Beschrijving

Deze opgave gaat over een pre-historisch e-mail protocol. Computers kunnen mailtjes met elkaar uitwisselen, maar er is een aantal beperkingen. Een computer kan slechts één verbinding opzetten met een andere computer. Deze verbinding is rechtstreeks (dus andere computers merken hier niets van) en de verbinding kan ofwel zenden, ofwel ontvangen. Een computer kan enkel e-mail ontvangen van computers die:

1. een zendverbinding met hem hebben èn
2. waarmee hij een ontvangverbinding heeft

Als een e-mail tussen twee computers is uitgewisseld, worden de verbingen opgeheven en kan deze computer een andere verbinding opzetten.

Probleem

Het kan voorkomen, dat computers op elkaar wachten. Een voorbeeld hiervan is als A een zendverbinding naar B opzet, maar B tegelijkertijd zend naar C. Dan moet A wachten totdat B klaar is met C.

Het kan echter ook voorkomen, dat men in een soort lus terecht komt, waarbij een aantal computers eeuwig op elkaar zal blijven wachten: A zend naar B, B zend naar C en C zend naar A. Er is dan een zogenaamde ‘deadlock’ ontstaan.

De bedoeling is om gegeven een aantal computers met onderling opgezette zend- danwel ontvangverbindingen, of er sprake is van een deadlock.

Invoer — Bestand: A.IN

Een regel met het aantal runs. Vervolgens per run een regel met het aantal computers n ($0 \bullet n \bullet 100$) en voor iedere n een regel met:

- ‘`Z c`’, indien de computer een zendverbinding heeft met computer c
- ‘`O c`’, indien de computer een ontvangverbinding heeft met computer c
- ‘`G` ∅’, indien de computer geen verbindingen heeft

De nummering van de computers loopt van 1 tot n.

Uitvoer — Bestand: A.UIT

Voor iedere run een regel met daarop een ‘ja’ als er sprake is van deadlock of een ‘nee’ indien dat niet het geval is.

Voorbeeld

Bij de volgende invoer (A.IN):

```
2
3
Z 2
G ∅
O 1
4
Z 2
Z 3
O 4
Z 2
```

Hoort de uitvoer: (A.UIT):

```
nee
ja
```

Beschrijving

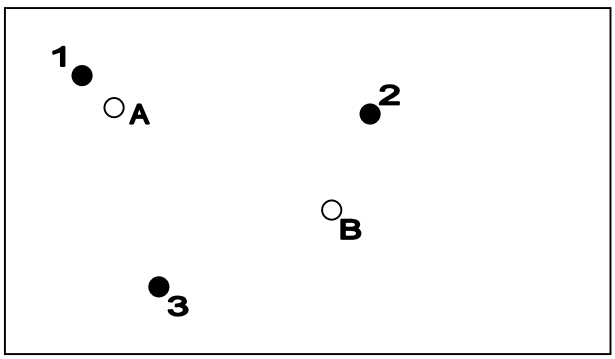
Met dank aan alle deelnemers aan de voorronde van het NKP heeft, is de heer Bill Poorts al een flink stuk opgeschoten met zijn Vensters 2001.

Om het geheel nu wat aantrekkelijker te maken dan het toch al is will hij aan zijn besturingssysteem een spelletje toevoegen, maar dan moet dat wel een stukje moderner zijn dan een-persoons patience.

Wat hij in gedachten heeft is een spel dat zich in de ruimte afspeelt en waarbij meerdere mensen tegenelkaar spelen vie het netwerk. Omdat hij de expertise zelf niet in huis heeft roept hij de hulp in van Tim Wijzeman, die met het idee 'SVGA-Planeten' komt.

Probleem

Zoals ingewijden zullen weten kan Tim Wijzeman echter totaal niet programmeren. Hij loopt dan ook al snel vast met de routine in zijn programma waarmee hij moet bepalen wat de dichtstbijzijnde planeet is als de speler ergens op de ruimtekaart met de muis klikt.



