A DJ SWAPPER

De automatisering slaat nu ook toe in de wereld van de radio. Jeroen van Inkel wordt binnenkort uitkeringstrekker. Zijn plaats zal namelijk ingenomen worden door de volautomatische robot DJ Swapper. Robotische DJs zijn namelijk goedkoper en ze hebben niet de neiging op een irritante manier door de gedraaide muziek heen te praten.

Tijdens het testen van DJ Swapper bleek echter dat de robot niet helemaal bugvrij is. Aan het eind van een lange dag muziek draaien blijken er namelijk veel gedraaide CDs in het verkeerde doosje te zijn terechtgekomen. Deze bug is echter snel gefixt, door de DJ aan het eind van de dag de CDs weer in de juiste doosjes te laten doen. Hiervoor is een eenvoudig sorteeralgoritme gebuikt, genaamd swapsort. Zo komt de automatische DJ dan ook aan zijn naam.

Hoe werkt swapsort? Als alle CDs in de goede doosjes liggen, is DJ Swapper klaar. Zolang dat niet het geval is, kiest hij een willekeurige CD, met nummer X (om het de robot makkelijk te maken, hebben alle CDs en doosjes een nummer gekregen), die in het doosje van een andere CD Y ligt. Dit schijfje X haalt DJ Swapper met zijn ene hand uit het doosje van Y. Vervolgens zoekt hij het doosje van X, waar een CD Z in zit. Hij pakt deze CD Z in zijn andere hand, en swapt nu deze CDs. Het resultaat van deze stap is dus, dat CD X in het juiste doosje zit, en CD Z in doosje Y zit.

Jeroen's baas wil echter graag weten, of dit sorteerproces niet te lang duurt. Tijd is immers geld. Hij is er dus in geïnteresseerd, na hoeveel swap-stappen alle CDs in de juiste doosjes terecht zullen komen, gegeven een bepaalde beginsituatie. Omdat Jeroen van Inkel beter is in plaatjes draaien dan in programmeren, wordt aan jou gevraagd om dit uit te zoeken.

Invoer

De invoer bevat op de eerste regel het aantal runs. Elke run bestaat uit 2 regels. De eerste regel bevat een getal N, dat het aantal CDs weergeeft. Er zijn maximaal 100 CDs. De CDs zijn genummerd van 1 t/m N. De volgende regel bevat N getallen P_1 t/m P_n. Het getal P_i geeft aan dat de CD met nummer P_i in het doosje met nummer i zit, voordat er gesorteerd is.

Uitvoer

De uitvoer bevat voor elke run een regel met de volgende string:

`Aantal benodigde stappen: `*`N`*

waarbij N vervangen moet worden door het aantal swap-stappen dat voor die run moet worden uitgevoerd om alle CDs volgens het swapsort-algoritme te ordenen.

Voorbeeldinvoer

```
2
2
2 1
3
3 1 2
```

Voorbeelduitvoer

```
Aantal benodigde stappen: 1
Aantal benodigde stappen: 2
```

B NIET-NULLEN

Men neme de representatie in het N-tallig stelsel van de getallen 0 tot en met M (dit noteren we als 0..M). Bijvoorbeeld, voor $N = 3$ en $M = 10$, zie de eerste twee kolommen van onderstaande tabel.

Getal	N-tallige ontwikkeling	Met nullen	# niet-nullen	benadering
0	0	000	0	0
1	1	001	1	1
2	2	002	2	2
3	10	010	3	5
4	11	011	5	6
5	12	012	7	8
6	20	020	8	9
7	21	021	10	10
8	22	022	12	12
9	100	100	13	20
10	101	101	15	22

