști este formată din mai multe obiective (depozite de muniție, pavilioane pentru teroriști etc.), considerate punctiforme în plan.

Agentul 007 primește de la serviciul de informații o hartă cu n obiective din tabăra teroriștilor, date prin coordonatele carteziene.

Pe lângă hartă, agentul 007 mai primește și o armă specială (construită pentru această misiune). Arma primită are două țevi și permite tragerea simultană pe aceeași direcție (rectilinie), dar în sens invers a două rachete cu aceeași viteză.

După ce se trage cu arma, odată cu atingerea unei ținte explodează și cealaltă rachetă (chiar dacă aceasta din urmă nu și-a atins ținta).

Agentul 007 vrea să distrugă tabăra cât mai repede și cu cât mai puține rachete. Pentru acest lucru el studiază posibilitatea să se așeze într-un punct din tabără (diferit de obiective) care să permită trageri eficace, adică la fiecare tragere să distrugă câte două obiective simultan.

Determinați dacă este posibil să se găsească un astfel de punct.

Date de intrare
În fișierul **a007.in** pe prima linie se află numărul de teste k, după care urmează date pentru fiecare test. Pentru fiecare test, pe o linie se află numărul n, iar pe următoarele n linii sunt coordonatele obiectivelor din tabăra teroriștilor (separate printr-un spațiu în ordinea abscisă ordonată).

Date de ieșire
În fișierul **a007.out** se vor scrie k linii, pe fiecare linie se va scrie 1, dacă există soluție și 0 dacă nu există soluție. În cazul în care există soluție se va scrie în continuare pe aceeași linie, separate printr-un spațiu, coordonatele punctului cerut (numere reale trunchiate la patru zecimale, în ordinea abscisă ordonată).

Restricții și precizări
- $0 \le n \le 10.000$;
- $1 \le k \le 3$;
- coordonatele punctelor sunt întregi din intervalul [-10.000, 10000];
- un obiectiv este distrus dacă racheta explodează exact în punctul corespunzător lui;
- coordonatele punctelor sunt distincte.

Exemplu

a007.in
```
2
4
10 0
10 10
0 10
0 0
6
0 0
10 0
2 10
12 0
5 0
7 0
```

a007.out
```
1 5.0000 5.0000
0
```

Timp de execuție: 0,2 secunde/test

P050326: Asmin

Se consideră un arbore (graf conex aciclic) cu N vârfuri, fără rădăcină fixată. Drept rădăcină, poate fi ales oricare dintre vârfuri. Să presupunem că a fost ales vârful cu numărul T. Între oricare vârf și T există un drum unic care conține fiecare vârf al arborelui cel mult o singură dată (un drum între vârfurile i și j este o secvență de vârfuri, care începe cu i, se termină cu j, iar între oricare două vârfuri consecutive există o muchie în arbore).

Fiecărui vârf i (inclusiv T) trebuie să i se asocieze o valoare V_i, mai mare sau egală cu 0, astfel încât suma valorilor vârfurilor de pe drumul dintre i și rădăcina T, împărțită la K, să dea restul R_i. Se definește costul arborelui cu rădăcina fixată în T, C_T, ca fiind suma valorilor asociate fiecărui nod. Dintre toate posibilitățile de alegere a valorilor V_i care respectă condiția precizată anterior, se va alege aceea pentru care C_T este minim.

Se constată ușor că alegând o altă rădăcină, de exemplu, vârful S (diferit de T), C_S nu este neapărat egal cu C_T.

Dându-se un arbore cu N vârfuri, un număr întreg K și valorile R_i, $i = 1, 2, \ldots, N$, corespunzătoare fiecărui vârf, determinați acele vârfuri T care pot fi alese drept rădăcină, pentru care costul C_T este minim (adică $C_T \le C_S$, oricare ar fi S diferit de T), precum și costul respectiv.

Date de intrare
Pe prima linie a fișierului de intrare **asmin.in** se află două valori întregi: N și K. Pe următoarele N - 1 linii se află câte două numere întregi a și b, separate printr-un spațiu, având semnificația că există muchie între vârfurile a și b. Vârfurile sunt numerotate de la 1 la N. Pe următoarea linie se află N numere întregi, reprezentând valorile R_i, $i = 1, \ldots, N$.

