Opgave A - Kwadraten

Kwadrateren van grote gehele getallen is vaak een lastige klus, zeker zonder hulp van een computer. Jullie hebben een computer tot jullie beschikking, dus voor jullie hoeft het niet lastig te zijn! Jullie taak is dan ook een gegeven niet negatief geheel getal (dat uit maximaal 80 cijfers bestaat) te kwadrateren.

Invoer

De eerste regel van de invoer geeft het aantal runs. Elke volgende regel beschrijft een run en bestaat uit een aaneengesloten rij cijfers (die het te kwadrateren getal weergeeft) en wordt afgesloten met een return. Op een regel komen dus slechts de cijfers 0-9 en het end-of-line teken voor.

Uitvoer

Geef per run een regel als uitvoer en wel het kwadraat van het ingelezen getal, direct gevolgd door een end-of-line karakter. Dit getal mag niet worden voorafgegaan door nullen.

Voorbeeld Invoer

```
3
7
16
1234567
```

Voorbeeld Uitvoer

```
49
256
1524155677489
```

Opgave B - Wandelroute

De Koninklijke Nederlandse Wandel Bond (KNWB) wil in verschillende regio's en over verschillende wegen, wandeltochten uitzetten. Ze krijgen in de diverse regio's een beperkt aantal wegen toegewezen die lopen tussen twee punten A en B. Van deze wegen mag niet worden afgeweken in verband met de veiligheid voor de wandelaars. Nu is de totale lengte van de wegen dusdanig kort, dat besloten is om elke wandeltocht langs alle wegen in een bepaalde regio te laten lopen.

Er moet dus voor elke regio gekeken worden of het mogelijk is om een route uit te zetten, die langs alle toegewezen wegen gaat. Extra voorwaarde is dat er nooit twee maal over een gelijke weg mag worden gelopen. Het liefst heeft men een gesloten route, een circuit. Maar als dit niet kan dan mag het ook een etappe worden. Dit is een route langs alle wegen, waarbij begin- en eindpunt verschillen. Als dit niet kan dan vervalt de route in de desbetreffende regio. Een toegewezen weg is altijd te bereiken vanuit elk punt.

Invoer

Het formaat voor de invoer is als volgt. Op de eerste regel het aantal regio's z. Vervolgens komen de regio's. Deze beginnen met x, y. x Is het aantal punten in de regio (< 50), y is het aantal toegewezen wegen. Deze wegen zijn bidirectioneel. Vervolgens een lege regel. En tenslotte een aantal regels met daarop maximaal 5 wegen per regel. De scheiding tussen twee regio's bestaat uit een lege regel.

Uitvoer

De uitvoer bestaat uit z regels. Op elke regel komt te staan of voor de betreffende regio een route bestaat, en wat voor soort dit dan is. De mogelijke antwoorden zijn: `etappe`, `circuit`, `niet mogelijk`.

Voorbeeld Invoer

```
2
10,20

1,2 1,3 1,4 1,5 1,6
1,7 1,8 1,9 1,10 2,3
2,4 2,5 2,6 2,7 2,8
```

```
2,9 2,10 3,4 4,5 5,6

3,3

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

Voorbeeld Uitvoer

```
niet mogelijk
circuit
```

Opgave C - Blokjes

Er zijn N blokjes gegeven. De vraag is op hoeveel verschillende manieren je deze kunt rangschikken. Twee rangschikkingen zijn verschillend als ze niet door draaiing of spiegeling in elkaar over kunnen gaan.

De enige rangschikking met twee blokjes is bijvoorbeeld deze:

```
+--+--+
|  |  |
+--+--+
```

Invoer

De invoerfile bevat op de eerste regel het aantal runs n. Verder bevat de file n regels die elk een getal tussen 1 en 9 inclusief bevatten.

Uitvoer

De uitvoer bestaat uit n regels, die elk het corresponderende aantal rangschikkingen bevat.

Voorbeeld Invoer