Het antwoord op de vraag, hoeveel cijfers ongelijk aan 0 de N-tallige ontwikkelingen van deze getallen in totaal bevatten, kan in veel gevallen aardig worden *benaderd* door de formule $\lfloor (M + 1) \times |M| \times (N - 1)/N \rfloor$ (zie kolom 5), waarbij $|M|$ het aantal cijfers is waaruit M bestaat in het N-tallig stelsel.
Dit is gebaseerd op de waarneming, dat alle cijfers 0..N ongeveer even vaak voorkomen; het totaal aantal cijfers in de M+1 getallen 0..M (waarbij getallen bestaande uit minder cijfers dan $|M|$ met nullen worden opgevuld, zie kolom 3) is (M+1)*$|M|$ en N-1 van de N mogelijke cijfers zijn niet-nul. Omdat de onafgeronde schatting vaak wat aan de hoge kant is, wordt deze naar beneden afgerond voor een beter resultaat.
Echter, het het *exacte* antwoord op deze vraag (zie kolom 4) kan niet via deze eenvoudige weg worden verkregen. Aan jou nu de taak een programma te schrijven, dat voor een gegeven M en N, het totale aantal niet-nullen in de N-tallige ontwikkeling van de getallen 0..M berekent.

Invoer

De eerste regel van de invoer bevat een getal k. Er volgen nu k regels, elk een testcase bevattend. Deze regels bevatten elke twee getallen M ($0 \leq M \leq 10000000$) en N ($2 \leq N \leq 50$).

Uitvoer

De uitvoer moet voor iedere testcase bestaan uit de volgende regel:

`Het aantal niet-nullen is X.`

Waarbij X (decimaal afgedrukt) het exacte aantal niet-nullen in de N-tallige representatie van de getallen $0..M$ is.

Voorbeeldinvoer

```
1
10 3
```

Voorbeelduitvoer

```
Het aantal niet-nullen is 15.
```

C PEUKEN

Roken is slecht voor je, en kettingroken al helemaal. Toch zijn er veel mensen die het doen. Zo ook de held van deze opgave, Piet. Piet rookte ketting, en dat was niet goed voor hem. Al die sigaretten kostten hem handenvol geld. Zijn vrienden vonden hem stinken. Zijn vrouw vond het vervelend dat het echtelijk bed steeds in brand vloog als Piet tijdens het roken van een nachtelijke sigaret weer eens in slaap was gevallen. Zijn baas werd steeds boos op hem omdat hij tijdens zijn werk voortdurend pauze nam om te roken.

Zo verloor hij zijn vrouw, zijn vrienden, zijn werk en zijn geld. Uiteindelijk verloor Piet ook zijn huis, toen dit afbrandde door een onachtzaam weggesmeten smeulende peuk. Zo belandde hij in de goot en werd hij zwerver.

Als zwerver had hij natuurlijk geen geld meer om sigaretten te kopen. En hij was te eerlijk om ze te stelen. Daarom ging hij door mensen weggeworpen peuken verzamelen.
Immers:
- als je een sigaret rookt hou je een peuk over;
- 7 peuken zijn precies even lang als een sigaret.

Na enige oefening werd Piet er behendig in, 7 peuken weer aan elkaar vast te maken tot een sigaret. Zo kon hij toch weer roken.

De vraag is nu: hoeveel sigaretten kan Piet op een dag roken als hij die dag N peuken verzameld heeft?

Invoer

De eerste regel van de invoer bevat een getal M. Er volgen nu M regels, elk een testcase bevattend. Een testcase bestaat uit een enkel getal N ($0 \leq N$), het aantal peuken dat Piet verzameld heeft.

Uitvoer

Voor iedere testcase moet de volgende regel afgedrukt worden:

`Van` *`N`* `peuken kunnen` *`K`* `sigaretten gerookt worden.`

Waarbij K het aantal sigaretten is dat van N peuken gerookt kan worden.

Voorbeeldinvoer

```
2
6
12
```

Voorbeelduitvoer

```
Van 6 peuken kunnen 0 sigaretten gerookt worden.
Van 12 peuken kunnen 1 sigaretten gerookt worden.
```

D CIJFERS EN LETTERS

Deze opgave gaat niet over het bekende mathematisch-linguistische spelprogramma met dezelfde naam, maar over het omzetten van nederlandse telwoorden naar hun representatie in het tientallig getalstelsel. De opdracht is, een correct nederlands telwoord in te lezen, en het bijbehorende getal P ($0 \leq P \leq 1000$) uit te printen.

