Opgaven

voor het

Delfts **K**ampioenschap **P**rogrammeren 1999

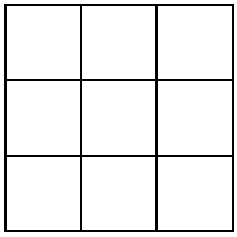
De wedstrijd bestaat uit de volgende opgaven:

A Hokjes tellen
B Convexië
C Benzine
D Taart-delingen voor gevorderden
E Tegelen
F Sommen
G Feest 2000

A Hokjes tellen

Inleiding

In jeugdtijdschriften staan vaak raadsels van de volgende vorm:



Hoeveel vierkanten bevinden zich in deze figuur?

Voor de lezers is het vaak verrassend om te ontdekken dat er naast het "grote" vierkant en de $3 \times 3 = 9$ "kleine" vierkantjes, ook 4 vierkanten van grootte 2×2 voorkomen, zodat het totaal aantal vierkanten op 14 komt.

Een variant op dit raadsel is het tellen van het aantal rechthoeken. Het verifiëren dat er in de bovenstaande figuur 36 rechthoeken voorkomen, wordt aan de lezer overgelaten.

Probleem

De wizz-kids onder de lezers lossen deze raadsels niet direct op, maar schrijven er een computerprogramma voor. Ook aan jullie wordt gevraagd een programma te schrijven voor het tellen van het aantal vierkanten en rechthoeken in een rooster.

Invoer (lezen uit: `hokjes.in`)

De invoer begint met een regel met het aantal roosters waarin vierkanten en rechthoeken geteld moeten worden.

Voor elk rooster wordt de afmeting gegeven, uitgedrukt in het aantal "1×1-vierkantjes." Per rooster wordt op één regel de breedte en de hoogte gegeven, gescheiden door een spatie. $1 \leq b \leq 300$, $1 \leq h \leq 300$.

Uitvoer (schrijven naar standard-output)

De uitvoer dient te bestaan uit één regel voor elk rooster, met daarop respectievelijk het aantal vierkanten en het aantal rechthoeken, gescheiden door één spatie.

Voorbeeld

Invoer

```
2
3 3
4 2
```

Uitvoer

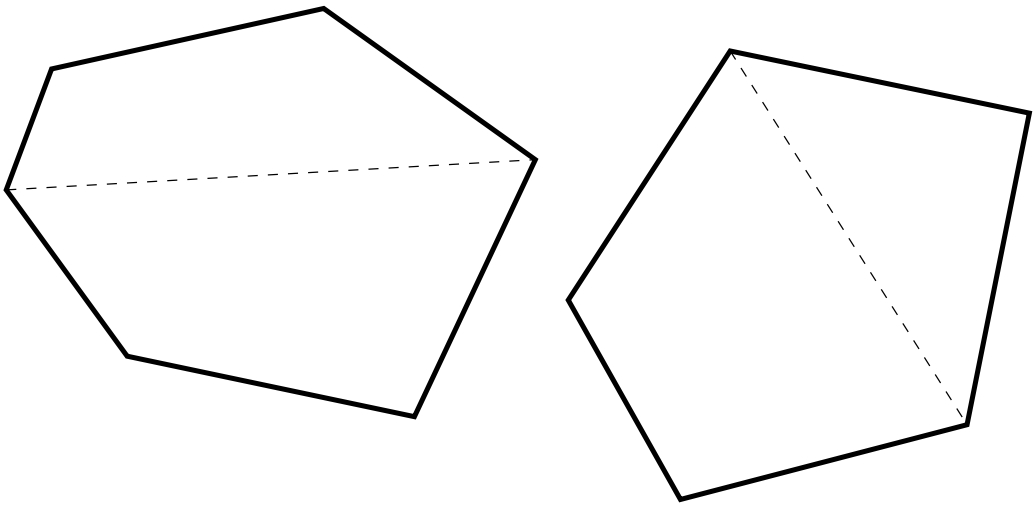
```
14 36
11 30
```

B Convexië

Beschrijving

Zoals bekend is *Convexië* een eilandengroep in de Verre Oceaan. De eilanden hebben een kenmerkende vorm: de kust bestaat uit aaneengesloten rechte lijnen. Bovendien is elk eiland convex; dat wil zeggen dat je van elk hoekpunt in een rechte lijn naar een willekeurig ander hoekpunt kan lopen, zonder dat je water hoeft over te steken. Convexië is zeer dun bevolkt; op de meeste eilanden woont slechts één persoon. In verband met de bevolkingsgroei moet daar nu verandering in komen.