```
1
3
```

Voorbeeld Uitvoer

```
2
```

Opgave D - DNA–Technologie

Zoals bekend zijn de laatste tijd grote vorderingen gemaakt in de biochemie. Van veel DNA-moleculen is de structuur bekend, en het is al mogelijk dit soort moleculen te synthetiseren uit kleinere fragmenten van DNA. Het door jullie te ontwerpen programma zal de onderzoekers kunnen helpen bij deze synthese. Gegeven wordt een aantal fragmenten DNA, dwz een aantal strings bestaande uit combinaties van de letters 'C','A','G' en 'T'. Daarna het te produceren eiwitmolecuul, eveneens een combinatie van de letters 'C', 'A', 'G' en 'T'. Gevraagd wordt op hoeveel manieren deze string gemaakt kan worden door van links naar rechts brokstukken van de gegeven vorm aan elkaar te plakken.

Voorbeeld:
Brokstukken: A, AC, CA, CAC.
Te maken: CACACACA.
Dit kan op 8 manieren, namelijk
CA:CA:CA:CA
CA:CA:CAC:A
CA:CAC:A:CA
CA:CAC:AC:A
CAC:A:CA:CA
CAC:AC:A:CA
CAC:AC:AC:A

Invoer

Op de eerste regel staat het aantal testgevallen. Op de volgende regels voor ieder testgeval:

- op een regel een positief geheel getal (N) dat niet groter is dan 100.
- vervolgens N regels waarop een geheel getal (k) uit 1..10, gevolgd door een spatie en een fragment, bestaande uit k letters uit de collectie C,A,G,T,
- de laatste regel voor ieder testgeval bestaat uit een geheel getal (g) niet groter dan 100, een spatie, en vervolgens een string van g letters uit de collectie C,A,G,T.

Uitvoer

De uitvoer bevat het aantal manieren om de laatste regel van de invoer te maken uit fragmenten beschreven in de daaraan voorafgaande regels.

Voorbeeld Invoer

```
2
4
1 A
2 AC
2 CA
3 CAC
8 CACACACA
2
4 CAGT
4 TGAC
7 CAGTGAC
```

Voorbeeld Uitvoer

```
8
0
```

Opgave E - Patience

Met 30 lucifers kun je leuke spelletjes spelen (niet alleen pyromanen). Een ervan – een vorm van patience – is zo flauw dat je het beter aan een computer kunt overlaten. Dit spelletje gaat zo:

Verdeel de gegeven lucifers in een aantal hoopjes. Neem van ieder hoopje een lucifer en maak hiervan een nieuw hoopje, dat je toevoegt aan de resterende hoopjes. (Het geheel van operaties beschreven in de vorige zin zullen in het vervolg een zet noemen.) Herhaal dit totdat de verdeling over de hoopjes qua aantal niet meer verandert (dan heb je gewonnen), of totdat duidelijk is dat dit niet zal gebeuren (dan heb je verloren).

Voorbeeld.
5,1 → 4,2 → 3,1,2 → 2,1,3 (gewonnen)
5,2 → 4,1,2 → 3,1,3 → 2,2,3 → 1,1,2,3 → 1,2,4 (verloren)

Theoretici zullen opmerken dat je binnen 2^{31} zetten kunt weten of je gewonnen of verloren hebt, de goede zullen er aan toevoegen dat ze het gevoel hebben dat het met veel minder zetten kan. In de praktijk blijkt het spel vaak binnen enkele tientallen zetten tot een einde te komen.

Wat we hier willen weten is nog iets meer dan winst of verlies, namelijk ook nog het (minimale) aantal zetten dat je moet doen om een verdeling in hoopjes te bereiken die al eerder is voorgekomen.

Invoer

eerste regel: het aantal testcases.
de volgende regels: per regel: aantal hoopjes, gevolgd door een of meer spaties en aantallen lucifers in de hoopjes, gescheiden door spaties. (deze aantallen zijn uiteraard positief)

Uitvoer

voor iedere testcase een regel bestaande uit het minimale aantal zetten dat benodigd is om de uitslag te bepalen, gevolgd door een spatie en de tekst `GEWONNEN` danwel `VERLOREN`.

Voorbeeld Invoer