Invoer

De eerste regel van de invoer bestaat uit een getal N. Hierna volgen N regels met ieder een testgeval. Ieder testgeval bestaat uit een string S.

Uitvoer

Voor ieder testgeval bestaat de uitvoer uit de volgende regel:

S = P

waarbij S en P vervangen moeten worden door de ingelezen string S en het bijbehorende getal P.

Voorbeeldinvoer

```
9
een
elf
tachtig
drieentachtig
zevenennegentig
honderdnegentien
tweehonderdtweeentwintig
zeshonderdvijfenzeventig
negenhonderdeen
```

Voorbeelduitvoer

```
een = 1
elf = 11
tachtig = 80
drieentachtig = 83
zevenennegentig = 97
honderdnegentien = 119
tweehonderdtweeentwintig = 222
zeshonderdvijfenzeventig = 675
negenhonderdeen = 901
```

E DFA

In de informatica spelen eindige automaten een belangrijke rol. Zij worden gebruikt bij vele theoretische en praktische toepassingen. Een speciaal type eindige automaten zijn de deterministic finite acceptors.

Een deterministic finite acceptor (dfa) kan worden gedefinieerd door M = (Q, Σ, δ, q_0, F), met:
Q is een eindige verzameling van toestanden;
Σ is een eindige verzameling van symbolen, het input alphabet;
δ: $Q \times \Sigma$ -> Q is een totale functie, de transitiefunctie;
q_0 (element van Q) is de initiële toestand;
F (deelverzameling van Q) is de verzameling eindtoestanden.

Een dfa werkt als volgt. In het begin is de dfa in toestand q_0 en wordt een bepaalde invoerstring aangeboden. De dfa leest het eerste symbool van de invoerstring (een string is een (eventueel lege), eindige lijst van symbolen uit Σ). Op basis van de functie δ met als parameters de huidige toestand en het huidige invoersymbool gaat de dfa naar een nieuwe toestand, waarna het het volgende invoersymbool leest.
Dit herhaalt zich totdat de hele invoerstring van links naar rechts is gelezen. De invoerstring is nu geaccepteerd dan en slechts dan als de dfa zich in een van de eindtoestanden bevindt.

De taal die door een dfa wordt geaccepteerd is de verzameling van alle strings die geaccepteerd worden. Een prefix van lengte n van een string w, met $0 \leq n \leq |w|$ (de lengte van w), is de string die bestaat uit de eerste n symbolen van w (in dezelfde volgorde).

De opdracht is nu, de transitiefunctie δ te bepalen voor een dfa die aan de volgende eisen voldoet:
1. de dfa accepteert de taal die bestaat uit de strings xwy, waarbij w gegeven is en x en y willekeurige strings zijn en $1 \leq |w| \leq 20$;
2. $|Q|$ (het aantal elementen van Q) is minimaal, dwz. gelijk aan $|w|+1$;
   deze toestanden zijn genummerd van 0 t/m $|w|$;
3. Σ = { ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘h’ };
4. de dfa bevindt zich in toestand i ($0 \leq i \leq |w|$) als de string die gevormd wordt door de laatste i gelezen invoersymbolen gelijk is aan de prefix van w van lengte i.

Hieruit volgt dat:
- q_0 gelijk is aan de toestand genummerd 0;
- het enige element dat F bevat, gelijk is aan de toestand genummerd $|w|$.

Invoer

De eerste regel van de invoerfile bevat een getal *N*, het aantal testgevallen. Hierna volgen *N* regels die ieder een string *w* bevatten. De te accepteren taal moet gelijk zijn aan *xwy*, waarbij *x* en *y* willekeurige strings zijn.