De koning heeft besloten dat op elk eiland best twee mensen kunnen wonen. Omdat convexianen hun privacy erg belangrijk vinden, moet er op elk eiland een duidelijke grens komen tussen de gebieden van de bewoners. Om het eenvoudig te maken wil de koning op elk eiland de grens vaststellen als een rechte lijn tussen twee van de hoekpunten van het eiland. Natuurlijk is het ook belangrijk dat beide bewoners ongeveer evenveel ruimte krijgen.



Probleem

De koning staat er op dat elk eiland verdeeld wordt door het trekken van een rechte lijn tussen twee van de hoekpunten. Onder deze voorwaarde blijkt het helaas niet altijd mogelijk te zijn om de twee gebieden exact even groot te maken.

Schrijf een programma dat, gegeven een beschrijving van de vorm van een eiland, bepaalt wat het kleinst mogelijke verschil in oppervlakte is wanneer het eiland op bovenstaande wijze in twee gebieden wordt verdeeld.

Invoer (lezen uit: `convex.in`)

Op de eerste regel staat het aantal eilanden dat gesplitst moet worden. Dan volgt per eiland:

- Een regel met het aantal hoekpunten van het eiland ($4 \leq n \leq 100$).
- Een opsomming van de n hoekpunten van het eiland, tegen de volgorde van de klok in (linksom dus). Per hoekpunt is er een regel met twee gehele getallen, gescheiden door een spatie: de coördinaten van het hoekpunt in meters ($0 \leq x, y \leq 10000$).

Uitvoer (schrijven naar standard-output)

Voor elk eiland geef je een regel met daarop het minimale verschil in oppervlakte in vierkante meters ($d \geq 0.0$). Geef alle antwoorden met precies één cijfer achter de decimale punt!

Voorbeeld

Invoer

```
2
4
0 0
2 0
2 3
1 3
5
10 1
12 1
12 4
10 3
9 2
```

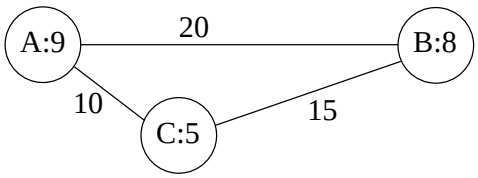
Uitvoer

```
1.5
0.0
```

C Benzine

Beschrijving

In het Verenigd Europa wordt voor alles voorschriften, richtlijnen en regeltjes opgesteld. Een van de zaken die zich echter geheel aan centrale directie lijken te onttrekken is de prijs van benzine. Verschillen in belasting maken dat de benzineprijs van land tot land, en zelfs van stad tot stad zeer uiteen kan lopen. Het gevolg is dat conventionele routeplanners niet meer voldoen aan de wensen van de calculerende automobilist. De automobilist die door Europa reist zoekt niet meer de route met de minste kilometers, maar de goedkoopste route. Dat wil zeggen dat omrijden voor goedkopere benzine acceptabel is.



De softwaregigant About besluit op deze wens in te spelen. Hun programma houdt al rekening met het rijgedrag van de individuele bestuurder, en met het gebruik van de auto. Zodoende drukt dit programma afstanden niet uit in kilometers, maar in liters benzine. Voorts zijn de benzineprijzen in alle plaatsen bekend. Waar de programmeurs van About nog niet uit zijn is het volgende: hoe bereken je nu de goedkoopste route van A naar Z, gegeven alle afstanden (in liters) en de benzineprijzen in alle plaatsen (in dE: deci-Euros).

In bovenstaand plaatje moet je om van A direct naar B te rijden, in A 20 liter tanken, tegen 9 dE, kost 180 dE. Van A naar C kost 90 dE, tanken in C voor de rit naar B kost $5 \times 15 = 75$ dE, totaal 165 dE. De omweg over C is dus goedkoper, hoewel het meer benzine kost. (Groen Links heeft de Minister dan ook verzocht dit soort programma's te verbieden).

Probleem

Gegeven is een routekaart, met de afstand tussen steden in liters, en bij elke stad de benzineprijs in die stad (in dE). Zowel de afstanden als de literprijzen zijn altijd gehele getallen. Ook zijn twee steden s_1 en s_2 gegeven. Bepaal de minimale kosten om van s_1 in s_2 te komen.