Date de ieșire
Pe prima linie a fișierului de ieșire **asmin.out** se vor afișa două valori întregi: C și M. C reprezintă costul minim posibil al arborelui. M reprezintă numărul de vârfuri care pot fi alese drept rădăcină și pentru care se obține costul C. Pe a doua linie se află M numere întregi separate prin câte un spațiu, scrise în ordine crescătoare, reprezentând numerele vârfurilor care pot fi alese ca rădăcină, astfel încât să se obțină costul C.

Restricții
- $2 \le N \le 16.000$;
- $2 \le K \le 1.000$;
- $2 \le R_i \le K - 1$.

Exemplu

asmin.in
```
5 3
1 2
1 3
2 4
2 5
0 1 2 1 0
```

asmin.out
```
5 2
1 5
```

Timp de execuție: 0,2 secunde/test

P050327: Căutare

Știm cu toții ce este un arbore binar de căutare. Este acel arbore binar în care informația din orice nod este mai mare decât informațiile nodurilor din subarborele stâng al nodului respectiv și mai mică decât cele din subarborele drept.

Când se caută o informație într-un arbore binar de căutare, se începe din rădăcina arborelui și valoarea căutată se compară cu informația din rădăcină.

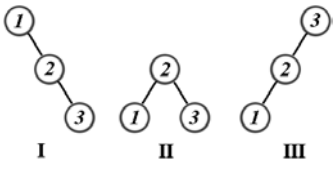
Dacă informația căutată e mai mică decât informația din rădăcină, se continuă căutarea în subarborele stâng, iar dacă e mai mare se continuă căutarea în subarborele drept.

Dacă cele două informații sunt egale, căutarea se termină cu succes.

Dacă în direcția în care continuăm căutarea (stânga sau dreapta) subarborele nu mai are noduri, înseamnă că informația căutată nu se găsește în arbore. Evident, numărul de comparații efectuate la o căutare depinde de distanța dintre rădăcină și nodul în care se găsește informația, respectiv cel la care putem decide că informația nu se află în arbore.

Ce s-ar întâmpla dacă înainte de a construi arborele binar de căutare am ști ce informații urmează să fie căutate în el? Nu cumva am putea construi arborele în așa fel încât să minimizăm numărul de comparații efectuate?

De exemplu, cu informațiile 1, 2 și 3 putem construi un arbore binar de căutare în următoarele trei moduri (din totalul de cinci posibile).



Dacă vom căuta informațiile (1, 1, 3, 1, 2), atunci timpul total de căutare este pentru arborele din stânga 1 + 1 + 3 + 1 + 2 = 8, pentru arborele din centru 2 + 2 + 2 + 2 + 1 = 9, iar pentru arborele din dreapta 3 + 3 + 1 + 3 + 2 = 12.

Deci arborele din stânga este cel adecvat pentru căutările noastre, iar timpul total de căutare minim este 8.

De remarcat că, dacă se caută o informație care nu există în arbore, atunci numărul de comparații efectuate la căutare este egal cu nivelul ultimului nod interogat. De exemplu, dacă se caută informația 4 pe cei trei arbori, atunci timpii vor fi: 3, 2, respectiv 1 (de la stânga la dreapta).

Scrieți un program care, pentru anumite informații căutate, determină timpul total de căutare minim.

Date de intrare

Din fișierul **cautare.in** se citește de pe prima linie numărul T al testelor. În fișier urmează cele T teste. Pentru fiecare test în fișier sunt scrise următoarele linii:
- prima linie conține N, numărul de noduri ale arborelui de căutare și M, numărul de interogări, separate prin spațiu;
- pe următoarea linie urmează N numere întregi distincte, separate prin câte un spațiu, reprezentând informațiile din nodurile arborelui;
- pe fiecare dintre următoarele M linii sunt câte două numere întregi separate printr-un spațiu, reprezentând un număr căutat și respectiv de câte ori a fost căutat (între 1 și 100).