In het plaatje is duidelijk te zien dat muisklik A het dichtst bij planeet 1 is en muisklik B het dichtst bij planeet 2.

Invoer — Bestand: B.IN

Eerst een regel met het aantal runs, vervolgens per run een regel met het aantal planeten n ($1 \bullet n \bullet 5000$). Dan volgt voor elke planeet een regel met het x- en y-coördinaat, gescheiden door een spatie ($0 \bullet x,y \bullet 25000$). Er zijn geen twee planeten met dezelfde x en y coördinaat.

Daarna volgt een regel met het aantal muiskliks m ($0 \bullet m \bullet 30000$), gevolgd door per muisklik een regel met het x- en y-coördinaat ($0 \bullet x,y \bullet 25000$).

Uitvoer — Bestand: B.UIT

Voor elke muisklik een regel met daarop het nummer van de dichtstbijzijnde planeet, waarbij de eerste planeet in de invoer nummer 1 is enz.

Als er twee planeten op gelijke afstand van een muisklik staan hoeft alleen die planeet met het laagste nummer genoemd te worden.

Voorbeeld

Bij de volgende invoer (B.IN):

```
1
3
1 1
8 2
3 7
2
2 2
7 4
```

Hoort de uitvoer: (B.UIT):

```
1
2
```

Beschrijving

Kwak Zalver is een gerenommeerd chemicus, hij is dan ook hoogleraar aan de TUL (Technische Universiteit Lutjebroek). Kwak heeft echter één probleem, hij is ontzettend chaotisch.

Het komt dan ook regelmatig voor dat hij een bepaald mengsel heeft, waarvan hij niet weet wat het is. Aan de kleur en geur kan hij wel een beetje zien wat het kan zijn, maar zeker weten kan hij het alleen door te kijken welke vloeistoffen er nu werkelijk inzitten.

Dit doet hij door het mengsel te destilleren; hij verhit het mengsel en kijkt bij welke temperaturen er stoffen verdampen. Als hij bijvoorbeeld een mengsel heeft van water en alcohol, dan ziet hij dat de alcohol het eerst verdampt (bij ongeveer 79 graden Celsius). Bij 100 graden verdampt het water dan.

Natuurlijk heeft elke meetopstelling een onnauwkeurigheid. Gelukkig is die onnauwkeurigheid bekend. Als dit bijvoorbeeld 2 graden is, dan kan in plaats van 100 graden ook 98 graden of 102 graden worden gemeten als kookpunt van water. Door deze onnauwkeurigheid kan het ook voorkomen dat twee stoffen, die heel dicht bijelkaar liggen qua kookpunt tegelijkertijd lijken te verdampen of dat één stof op twee verschillende temperaturen lijkt te verdampen.

Probleem

Gegeven de resultaten van deze destilleerproef: een lijstje met temperaturen waarbij er iets verdampte en een lijst met mogelijke mengsels en stoffen met hun kookpunten, welk mengsel komt overeen met het onbekende mengsel. Of indien dit niet bepaald kan worden: hoeveel moet de meetopstelling nauwkeuriger worden?

Invoer — Bestand: C.IN

Eerst een regel met het aantal runs Daarna per run een regel met het aantal bekende stoffen *n* gevolgd door *n* regels met de kookpunten van de stoffen. Daarna een regel met het aantal mengsels *m*, gevolgd door *m* regels met de beschrijving van een mengsel. De beschrijving van een mengsel bestaat uit een getal *k* dat aangeeft uit hoeveel stoffen het mengsel bestaat, gevolgd door *k* stoffen aangeduid door hun volgnummer. (1..*n*)

Tenslotte worden per run de testresultaten gegeven. Allereerst een getal *p* dat aangeeft hoeveel testen er uitgevoerd zijn, gevolgd door *p* testresultaten. Elk test resultaat bestaat uit een regel met daarop allereerst een getal *d* dat de nauwkeurigheid van de meetopstelling aangeeft, gevolgd door het aantal gemeten kookpunten *q*. Tenslotte volgen de *q* kookpunten zelf.