Uitvoer

Voor ieder testgeval moet de volgende uitvoer gegenereerd worden;

- de eerste regel van de uitvoer bevat de string '`specificatie voor "`*w*`"`', waarbij *w* vervangen moet worden door de *w* uit de invoer;
- de tweede regel bevat de string '`  a  b  c  d  e  f  g  h`', waarbij '`a`' vooraf gegaan wordt door 4 spaties, hierna wordt ieder opeenvolgend paar letters (bijvoorbeeld '`e`' en '`f`') gescheiden door 2 spaties;
- de 3e t/m $|w|$+3de regel zijn genummerd van 0 t/m $|w|$; dit getal staat rechts uitgelijnd in een veld van lengte 2 dat begint in kolom 1; op deze regels volgen nog 8 getallen, rechts uitgelijnd in een veld van lengte 3 dat voor het *i*-de getal begint in kolom $i*3$ (dus recht onder de *i*-de letter op de tweede regel). Deze getallen vormen tesamen de δ van de dfa: de functiewaarde van $\delta(t, s)$, $0 \le t \le |w|$, $s \in \Sigma$, staat op de regel die genummerd is met *t*, en is daar het getal dat recht onder de letter *s* uit de tweede regel staat;
- de uitvoer voor een testgeval wordt afgesloten door een lege regel.

Voorbeeldinvoer

```
1
abc
```

Voorbeelduitvoer

```
specificatie voor "abc"
    a  b  c  d  e  f  g  h
 0  1  0  0  0  0  0  0  0
 1  1  2  0  0  0  0  0  0
 2  1  0  3  0  0  0  0  0
 3  3  3  3  3  3  3  3  3
```

F ZAKJAPANNER

Je grootste ontdekking tijdens de wiskundeles op school is ongetwijfeld geweest, dat een getal dat wordt afgebeeld op het display van je zakjapanner een woord kan vormen als je dit apparaatje omkeert. Bijvoorbeeld, het getal '73083734' wordt het woord 'hELEBOEL' als het ondersteboven gelezen wordt. Voor bijna alle cijfers is er een corresponderende letter, volgens onderstaande tabel:

Cijfer	Letter
0	O
1	I
2	Z
3	E
4	h
5	S
6	g
7	L
8	B

Het cijfer 9 ontbreekt in bovenstaande tabel. Als dit cijfer omgekeerd gelezen wordt, is het nog steeds gewoon een omgekeerde 9, en geen letter. Getallen die een of meerdere negens bevatten vormen dus omgekeerd geen woord.

Met de woorden opgebouwd uit de omgekeerde cijfers 0 t/m 8 kunnen berekeningen worden uitgevoerd. De opgave is nu, om gegeven een berekening bestaande uit woorden (die omgekeerde getallen representeren) en operatoren, het woord te geven dat het omgekeerde is van het resultaat van de uitgevoerde berekening.

We hebben hier te maken met een vrij primitieve zakjapanner. Er zijn slechts vier operanden: +, -, *, /, die alle een even hoge prioriteit hebben en links-associatief zijn, d.w.z. zo gauw als een operator en een operand worden ingevoerd in de zakjapanner, wordt een berekening uitgevoerd met het resultaat van de vorige berekening als linker operand van de operator en het nieuw ingevoerde getal als rechter operand. De uitkomst van de berekening wordt vervolgens weer gebruikt als linker operand van de volgende operator, zo die er is, etc.
Verder mag je er vanuit gaan dat:

- alle ingevoerde getallen kleiner zijn dan 30.000;
- het resultaat van elke berekening (evenals elk tussenresultaat) kleiner is dan 1.000.000.000;
- voor delingen geldt: als $A/B = C$, dan is C een geheel getal en $B*C=A$;
- er worden geen negatieve getallen gebruikt, noch is het resultaat van een berekening (of tussenresultaat), negatief.

Invoer

De invoerfile bevat op de eerste regel een getal *N*, dat het aantal testgevallen weergeeft. Daarna volgen *N* regels, die elk een berekening bevatten. Een berekening bestaat uit een woord (een niet-lege string bestaande uit karakters uit de verzameling {'O', 'I', 'Z', 'E', 'h', 'S', 'g', 'L', 'B'}), nul of meer keer gevolgd door een operator uit de verzameling {'+', '*', '-', '/'} en een woord. De berekening wordt afgesloten door het karakter '='. De berekeningen bevatten geen andere karakters.