Date de ieșire

În fișierul **cautare.out** se va scrie, pentru fiecare test câte o linie care conține timpul total minim de căutare pentru interogările din testul respectiv.

Restricții și precizări
- $0 < T, N < 101$;
- $0 < M < 1001$;
- informațiile nodurilor arborelui sunt numere întregi din intervalul [-10000, 10000];
- informațiile căutate sunt numere întregi cuprinse în intervalul [-1000000, 1000000] și pot fi căutate de un număr de ori cuprins între 1 și 100.

Exemplu

cautare.in
```
3
3 3
1 2 3
1 3
2 1
3 1
3 3
1 2 3
1 50
2 49
3 51
3 2
1 2 3
2 1
4 20
```

cautare.out
```
8
251
22
```

Timp de execuție:
- *Microsoft Windows*: 1,5 secunde/test
- *Linux*: 0,3 secunde/test

P050328: Inter

În țara *Smar* sunt N autostrăzi, sub forma unor drepte în plan. Se știe că la intersecții de drumuri (care includ și autostrăzi) există un risc ridicat de accidente. De aceea polițiștii din această țară au hotărât stabilirea unei zone compacte care să includă toate intersecțiile și în care să se supravegheze atent circulația. Din motive financiare zona trebuie să fie de perimetru minim.

Scrieți un program care să determine aria zonei de supraveghere alese.

Date de intrare
Fișierul de intrare **inter.in** va conține pe prima linie numărul de autostrăzi, iar de pe fiecare dintre următoarele N linii se vor citi câte patru numere reale, separate prin câte un spațiu, reprezentând coordonatele a două puncte distincte care determină câte o dreaptă. Ele sunt date în ordinea $X_1\ Y_1\ X_2\ Y_2$, adică abscisa și ordonata primului punct, apoi abscisa și ordonata celui de-al doilea punct.

Date de ieșire
Fișierul de ieșire **inter.out** va conține o singură linie pe care se va afla un număr real, cu două zecimale exacte (rotunjit), care reprezintă aria zonei alese pentru supraveghere.

Restricții și precizări
- între oricare două autostrăzi există exact o intersecție;
- aria suprafeței de supraveghere este întotdeauna strict pozitivă;
- $2 \le N \le 5001$.

Exemplu
inter.in
```
4
0 0 1 0
0 0 0 2
0 2 1 0
-2 0 0 1
```
inter.out
```
3.00
```

Timpi de execuție:
- *Microsoft Windows*: 1,5 secunde/test
- *Linux*: 0,2 secunde/test

P050329: Număr
Se consideră un număr natural x, scris în baza 10, care conține exact n cifre. Scrieți un program care să determine cel mai mic număr natural strict mai mare decât x, care are aceleași cifre ca și numărul x și care este palindrom.

Date de intrare
Fișierul de intrare **nr.in** conține două linii. Pe prima linie este scris n, numărul de cifre ale numărului x. Pe cea de a doua linie sunt scrise cele n cifre ale lui x.

Date de ieșire
Fișierul de ieșire **nr.out** conține o singură linie pe care se află cel mai mic număr natural strict mai mare decât x, care are aceleași cifre ca și numărul x și care este palindrom. Dacă nu există soluție, pe prima linie a fișierului de ieșire va fi scrisă valoarea 0.

Restricții și precizări
- $1 \le N \le 1000$;
- numim palindrom un număr care citit de la stânga la dreapta, cât și de la dreapta la stânga este același (de exemplu 1331, 12321 etc);
- prima cifră a unui număr trebuie să fie nenulă;
- prin aceleași cifre se înțelege că fiecare cifră apare în rezultat de același număr de ori ca și în numărul x.

Exemple
nr.in
```
5
12022
```
nr.out
```
0
```

nr.in
```
5
12200
```
nr.out
```
20102
```

Timp de execuție: 0,1 secunde/test

P050330: Proc
O aplicație care trebuie executată pe un calculator multiprocesor constă din N fragmente de cod independente, care pot fi rulate în paralel. Fiecare fragment trebuie executat în totalitate pe un singur procesor. Din dorința de a paraleliza cât mai mult aplicația, fragmentele de cod au dimensiuni mici și, în consecință, timpi de execuție mici. Mai precis, execuția fiecărui fragment durează ***unul*** sau ***două*** cicluri de ceas pe un procesor de tipul *Pentium* IV.