Alle kookpunten en nauwkeurigheden zijn positief en kleiner dan 1000 graden. Voor alle aantallen geldt: $1 \leq k, n, m, p, q \leq 100$.

Uitvoer — Bestand: C.UIT

Als het juiste mengsel bepaald kan worden, dan een regel met: ‘Het mengsel is: ’ gevolgd door het nummer van het mengsel.

Kan het juiste mengsel niet bepaald worden, dan volgt eerst een regel met de tekst: ‘`Mogelijke mengsels:`’ gevolgd door een gesorteerde lijst van de mogelijke mengsels, gescheiden door een komma en een spatie.

Op de regel erna de tekst: ‘`Nauwkeurigheid:`’ gevolgd door de hoogste nauwkeurigheid, waarbij met dezelfde resultaten net wel het juiste mengsel gevonden kan worden.

Let op: hoogste nauwkeurigheid betekent dat de maximale afwijking dus zo groot mogelijk is en niet zo klein mogelijk, zoals men misschien verwacht bij de term hoogste nauwkeurigheid.

Er is altijd minstens één mengsel dat gegeven de testdata mogelijk is. Bovendien is het in de gebruikte testsets altijd mogelijk om een uniek mengsel aan te wijzen door de nauwkeurigheid te verbeteren (het getal dus te verlagen).

Voorbeeld

Bij de volgende invoer (C.IN): Alles na de ; is commentaar en hoort niet in de invoerfile.

```
1                   ;1 run
4                   ;4 stoffen
100                 ;1: kookpunt 100°
79                  ;2: kookpunt 79°
85                  ;3: kookpunt 85°
110                 ;4: kookpunt 110°
5                   ;5 mengels
2 1 2               ;1: stof 1+2
2 1 3               ;2: stof 1+3
3 1 2 3             ;3: stof 1+2+3
1 1                 ;4: stof 1
1 4                 ;5: stof 4
3                   ;3 onbekenden
1 2 79 100          ;d=1°   79° 100°
6 1 104             ;d=6°   104°
5 2 80 100          ;d=5°   80° 100°
```

Hoort de uitvoer: (C.UIT):

```
Het mengsel is: 1
Mogelijke mengsels: 4, 5
Nauwkeurigheid: 5
Mogelijke mengsels: 1, 2, 3
Nauwkeurigheid: 4
```

Beschrijving

Op de doorgaans o-zo-vredige campus (op wat exploderende vuilnisbakken en wat uit de hand gelopen flatfeesten na) van de Universiteit Twente op het oude landgoed Drienerlo is er nu iets ernstigs aan de hand.

De CVD (Campus Veiligheids Dienst) heeft geheime informatie ontvangen dat een aantal studenten die op de campus wonen van plan zijn om monitoren te gaan stelen in de verschillende onderwijsgebouwen.

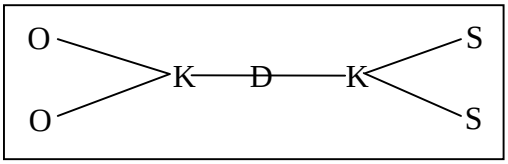
De enige informatie die zij hierbij hebben is een lijst met onderwijsgebouwen waar de fel begeerde 17” monitoren staan en een lijst met studenten en hun adressen die van plan zijn monitoren te gaan stelen. Omdat dit natuurlijk niet voldoende is voor een aanhouding, moeten de studenten op heterdaad betrapt worden dat ze met een monitor rondlopen.

Probleem

De Drienerloër Dienders die met de uitvoering van dit soort taken belast zijn hebben door de nieuwe financieringsstructuur van de universiteit een gebrek aan zesde geldstroom, waardoor ze dus te weinig mankracht hebben om alle onderwijsgebouwen te bewaken.