- Je begint in s_1 met een lege tank (gelukkig sta je net bij een benzinestation).
- Je mag in s_2 eindigen met een lege tank (hopelijk naast een benzinestation).
- In het Verenigd Europa hebben alle auto's een tankinhoud van 60 liter.

Invoer (lezen uit: `benzine.in`)

De eerste regel van de invoer bevat het aantal tests. Dan volgt een aantal tests. Elke test is als volgt opgebouwd:

1. De eerste regel bevat het getal s, het aantal steden.
2. Dan volgen regels met de beschrijving van de steden. Elke stad staat op een aantal regels beschreven. De eerste regel bevat achtereenvolgens:
   - De naam van de stad: een enkele hoofdletter.
   - De prijs p van een liter benzine in deze stad ($1 \leq p \leq 100$), in dE.
   - Het aantal buren b van deze stad, dat wil zeggen het aantal steden dat vanuit deze stad te bereiken is.

   De volgende b regels van een stadbeschrijving bevatten de beschrijving van de buursteden, dat wil zeggen: b maal: de naam van een buurstad, gevolgd door de afstand d ($d \leq 500$) naar b in liters benzine.
3. De laatste regel bevat de naam van twee steden: de plaats van vertrek en de plaats van bestemming, zonder scheiding er tussen.

Voor een test gelden de volgende regels:

- Voor willekeurige steden x en y, is de afstand van x naar y altijd gelijk aan de afstand van y naar x.
- Een stad die als buurstad voorkomt, komt ook als beschreven stad voor.
- De plaats van vertrek en de plaats van aankomst komen ook als beschreven stad voor.

Uitvoer (schrijven naar standard-output)

Voor elk testgeval apart op een regel: de minimale kosten (in dE) om van de plaats van vertrek in de plaats van aankomst te komen. Als de plaats van aankomst niet bereikbaar is vanuit de plaats van vertrek, het woord: '`ONMOGELIJK`'

Voorbeeld

Invoer

```
2
3
A 9 2
B 20
C 10
B 8 2
A 20
C 15
C 5 2
A 10
B 15
AB
2
A 30 1
B 100
B 30 1
A 100
AB
```

Uitvoer

```
165
ONMOGELIJK
```

D Taart-delingen voor gevorderden

Beschrijving

In groep 4 van de basisschool wordt leerlingen uitgelegd hoe delingen werken. Vaak gebeurt dit aan de hand van taartpunten. De volgende som zou voor een groep 4-ganger weinig problemen mogen opleveren:

"Jan is jarig. Hij heeft 20 taartpunten, die hij eerlijk moet delen met Kees en Piet. Hoeveel taartpunten krijgen Jan, Kees en Piet ieder?"

Een fascinerend gegeven is dat er vaak taartpunten overblijven. Zoals je wellicht inziet krijgen Jan, Kees en Piet in het bovenstaande voorbeeld ieder 6 taartpunten; er blijven 2 taartpunten over.

Probleem

In deze opgave zijn we geïnteresseerd in het aantal taartpunten r dat overblijft indien we t taartpunten delen met p personen. Met de computer kunnen we dit soort sommen natuurlijk veel sneller maken dan met pen en papier. Om de opgave toch enigzins lastig te maken, staan we (zeer) grote getallen toe voor t en p.

Invoer (lezen uit: `taart.in`)

Op de eerste regel staat een getal n, dat aangeeft hoeveel testgevallen er volgen. Op de volgende n regels staan 2 getallen p en t, gescheiden door precies één spatie.

Verder geldt dat $1 \leq p < 100.000.000$ (honderd miljoen) en dat $0 \leq t \leq 10^{1000}$

Uitvoer (schrijven naar standard-output)

Voor ieder testgeval uit de invoer moet één regel in de uitvoer komen die er als volgt uitziet:

```
Indien we T taartpunten delen met P personen, is het aantal taartpunten
dat overblijft gelijk aan R.
```

N.B.: iedere regel moet afgesloten worden met een punt!

Voorbeeld

Invoer

```
3
113 355
123456789 987654321
11447 158456325028528675187087900671
```

Uitvoer

```
Indien we 355 taartpunten delen met 113 personen, is het aantal taartpun
ten dat overblijft gelijk aan 16.
Indien we 987654321 taartpunten delen met 123456789 personen, is het aan
tal taartpunten dat overblijft gelijk aan 9.
Indien we 158456325028528675187087900671 taartpunten delen met 11447 per
sonen, is het aantal taartpunten dat overblijft gelijk aan 0.
```

E Tegelen

Beschrijving

Na het omwaaien van Mekelweg 4 werd de huisvestingproblematiek van de faculteit ITS voortvarend aangepakt. Het architectencollectief GROOTMOEDER kreeg opdracht een nieuw gebouw te ontwerpen. Ofschoon hun ontwerp op zijn minst opmerkelijk was, werd het na verrassend weinig geharrewar aangenomen en gerealiseerd.