Sistemul pe care urmează să fie executată aplicația constă din P procesoare. Dar, spre deosebire de majoritatea sistemelor de acest fel, cele P procesoare au viteze de execuție diferite. Primul procesor este un *Pentium* IV și este cel mai rapid. Al doilea procesor este de două ori mai lent decât primul, al treilea de trei ori mai lent, ..., al i-lea procesor este de i ori mai încet decât primul. În aceste condiții, timpul de execuție al fiecărui fragment de cod diferă, în funcție de procesorul pe care va fi executat. Să presupunem că un segment de cod are timpul de execuție T (unde T este 1 sau 2) pe primul procesor. Atunci timpul de execuție al procesorului cu numărul de ordine i va fi $i \cdot T$.

Știind că fragmentele de cod pot fi executate în orice ordine și pe orice procesor și că orice procesor poate executa, la un moment dat, un singur fragment de cod, determinați timpul minim după care se va termina execuția aplicației (adică a tuturor fragmentelor de cod). Timpul după care se termină aplicația este egal cu maximul dintre timpii după care fiecare procesor redevine disponibil. Timpul după care un procesor redevine disponibil este egal cu suma timpilor de execuție a fragmentelor de cod rulate pe procesorul respectiv.

Date de intrare
Fișierul de intrare **proc.in** conține pe o singură linie trei numere întregi, separate printr-un singur spațiu: N - numărul de fragmente de cod, K - numărul de fragmente de cod care au timpul de execuție pe un *Pentium* IV, egal cu 1 (implicit, $N - K$ fragmente de cod au timpul de execuție egal cu 2) și P - numărul de procesoare ale sistemului.

Date de ieșire
Fișierul **proc.out** va conține o singură linie pe care se află timpul minim după care se termină de executat aplicația.

Restricții
- $0 \le K \le N \le 1.000.000.000$;
- $0 \le K \le N \le 65535$.

Exemplu

proc.in	proc.out
4 3 2	4

Explicație

Pe primul procesor se execută un fragment de cod cu timpul de execuție (calculat pe un *Pentium* IV) egal cu 1 și un fragment de cod cu timpul de execuție egal cu 2 ⇒ timpul după care acest procesor devine disponibil este 1 · 1 + 1 · 2 = 3. Pe al doilea procesor se execută două fragmente de cod cu timpul de execuție (calculat pe un *Pentium* IV) egal cu 1 ⇒ timpul după care acest procesor devine disponibil este 2 [*numărul de fragmente*] · (2 · 1) [*timpul de execuție al fiecărui fragment pe procesorul* 2] = 4.

Timpi de execuție:

- *Microsoft Windows*: 1,5 secunde/test
- *Linux*: 0,2 secunde/test

Baraj

P050331: Excursie

În una dintre zile, la *Olimpiada de Informatică* se organizează *P* excursii atractive. La aceste excursii participă în total *N* persoane. Pentru simplitate, persoanele au fost numerotate de la 1 la *N*, primele *K* persoane fiind ghizii. O persoană se poate înscrie la exact una dintre cele *P* excursii organizate.

Pentru a evita surprizele neplăcute (insuficiente mijloace de transport, insuficiente locuri la restaurant etc) organizatorii intenționează să studieze toate configurațiile care pot să apară în urma înscrierilor participanților, considerând că în fiecare excursie va exista cel puțin un participant.

Scrieți un program care să determine numărul de configurații distincte care se pot obține după înscrierea celor *N* persoane la cele *P* excursii organizate, astfel încât cei *K* ghizi să fie înscriși în excursii diferite.

Date de intrare

Fișierul de intrare se numește **ex.in** și conține o singură linie pe care se află trei numere naturale separate prin câte un spațiu: *N K P* (reprezentând numărul de persoane, numărul de ghizi și respectiv numărul de excursii).

Date de ieșire

Fișierul de ieșire **ex.out** conține o singură linie pe care se află numărul de configurații distincte.