Uitvoer

De uitvoer voor elke berekening bestaat uit de string '`het resultaat is: `', gevolgd door het woord dat (omgekeerd) het resultaat is van de berekening. Als er geen woord gevormd kan worden van het resultaat, moet deze string gevolgd worden door de string '`fout!`'.

Voorbeeldinvoer

```
2
LOL+LOL=
OI-I=
```

Voorbeelduitvoer

```
het resultaat is: hIhI
het resultaat is: fout!
```

G EINDSTAND

De organisatoren van het NKP van dit jaar kunnen zelf niet programmeren! Ze zijn wanhopig op zoek naar iemand die een programma voor ze kan schrijven, dat de eindstand van het NKP kan uitprinten. Misschien kun jij ze helpen.

Dit programma moet eerst voor elke opgave van de wedstrijd de letter en de naam inlezen en voor ieder deelnemend team het teamnummer en de naam. Verder heeft de jury tijdens de wedstrijd de gegevens van alle opgaven die ingeleverd zijn verzameld. De gegevens van een ingeleverde opgave bestaan uit een teamnummer, de verstreken tijd in minuten vanaf het begin van de wedstrijd, de letter van de ingeleverde opgave en het antwoord van de jury.

Zoals bekend eindigt een team met nummer *X* hoger dan een team met nummer *Y* als:
- team *X* meer opgaven goedgekeurd heeft gekregen dan *Y*, of
- team *X* en team *Y* evenveel opgaven goedgekeurd hebben gekregen en team *X* heeft een lagere totaaltijd dan team *Y*.

De totaaltijd van een team is de som van de inlevertijden van al hun goedgekeurde opgaven plus 20 minuten straftijd voor elke afgekeurde inleverpoging van een opgave die later goedgekeurd is. Merk op dat alle inleverpogingen voor een opgave gedaan na de eerste goedgekeurde poging voor die opgave niet meer in meetellen voor de inlevertijd (voor een goedgekeurde poging) dan wel straftijd (voor een afgekeurde poging).

Op basis van deze gegevens moet je programma in staat zijn om de eindstand van een wedstrijd uit te printen in het formaat dat hieronder beschreven is.

Invoer

De eerste regel van de invoer bevat een getal *N*. Er volgen *N* testgevallen.

Ieder testgeval begint met een regel die drie getallen *P*, *T* en *S* bevat. De volgende *P* regels bevatten de beschrijvingen van de *P* opgaves. Deze regels bevatten als eerste karakter de letter van de opgave gevolgd door een spatie, de naam van de opgave en een return.

De volgende *T* regels bevatten de beschrijvingen van de *T* deelnemende teams. Deze regels bevatten een teamnummer, een spatie en de naam van het team, gevolgd door een return.

Tenslotte bevatten de *S* volgende regels de gegevens van de jury over de *S* inleverpogingen tijdens de wedstrijd. Deze regels bestaan uit het nummer van het team dat de inleverpoging heeft gedaan, de inlevertijd in minuten vanaf het begin van de wedstrijd, de letter van de opgave die is ingeleverd, het resultaat van de jury (‘`Goed!`’ of ‘`Fout`’).

Je mag verder aannemen dat:
- $1 \leq P \leq 8$;
- $1 \leq T \leq 30$;
- $S \geq 0$;

- alle letters van opgaven zijn uniek en uit het domein '`A`..'`Z`';
- alle teamnummers zijn uniek en uit het domein 1..99;
- de namen van opgaven en teams hebben een lengte van minimaal 1 en maximaal 16 karakters;
- de namen van opgaven en teams beginnen en eindigen met een karakter dat geen spatie is;
- de teamnummers en opgaveletters in de gegevens van de jury zijn bestaande teamnummers respectievelijk opgaveletters;
- de inlevertijden liggen tussen 0 en 299 minuten inclusief;
- de gegevens van de jury zijn gesorteerd op inlevertijd;
- als een team een inleverpoging doet voor een opgave die voor dat team al is goedgekeurd, moet deze poging volledig genegeerd worden voor de einduitslag;
- per team zijn er hoogstens 10 inleverpogingen per opgave;
- Teams eindigen niet met hetzelfde aantal opgaven plus dezelfde totaaltijd.