Om hygiënische redenen werd besloten in alle ruimtes een tegelvloer te leggen. Er zijn tegels in diverse maten beschikbaar. Tegels mogen gedraaid worden. Elke zijde van elke tegel is echter een geheel aantal decimeters, hetzelfde geldt voor de te betegelen vloeren. Voor elke ruimte is vastgesteld welke tegels er gebruikt kunnen worden. In elke ruimte kan in elk geval een 1×1-tegel worden gebruikt. Alle tegels, groot of klein kosten 1 Euro per stuk (inclusief leggen). Omdat het gebouw toch al zo duur is, moet elke vloer tegen minimale kosten worden gelegd.

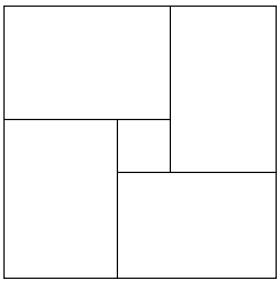
De architect heeft echter wel een esthetische eis: een betegelde vloer moet een *deelbare* rechthoek zijn.

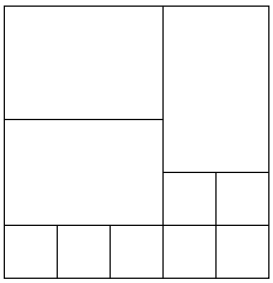
Definitie

- Een rechthoek bestaande uit één tegel is deelbaar.
- Een rechthoek bestaande uit meer dan één tegel is deelbaar, als er een rechte lijn is die, zonder een tegel te splitsen (dus geheel over voegen lopend), de rechthoek in twee kleinere rechthoeken verdeelt, die elk ook weer deelbaar zijn.

Voorbeeld

Als de vloer 5×5 is en de tegels 1×1 resp. 2×3, dan is de linker betegeling niet geoorloofd. De rechter betegeling is wel geoorloofd, maar veel duurder (tweemaal zo duur).





Probleem

Gegeven een tegelset en een aantal vloeren, bereken voor elke vloer de minimale kosten om deze vloer met deze tegelset deelbaar te beleggen. Elke tegelset bevat gegarandeerd de tegel 1×1, dus elke vloer kan belegd worden.

Invoer (lezen uit: `tegelen.in`)

De eerste regel van de invoer bevat het aantal tests. Dan volgt een aantal tests. Elke test is als volgt opgebouwd:

1. De eerste regel bevat het getal t, het aantal tegels.
2. De volgende regel bevat de beschrijving van t tegels. In elk geval is de tegel 1×1 aanwezig. Alle tegels zijn verschillend. Een tegelbeschrijving bestaat uit twee getallen $b\ l$, met $0 < b \leq l < 100$.
3. De volgende regel bevat het getal v, het aantal vloeren ($v < 100$).
4. De volgende regel bevat de beschrijving van v vloeren. Een beschrijving van een vloer bestaat uit twee getallen $b\ l$, met $0 < b \leq l < 100$.

De getallen op één regel worden telkens door één of meer spaties gescheiden.

Uitvoer (schrijven naar standard-output)

De uitvoer bevat, voor elk testgeval apart op een regel, de minimale kosten van betegeling van de vloeren.

Voorbeeld

Invoer

```
2
2
1 1   2 3
3
4 4   5 5   6 6
3
1 1   1 10   3 5
3
4 4   5 6   4 10
```

Uitvoer

```
6 10 6
16 2 3
```

F Sommen

Beschrijving

De kinderen in de klas van meester Van Dalen zijn erg goed in rekenen. Ze kunnen optellen, aftrekken, vermenigvuldigen en delen. Bovendien maken ze al sommen waarin meerdere rekenoperaties en zelfs haakjes voorkomen. De meester laat ze vaak oefenen met een computerprogramma. Het programma verzint dan sommen, en de leerlingen moeten het antwoord intypen. Na afloop kunnen alle sommen opgeslagen en nagekeken worden.