Restricții

- $0 \le K \le P \le N \le 1.000.000.000$;
- într-o configurație nu contează ordinea excursiilor sau ordinea în care se înscriu persoanele la o excursie.

Exemplu

ex.in	ex.out
5 3 4	7

Explicație

Cele șapte configurații distincte sunt:

```
(1,4)(2)(3)(5)
(1)(2,4)(3)(5)
(1)(2)(3,4)(5)
(1)(2)(3)(4,5)
(1,5)(2)(3)(4)
(1)(2,5)(3)(4)
(1)(2)(3,5)(4)
```

Timpi de execuție:

- *Microsoft Windows*: 0,5 secunde/test
- *Linux*: 0,1 secunde/test

P050332: Robot

Lucrați la o firmă care produce microprocesoare. Pentru asamblarea microprocesoarelor, firma utilizează un robot constituit dintr-un singur braț.

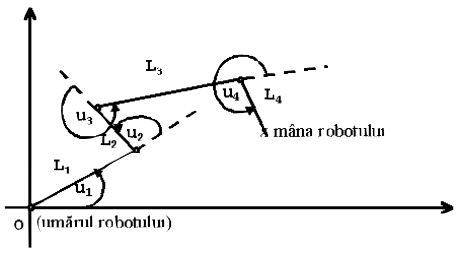
Brațul este fixat la unul dintre capete ("umărul") într-un punct plasat în centrul platformei de lucru, iar la celălalt capăt are un dispozitiv de lungime neglijabilă cu care poate "culege" componentele de pe platforma de lucru ("mâna"). Brațul se poate mișca numai în plan orizontal, deasupra platformei de lucru.

Brațul este constituit dintr-o succesiune de *N* segmente rigide de lungimi L_1, L_2, ..., L_N, conectate prin puncte de articulație. Mai exact, primul segment este conectat printr-un punct de articulație în umărul robotului, al doilea segment este conectat printr-un punct de articulație de primul segment etc. Ultimul segment este conectat printr-un punct de articulație de penultimul segment și are la celălalt capăt "mâna". Un punct de articulație permite rotația liberă (la orice unghi) a segmentului conectat în acel punct de articulație.

Pentru asamblarea unui microprocesor, robotul trebuie să culeagă succesiv componentele acestuia de pe platforma de lucru. Fiecare componentă are o poziție bine determinată pe platforma de lucru, prin coordonatele sale relativ la un sistem de coordonate cartezian, cu centrul în "umărul robotului".

Rolul dvs. în firmă este de a programa mișcările robotului. În acest scop, pentru fiecare componentă pe care robotul o va culege trebuie să specificați "configurația" brațului robotului care să permită atingerea componentei respective (mâna robotului să fie plasată deasupra poziției în care se află componenta).

Configurația brațului robotului este dată de unghiurile dintre segmentele brațului rigid.



Scrieți un program care, pentru o poziție dată, determină o configurație pentru brațul robotului care să-i permită acestuia să culeagă componenta din poziția respectivă, dacă este posibil.

Date de intrare
Fișierul de intrare **robot.in** conține pe prima linie un număr natural N, care reprezintă numărul de segmente din care este format brațul robotului.

Pe fiecare dintre următoarele N linii se află câte un număr natural. Numărul aflat pe a $(i + 1)$ - a linie reprezintă lungimea celui de-al i-lea segment al brațului robotului.

Pe ultima linie se află două numere întregi x și y, separate prin câte un spațiu, reprezentând coordonatele poziției la care trebuie să ajungă "mâna" robotului.

Date de ieșire
Fișierul de ieșire **robot.out** va conține o singură linie pe care se află valoarea 0 dacă nu este posibil ca mâna robotului să ajungă în poziția x, y. Dacă problema are soluție, fișierul de ieșire conține N linii. Pe linia i se află valoarea reală u_i care reprezintă unghiul dintre segmentul i și segmentul i - 1 (pentru orice i de la 2 la N), iar valoarea u_1 reprezintă unghiul pe care segmentul 1 îl formează în umărul robotului cu axa Ox.