```
2
2 5 1
```

```
2 5 2
```

Voorbeeld Uitvoer

```
3 GEWONNEN
6 VERLOREN
```

Opgave F - Torens

Iedereen kent wel de torens van Hanoi. Een variant op dit spelletje is quadtoren. Het bestaat uit vier torentjes en negen blokjes. Er zijn drie kleuren blokjes (rood, blauw en groen) en elke kleur heeft een nummer 1 blokje, een nummer 2 blokje en een nummer 3 blokje. Men begint met een random configuratie van de blokjes op de vier torentjes, die elk hoogstens drie blokjes kunnen bevatten.

Schrijf nu een programma dat, gegeven een beginconfiguratie van de blokjes op de torens, het minimaal aantal stappen uitrekent dat gedaan moet worden om alle blokjes van 1 kleur op een apart torentje te krijgen met de voorwaarde dat de 1 onderop ligt, de 2 in het midden en de 3 bovenop. Het maakt hierbij niet uit welk torentje leeg blijft. Een stap is het verplaatsen van het bovenste blokje van een toren naar een ander torentje, daarbij rekening houdend met de zwaartekracht.

Invoer

De invoer bestaat uit:

- een regel met het aantal problemen (n)
- n keer 3 regels met 4 paren van een letter (`r`,`g`,`b`) en een cijfer(1,2,3). Een empty blokje wordt aangegeven door een `e0`. Men kan ervan uitgaan dat de beginconfiguraties consistent zijn.

Uitvoer

De uitvoer moet een file worden met evenveel regels als er problemen zijn opgelost (n). Op elke regel wordt het minimum aantal stappen vermeld in welke het puzzeltje kan worden opgelost.

Voorbeeld Invoer

```
2
e0 b3 r3 g3
e0 b2 r2 g2
e0 b1 r1 g1
b3 e0 e0 e0
r3 b2 r2 g2
g3 b1 r1 g1
```

Voorbeeld Uitvoer

```
0
3
```

Opgave G - Biatlon

De sport biatlon bestaat uit een combinatie van langlaufen en schieten. Deelnemers moeten een bepaald parcours (langlaufend) afleggen en op bepaalde plaatsen op het parcours moet geschoten worden op een aantal doelen. Wordt een doel gemist dan wordt een straftijd in rekening gebracht. De totaaltijd van een deelnemer is de tijd die hij nodig heeft om het parcours af te leggen plus eventuele straftijd. Winnaar is de persoon met de snelste totaaltijd.

Om de halve minuut start een deelnemer. Deze legt 5 keer een ronde af. Tussen deze rondes moet telkens geschoten worden, dus 4 keer in totaal. Een deelnemer mag per schietbeurt 5 keer schieten op vijf doelen (dus een poging per doel). Voor elk gemist doel wordt een straftijd van 1 minuut in rekening gebracht.

Duidelijk is dat in deze sport de winnaar niet degene is die als eerste de finishstreep passeert. Daarom is een programma nodig dat voor een gegeven gang van zaken bij een biathlonwedstrijd een eindstand afdrukt.

Invoer

De invoerfile bevat een aantal runs. De eerste regel van de invoerfile bevat een getal dat het aantal runs aangeeft. Daarna volgen de runs, die steeds uit drie delen bestaan. De eerste regel van een run bevat een getal D (< 100) dat het aantal deelnemers in deze run aangeeft. Daarna volgt de startlijst die uit D regels bestaat, een regel per deelnemer. Zo'n regel ziet er als volgt uit: De regel begint met drie cijfers, het startnummer van de deelnemer. Dit startnummer is uniek. Daarna volgen de naam van de deelnemer (maximale lengte is 20 karakters) en een code van drie karakters die het land van de deelnemer weergeeft. Het startnummer, de naam en de landcode zijn steeds gescheiden door spaties. Ook de volgorde van de deelnemers is van belang: de eerste deelnemer op de startlijst start als eerste, de volgende een halve minuut later enz.

Na de startlijst volgen weer D regels, dit is de finishlijst. Elke regel bevat informatie van een deelnemer die aangekomen is bij de finish. Deze lijst is chronologisch geordend. Een regel van de finishlijst begint met het tijdstip van aankomst in het formaat `uu:mm.ss`, dus uren, minuten en seconden gescheiden door eerst een ':' en daarna een '.'. Dit tijdstip is de verstreken tijd vanaf het begin van de wedstrijd (het begin van de wedstrijd is het moment waarop de eerste deelnemer start), dus niet noodzakelijk de verstreken tijd van de deelnemer! Na dit tijdstip volgt het startnummer, een rij van drie cijfers. De regel wordt afgesloten met vier getallen. Dit zijn de resultaten (= het aantal missers)

van de vier schietbeurten. De tijd, het startnummer en de vier schietresultaten zijn steeds gescheiden door spaties.

Uitvoer

De uitvoer moet een einduitslag per run bevatten. De eerste regel geeft het nummer van de run in het formaat `Uitslag Wedstrijd` #, met # het nummer van de run. Daarna volgen D regels die gesorteerd zijn van de eerste tot de laatste plaats. Een deelnemer met een snellere totaaltijd wint van een deelnemer met een langzamere totaaltijd. Als twee deelnemers dezelfde totaaltijd hebben is het aantal missers doorslaggevend: hoe minder hoe beter. Als zowel de totaaltijd als het aantal missers van twee deelnemers gelijk is, dan wint de als eerst gestarte deelnemer. De D regels met de einduitslag moeten er als volgt uit zien: Beginnend in kolom 1 de positie waarop de deelnemer geeindigd is. Beginnend in kolom 4 het startnummer van de deelnemer. Beginnend in kolom 9 de naam van de deelnemer. Beginnend in kolom 30 de landcode van de deelnemer. Beginnend in kolom 35 de totaaltijd van de deelnemer, gevolgd door een spatie en het aantal missers tussen haakjes. Elke run wordt beeindigd met een regel die bestaat uit 50 min-tekens ('-')

Voorbeeld Invoer