Uitvoer

Voor ieder testgeval bestaat de uitvoer uit een eindstand bestaande uit een header van P+1 regels, gevolgd door T regels die de eigenlijke stand bevatten. Een eindstand moet gevolgd worden door een lege regel.

De eerste P regels van de header bevatten de namen van de P opgaves, elk op een aparte regel. De naam van de N-de opgave in de invoer voor dit testgeval moet zich op de N-de regel van de header bevinden, beginnend in kolom 15+N∗9. De naam van iedere opgave (behalve de laatste) is verbonden met zijn letter op de P+1-ste regel van de header door een vertikale lijn bestaande uit een of meer '|'karakters (voor opgave N bevatten de regels N+1 t/m P in de 15+N∗9-de kolom een '|').

De P+1-ste regel van de header bevat de string '`Team`' beginnende in kolom 4. Verder bevat het de letter van de eerste opgave in kolom 24, de letter van de tweede opgave (zo die aanwezig is) in kolom 33, enzovoorts. De letter van de N-de opgave verschijnt dus in kolom 15+N∗9 op deze regel. Tenslotte bevat deze regel de strings '`#Solved`' en '`Tijd`', beginnend in respectievelijk kolom 22+P∗9 en 30+P∗9.

De N-de van de T regels van de eigenlijke stand bevat de informatie van het team dat op de N-de plaats is geëindigd. Zo'n regel bevat het getal N beginnend in kolom 1, rechts uitgelijnd in een veld van lengte 2. Verder bevat het de naam van het team beginnend in kolom 4, links uitgelijnd in een veld van lengte 16. Ook bevat het, voor iedere opgave, de inlevertijd van de goedgekeurde poging voor die opgave of de string '`---`' als deze opgave niet door dit team is opgelost, beginnend in kolom 13+M∗9 voor de M-de opgave, rechts uitgelijnd in een veld van lengte 3. Verder, bevat de regel, als het team tenminste een inleverpoging heeft gedaan voor deze opgave (hetzij goedgekeurd, hetzij afgekeurd), de string '(x)' beginnend in kolom 17+M∗9, waarbij de x vervangen moet worden door het aantal inleverpogingen. Tenslotte bevat de regel in kolom 23+P∗9 het aantal opgaven van dit team dat goedgekeurd is en beginnend in kolom 30+P∗9 de totaaltijd van het team, rechts uitgelijnd in een veld van lengte 4.

Voorbeeldinvoer

```
1
3 3 6
A Opgave 1
B Opgave 2
C Opgave 3
1 The geeks
2 The nerds
3 The super-users
1 50 A Goed!
3 83 B Fout
1 85 B Fout
3 105 B Goed!
1 240 B Goed!
2 284 C Fout
```

Voorbeelduitvoer

```
                      Opgave 1
                      |        Opgave 2
                      |        |        Opgave 3
   Team               A        B        C       #Solved Time
 1 The geeks          50 (1)   240 (2)  ---        2       310
 2 The super-users    ---      105 (2)  ---        1       125
 3 The nerds          ---      ---      --- (1)   0         0
```

H DIERENTUIN

Dierentuin ‘De Aap’ maakt plannen voor een nieuwe safari-tuin, waarin een aantal wilde roofdieren ondergebracht moeten worden. Er moet echter met allerlei regels rekening gehouden worden bij het ontwerp van de tuin. Bijvoorbeeld: er mogen niet teveel dieren van een bepaald soort in, want dan gaan ze met elkaar vechten, maar ook weer niet te weinig, want dan worden ze eenzaam, of: er moeten niet te weinig dieren in de tuin, want dat stellen de toeschouwers niet op prijs, maar het mag ook weer niet te gek worden, want dan hebben de dieren geen bewegingsvrijheid.