Restricții și precizări
- $0 \le N \le 10.000$;
- $1 \le L_i \le 200$;
- $0 \le u_i < 360$;
- $-10.000 \le x, y \le 10.000$;
- unghiurile se măsoară în sens trigonometric și sunt exprimate în grade.

Exemple

robot.in	robot.out
3 10 5 25 15 20	125.6725 0 252.5424

robot.in	robot.out
3 10 5 25 2 4	0

Timpi de execuție:
- *Microsoft Windows*: 0,3 secunde/test
- *Linux*: 0,1 secunde/test

P050333: Telegraf
Până nu demult, comunicația la distanță se făcea cu ajutorul telegrafului. Folosind telegraful, se pot transmite două tipuri de semnale: punct și linie. În general, dorim să transmitem texte formate din litere ale alfabetului latin și cifre (în total, 36 de simboluri). Trebuie deci să folosim o codificare, adică să asociem fiecăruia dintre cele 36 de simboluri o succesiune distinctă de linii și puncte. Pentru a putea decodifica o succesiune recepționată de linii și puncte, este necesar ca nici un simbol să nu aibă o codificare identică cu începutul codificării pentru un alt simbol. Să considerăm câteva exemple (presupunând că nu vrem să transmitem decât literele A, B, C):
- exemplul 1: A = ·–; B = –·; C = –.
- exemplul 2: A = ·––; B = ·–; C = –.
- exemplul 3: A = ·–·–; B = –·; C = ·–·.

Primul exemplu reprezintă o codificare corectă. Al doilea exemplu reprezintă o codificare greșită, pentru că începutul codificării lui A este identic cu codificarea pentru B (deci, o secvență de genul ·–– este ambiguă, putând însemna fie A fie BC). Al treilea exemplu reprezintă, de asemenea, o codificare greșită pentru că începutul codificării pentru A este identic cu codificarea pentru C (o secvență precum ·–·–·– este ambiguă, putând însemna fie AB, fie CC).

Se știe că într-o transmisie telegrafică, punctul durează o secundă, iar linia două secunde. Putem calcula astfel timpul necesar transmiterii unui text.

Folosind codificarea din primul exemplu, transmiterea textului $CABCA$ = –·–·––·–·· durează 11 secunde. Observați că lungimea transmisiei se poate calcula și astfel: 2 · (*A*) + 1 · (*B*) + 2 · (*C*) = 2 · (··) + 1 · (·–) + 2 · (–) = 2 · 2 + 1 · 3 + 2 · 2 = 11.

Se consideră un text, dat prin frecvența apariției fiecărui simbol (dintre cele 36 considerate). Să se găsească durata minimă necesară transmiterii acelui text, folosind o codificare aleasă corespunzător.

Date de intrare
Fișierul **telegraf.in** conține o singură linie cu 36 de numere întregi nenegative, separate prin câte un spațiu, re-

prezentând numărul de apariții ale fiecărui simbol în textul care trebuie transmis.

Date de ieșire

Fișierul **telegraf.out** va conține un singur număr, și anume lungimea minimă (în secunde) necesară pentru transmiterea textului.

Restricții și precizări

- nici unul din cele 36 de simboluri nu apare de mai mult de 1.000.000 de ori în textul considerat;
- există cel puțin două simboluri cu număr de apariții nenul.

Exemplu

telegraf.in

```
2 12000000000000000000000000000000000000
```

telegraf.out

```
11
```

Timpi de execuție:

- *Microsoft Windows*: 0,3 secunde/test
- *Linux*: 0,1 secunde/test

P050334: Ajutor

Dacă te-ai rănit, nu folosi *Sprite* să tratezi rana. Cel mai bine e să fugi la cel mai apropiat post de prim ajutor. Norocul tău este că ai o hartă din care poți afla coordonatele carteziene ale posturilor de prim ajutor. Ghinionul este că, din cauza durerii, nu poți să fugi direct spre un post, ci numai pe direcțiile nord-sud și est-vest. Ca să știi dacă mai are sens să fugi spre un post sau să te bucuri de ultimele clipe de viață, cel mai bine e să evaluezi distanța până la cel mai apropiat post de prim ajutor. Cel mai apropiat post este cel pentru care distanța *Manhattan* este minimă.