```
2
2
123 Robin Vandriessen USA
747 Pjotr Smirnov RUS
1:13.45 747 0 1 2 3
1:14.22 123 2 2 0 1
5 017 Peter Fisher GER
126 Kari Eloranta FIN
555 Sven Neyrinck BEL
230 Hans Dieter Frenzen SWI
999 Olav Moen NOR
0:59.17 017 0 1 0 2
1:00.33 126 0 1 1 0
1:05.09 999 4 0 2 2
1:08.09 555 0 1 2 1
1:11.46 230 0 3 0 3
```

Voorbeeld Uitvoer

```
Uitslag Wedstrijd 1
1 747 Pjotr Smirnov RUS 1:19.15 (6)
```

```
2 123 Robin Vandriessen USA 1:19.22 (5)
-------------------------------------------------
Uitslag Wedstrijd 2
1 126 Kari Eloranta FIN 1:02.03 (2)
2 017 Peter Fisher GER 1:02.17 (3)
3 555 Sven Neyrinck BEL 1:11.09 (4)
4 999 Olav Moen NOR 1:11.09 (8)
5 230 Hans Dieter Frenzen SWI 1:16.16 (6)
-------------------------------------------------
```

Opgave H - Klok

Het leven van een student is hard. Niet alleen moet je om 12:00 naar een college, ook moet je daarvoor nog snel een praktikum afmaken. Bovendien moet je je ook nog scheren! Dus je staat op tussen 00:00 's nachts en 11:59 's ochtends (inclusief), afhankelijk van hoeveel werk je nog moet verrichten, hoeveel glazen bier je de vorige avond op hebt en hoe interessant het college is. Als je op het punt staat je te gaan scheren, kijk je even op je antieke Friesche staartklok (inderdaad, geheel analoog, met wijzers) om de tijd in de gaten te houden. Onmiddelijk daarop draai je je om, en kijk je in de spiegel naar je klok, die je nu gespiegeld ziet. Als je niet op de cijfers, maar alleen op de wijzers let, zie je een andere tijd. Je vraagt je af, over hoeveel minuten het werkelijk zo laat zal zijn.

Invoer

De eerste regel van de invoer geeft het aantal runs. Elke volgende regel beschrijft een run en bevat alleen een tijd in het formaat `hh:mm`. Dit is de werkelijke tijd op het moment van scheren.

Uitvoer

De uitvoer voor iedere run bestaat uit een positief getal, dat aangeeft over hoeveel minuten het zo laat is als in de spiegel waargenomen wordt.

Voorbeeld Invoer

```
2
08:11
11:59
```

Voorbeeld Uitvoer

```
458
2
```