Bij nader inzien blijken er twee groepen van eisen te zijn: zogenaamde ‘harde’ en ‘zachte’. Aan harde eisen moet absoluut worden voldaan, het zou fijn zijn om aan zachte eisen te voldoen. Wanneer de ontwerpers van de tuin zien wat voor lastige eisen er allemaal zijn en dat deze ook nog in twee groepen te verdelen zijn zakt de moed hen in de schoenen; ze hebben geen idee hoe ze dit probleem op moeten lossen. Daarom besluiten ze tot de volgende vereenvoudiging van het probleem: het aantal harde constraints wordt tot een minimum beperkt, en de zachte constraints worden als gelijkwaardig beschouwd. Dus een oplossing die aan de harde constraints voldoet en bovendien aan zoveel mogelijk zachte constraints zal gekozen worden.

Alle constraints zijn een lineaire combinatie van aantallen dieren van een bepaald soort, dus $a_1x_1 + a_2x_2 + ... + a_nx_n \leq C$ of $a_1x_1 + a_2x_2 + ... + a_nx_n \geq C$, met:
- a_i zijn de parameters van de constraint;
- x_i zijn de aantallen gekozen dieren van soort i;
- C is een constante ($i = 1..n$);
- n is het aantal soorten dieren waaruit gekozen mag worden.

De harde constraints zijn:
- Het aantal dieren van een soort is minimaal 0 en maximaal 4;
- Het totale aantal dieren in de tuin is minimaal 0 en maximaal 12.

Het probleem blijkt echter nog steeds te lastig. Daarom wordt jullie hulp ingeroepen. Bepaal voor een bepaalde set van (zachte) eisen het minimale aantal waaraan niet kan worden voldaan, gegeven de twee bovenstaande harde eisen.

Invoer

De invoer begint met een regel met een geheel getal R, dit is het aantal runs dat volgt. Daarna volgt R keer een run die bestaat uit:
- Een regel met een geheel getal D die het aantal soorten dieren geeft waaruit gekozen kan worden ($1 \leq D \leq 6$);
- Een regel met een geheel getal N die het aantal ‘minimum’ constraints aangeeft ($0 \leq N \leq 20$);
- Daarna volgen N regels met de minimum constraints. Een minimum constraint bestaat uit D+1 gehele getallen. Bijvoorbeeld, wanneer $D = 3$, dan zou een regel die bestaat uit de

getallen ‘2 0 3 5’ staan voor de constraint: 2 ∗ ‘Het aantal dieren van type 1’ + 0 ∗ ‘Het aantal dieren van type 2’ + 3 ∗ ‘Het aantal dieren van type 3’ ≥ 5 (moet minimaal 5 zijn);
- Een regel met een geheel getal X die het aantal ‘maximum’ constraints aangeeft ($0 \leq X \leq 20$);
- Daarna volgen X regels met maximum constraints. Een maximum constraint ziet er hetzelfde uit als een minimum constraint, maar het teken is ‘≤’, dus een lineaire combinatie van de aantallen dieren mag maximaal een bepaald getal zijn.

Uitvoer

Voor elke run moet de uitvoer bestaan uit een regel met die begint met de tekst

```
Het minimale aantal constraints waaraan niet kan worden voldaan is
```

gevolgd door een spatie en een geheel getal dat het minimale aantal constraints geeft waaraan niet voldaan kan worden wanneer de dieren in een gunstige hoeveelheid gekozen worden.

Voorbeeldinvoer

```
1
3
2
1 1 1 8
2 0 1 6
3
2 1 0 10
0 1 1 4
1 0 1 5
```

Voorbeelduitvoer

```
Het minimale aantal constraints waaraan niet kan worden voldaan is 1
```