De data aceasta ai scăpat pentru că timpul necesar ajungerii până la post a fost suficient; însă îți pui întrebarea: dacă accidentul s-ar fi întâmplat în alt loc, ai fi scăpat?

Se consideră N puncte în plan, reprezentând posturile de prim ajutor și alte M puncte reprezentând posibile locații ale accidentului. Se cere pentru fiecare dintre cele M puncte distanța *Manhattan* până la cel mai apropiat post.

Date de intrare

Fișierul **ajutor.in** conține pe prima linie numerele întregi N și M, în această ordine, separate printr-un spațiu. Pe următoarele N linii se află câte două numere întregi, separate printr-un spațiu, reprezentând coordonatele fiecărui post de prim ajutor. Pe următoarele M linii se află câte două numere întregi, separate printr-un spațiu, reprezentând coordonatele punctelor de accident. Coordonatele se dau în ordinea: abscisă, ordonată.

Date de ieșire

Fișierul **ajutor.out** va conține M linii, cu câte un număr pe fiecare linie, reprezentând distanța minimă până la cel mai apropiat post de prim ajutor.

Restricții și precizări

- $0 < N < 401$;
- $0 < M < 500.001$;
- toate coodonatele sunt numere întregi cuprinse în intervalul [0, 32000];
- dacă te afli deja la un post de prim ajutor (coordonatele sunt identice) distanța este 0;
- distanța *Manhattan* este cel mai scurt drum între două puncte, mergând doar pe direcții paralele cu axele de coordonate, adică $|x_1 - x_2| + |y_1 - y_2|$, unde (x_1, y_1) și (x_2, y_2) sunt coordonatele celor două puncte;
- pot exista puncte cu aceleași coordonate.

Exemplu

ajutor.in

```
4 4
1 1
5 5
1 5
5 1
0 0
1 1
3 3
4 1
```

ajutor.out

```
2
0
4
1
```

Timpi de execuție:

- *Microsoft Windows*: 2 secunde/test
- *Linux*: 1,3 secunde/test

P050335: Ghizi

Se caută ghizi pentru *Olimpiada Națională de Informatică*. Deoarece la *Olimpiadă* participă K echipe, trebuie ca în fiecare moment de timp din intervalul [0, 100) să existe exact K ghizi.

Pentru posturile de ghid s-au înscris N voluntari, care au fost numerotați cu numere distincte cuprinse între 1 și N. Fiecare dintre cei N voluntari a specificat un interval de timp în care poate asigura servicii de ghid. Voluntarul i poate fi ghid în intervalul $[T1_i, T2_i)$ (intervalul este închis în $T1_i$ și deschis în $T2_i$).

Dându-se intervalele de timp asociate celor N voluntari, determinați o variantă de angajare, astfel încât în fiecare moment de timp să fie prezenți exact K ghizi. Numărul total de voluntari angajați este irelevant.

Date de intrare

Prima linie a fișierului de intrare **ghizi.in** conține două numere întregi N și K, separate printr-un spațiu, cu semnificațiile de mai sus. Voluntarii sunt numerotați distinct, de la 1 la N. Fiecare dintre următoarele N linii conține descrierea unui voluntar; mai exact linia $i + 1$ conține valorile întregi $T1_i$ și $T2_i$ pentru voluntarul i.

Date de ieșire

În fișierul **ghizi.out** veți afișa pe prima linie numărul total al ghizilor angajați (M). Pe a doua linie veți scrie M numere distincte între 1 și N, ordonate crescător, reprezentând numerele de ordine ale ghizilor angajați.

Restricții și precizări
- $1 \le N \le 5.000$;
- $0 \le T1 \le T2 \le 100$ pentru fiecare dintre cei N voluntari;
- $0 \le K \le N$;
- pot exista doi voluntari cu același interval asociat;
- întotdeauna va exista soluție;
- dacă există mai multe soluții, se va afișa una oarecare.