Helaas is er bij het opslaan van de gemaakte sommen iets mis gegaan: door een foutje in het programma zijn alle haakjes weggevallen. Het nakijken van de sommen wordt nu een stuk lastiger omdat niemand nog weet waar de haakjes in de som precies stonden.

Probleem

De meester besluit dat een som goed gerekend wordt als het mogelijk is om haakjes te plaatsen zodat de som weer klopt; als dat niet mogelijk is wordt de som fout gerekend.

Daarbij houdt hij rekening met de beperkingen in de sommen die het programma kon verzinnen. Alle getallen, tussenresultaten en eindresultaten in de oorspronkelijke sommen waren gehele getallen met $0 \leq g \leq 1000000$ (één miljoen). Verder bestaat elke som uit maximaal 10 getallen. Delen kan alleen maar als de uitkomst een geheel getal is; er wordt niet doorgerekend met breuken en er wordt niet afgerond.

Het is natuurlijk een heel gepuzzel om met de hand uit te zoeken of een som nog goed gerekend kan worden. Het zou dus handig zijn als jij daar een computerprogramma voor maakt.

Invoer (lezen uit: `sommen.in`)

Op de eerste regel staat het aantal sommen dat moet worden nagekeken.
Dan volgt per som een regel met daarop de som en het antwoord van de leerling, in het volgende formaat: x_1 o_1 x_2 o_2 ... o_{n-1} x_n `=` a
Waarbij de x_i ($0 \leq x_i \leq 1000000$, $1 \leq i \leq n$) de getallen in de som zijn, en de o_i ($1 \leq i \leq n-1$) telkens één van de karakters '`+`', '`-`', '`*`' of '`/`' voor optellen, aftrekken, vermenigvuldigen en delen. Het getal a ($0 \leq a \leq 1000000$) is het antwoord dat de leerling heeft gevonden. Verder geldt dat $1 \leq n \leq 10$. De getallen en operatoren zijn van elkaar gescheiden door een spatie.

Uitvoer (schrijven naar standard-output)

De uitvoer bestaat uit een regel per nagekeken som met daarop het woord '`goed`' of '`fout`'.

Voorbeeld

Invoer

```
6
10 - 5 - 3 = 2
10 - 5 - 3 = 7
10 - 5 - 3 = 8
10 - 3 - 5 = 12
65535 + 1 * 65537 - 1 = 131071
65535 + 1 * 65537 - 1 = 0
```

Uitvoer

```
goed
fout
goed
fout
goed
fout
```

G Feest 2000

Beschrijving

Er wordt een grandioos feest georganiseerd om Nieuwjaar 2000 te vieren. Uiteraard wil iedereen handen schudden met iedere andere gast. Om dit soepel en efficient te laten verlopen vindt het handenschudden in een aantal rondes plaats.

Probleem

Schrijf een programma dat een schema opstelt waarin staat wie in elke ronde met wie handen schudt, zodanig dat het aantal rondes minimaal is.

Verder moet het schema aan de volgende eisen voldoen:

- In elke ronde kan iedere gast van maximaal één andere gast de hand schudden.
- Na afloop van de laatste ronde moet iedere gast van iedere andere gast precies éénmaal de hand geschud hebben.

In het algemeen zijn er meerdere correcte schema's mogelijk. Het maakt dan niet uit welk schema je geeft, zolang het maar aan bovenstaande eisen voldoet.

Invoer (lezen uit: `feest.in`)

De eerste regel bevat een getal r, het aantal testgevallen. Op de volgende r regels staat telkens een getal n ($1 \leq n \leq 100$), het aantal gasten bij dat testgeval. De gasten zijn genummerd van $0 \ldots n-1$.

Uitvoer (schrijven naar standard-output)

Per testgeval bevat de uitvoer als eerste een regel met daarop een getal k ($0 \leq k$), het aantal rondes voor het schema van dat testgeval. Daarna volgen k regels met op iedere regel een aantal paren a_{2i}-a_{2i+1}, met $0 \leq a_i < n$ en alle a_i verschillend; dit zijn de gasten die elkaar de handen schudden in die ronde. Tussen de nummers van gasten die elkaar de hand schudden staat alleen een '-'. De verschillende gast-paren zijn van elkaar gescheiden door telkens één spatie.

Voorbeeld

Invoer

```
2
3
6
```

Uitvoer

```
3
0-1
1-2
0-2
5
0-1 2-3 4-5
1-2 3-4 5-0
0-3 1-5 2-4
1-4 0-2 3-5
2-5 1-3 0-4
```