Exemplu

ghizi.in	ghizi.out
6 2 0 100 0 15 15 99 0 10 10 20 20 100	4 1 4 5 6

Timpi de execuție:
- *Microsoft Windows*: 0,3 secunde/test
- *Linux*: 0,1 secunde/test

P050336: Sala

Președintele unui *Concurs de Informatică* dorește ca la festivitatea de încheiere să fie o atmosferă plăcută. Pentru acest lucru el vrea să așeze participanții pe n rânduri după anumite reguli. Aceste reguli sunt:
- pe primul rând al sălii așează una lângă alta n persoane;
- pe rândurile următoare așeazarea pe poziția i (numerotarea se face de la stânga la dreapta) este condiționată de așezarea persoanelor de pe rândul anterior, adică dacă pe rândul din față pe pozițiile i și $i + 1$ stau fie numai băieți fie numai fete, atunci se va așeza o fată, iar dacă pe aceste poziții stau persoane de sex opus se va așeza un băiat.

Conform acestei reguli pe rândul cu numărul de ordine i ($i \in \{1, 2, \ldots, n\}$) se vor așeza $n - i + 1$ persoane.

Pentru n dat se cere să se determine numărul maxim de băieți care pot lua loc în sală, astfel încât să se respecte regulile anterioare.

Date de intrare
În fișierul de intrare **sala.in** pe prima linie se află numărul de cifre ale lui n, iar pe linia a doua se află cifrele numărului n separate între ele prin câte un spațiu.

Date de ieșire
În fișierul de ieșire **sala.out** pe prima linie se va afișa numărul din cerință.

Restricție
- $1 \le n \le 10^{101}$.

Exemple

sala.in	sala.out
1 5	10

sala.in	sala.out
2 1 3	61

Timp de execuție: 0,1 secunde/test

Câștigătorii

Clasa a V-a
- *Alexandru-Stan Mile, Timișoara - premiul I*
- *Mihail Drăghici, Ploiești - premiul II*
- *Robert Asaftei, Vaslui - premiul III*

Clasa a VI-a
- *Cotizu Sima, Ploiești - premiul I*
- *Adela Lica, Ploiești - premiul II*
- *Ana Moisin, București - premiul III*

Clasa a VII-a
- *Filip Buruiana, Galați - premiul I*
- *Mihai Morariu, București - premiul II*
- *Adela Rădoi, Drobeta Turnu-Severin - premiul II*

Clasa a VIII-a
- *Valentin Stanciu, București - premiul I*
- *Sorin Dascălu, Piatra Neamț - premiul II*
- *Andrei Ioniță, Suceava - premiul II*

Clasa a IX-a
- *Mircea Pașoi, Ploiești - premiul I*
- *Dan Spătărel, București - premiul II*
- *Radu Marin, Focșani - premiul III*

Clasa a X-a
- *Sorin Stancu-Mara, București - premiul I*
- *Daniel Fechete, Suceava - premiul II*
- *Leonard Crestez, Brăila - premiul II*
- *Gabriel Rizuc, Iași - premiul III*
- *Korodi Andor-Csaba, Târgu Mureș - premiul III*

Clasa a XI-a
- *Silviu Gânceanu, Vaslui - premiul I*
- *Emilian Miron, Bacău - premiul I*
- *Andrei Ion, Constanța - premiul II*
- *Mircea Digulescu, București - premiul III*
- *Dan Ghinea, București - premiul III*

Clasa a XII-a
- *Radu Berinde, București - premiul I*
- *Tiberiu Dăneț, București - premiul II*
- *Andrei Giurgiu, București - premiul III*

Membrii Lotului Național de Informatică:

Costel-Cătălin Antohi, Galați
Radu Berinde, București
Ștefan Ciobâcă, Suceava
Victor Costan, București
Leonard Crestez, Brăila
Tiberiu Dăneț, București
Mircea Digulescu, București
Constantin Dumitrașcu, Focșani
Dan-Ionuț Fechete, Suceava
Ștefan Gheorghe, București
Dan Ghinea, București
Matei Gruber, București
Mihnea Giurgea, Ploiești
Andrei Homescu, Gorj
Andrei-Gabriel Ion, Constanța
Emilian Miron, Bacău
Alexandru Moșoi, București
Marius Nicolae, Slatina
Mircea Pașoi, Ploiești
Iolanda Popa, Iași