P. Olitie, het hoofd van Drienerloër Dienders heeft echter het vermoeden dat door het posten van zijn mannen op een aantal strategische wegen toch alle studenten gepakt kunnen worden.

In onderstaand plaatje ziet u dat hier één diender voldoende is. (O=gebouw, K=kruispunt van wegen, S=studentenflat, D=diender)



Invoer — Bestand: D.IN

Eerst een regel met het aantal runs, dan op de volgende regel drie getallen die respectievelijk het aantal onderwijsgebouwen, studentenflats en kruispunten aangeven. Het totaal van deze drie is maximaal 100 punten. Er is minimaal één onderwijsgebouw en één studentenflat.

Dan volgt een regel met het aantal wegen en vervolgens per weg een regel met daarop twee punten die aangeven tussen welke twee punten de weg loopt. Daarbij zijn de onderwijsgebouwen gemerkt als `O1`, `O2`, … de studentenflats als `S1`, `S2`, … en de kruispunten als `K1`, `K2`, …

Uitvoer — Bestand: D.UIT

De uitvoer bestaat uit een regel met het aantal wegen waar gepost moet worden en daarna deze wegen zelf. Zowel tussen de twee punten in deze regel zelf als bij alle wegen onderling wordt de volgende sortering toegepast:

- Eerst word gesorteerd op type: de onderwijsgebouwen eerst, daarna de kruispunten en daarna de studentenflats.
- In de tweede plaats wordt gesorteerd op het nummer, dus kruispunt 1 voor kruispunt 2 enz.

Als er meerdere mogelijkheden zijn om de dienders te plaatsen, dan worden ze zo dicht mogelijk bij de onderwijsgebouwen geplaats. Dit heeft dus niets te maken met de volgorde waarin de punten genummerd zijn.

Voorbeeld

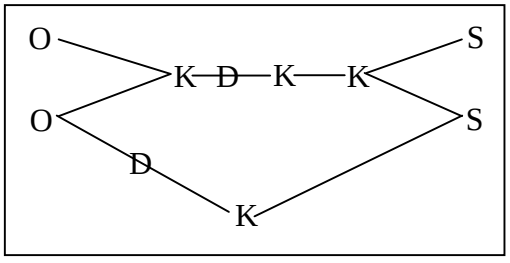
Bij de volgende invoer (D.IN):

```
2
2 2 2
5
01 K1
02 K1
K2 K1
K2 S1
K2 S2
2 4 2
8
01 K2
02 K2
02 K4
K2 K3
K3 K1
K1 S1
K1 S2
K4 S2
```

Hoort de uitvoer: (D.UIT):

```
1
K1 K2
2
02 K4
K2 K3
```

De eerste run uit de voorbeelduitvoer komt overeen met het plaatje op de vorige pagina. De tweede run zie er in een plaatje zo uit:



Beschrijving

Het is het jaar 2001, Vensters heeft ook zijn intrede gedaan op Starship Enterprise. Natuurlijk heeft Vensters een traditie hoog te houden en bevat dus een flink aantal bugs. Een van die bugs manifesteert zich bij het beroemde opstralen…

Om aan te geven welke objecten er opgestraald moeten worden, maakt de crew gebruik van Scotty's expressies. Normaal gesproken werkt dat als een zonnetje, maar als het aantal matches te groot is maakt Vensters er een potje van, wat voor de op te stralen personen niet fijn is, aangezien ze weleens half kunnen aankomen.

Probleem

Om dit te voorkomen, moet de crew dus van te voren weten of het fout kan gaan. Dus, moet gegeven een scotty's expressie, bepaald worden hoeveel matches deze maximaal kan opleveren. Je mag er vanuit gaan dat

- Elke ingevoerde expressie geldig is.
- 1 • maximum aantal matches • 2^{31}-1

Geldige Scotty's expressies voldoen aan de volgende grammatica. :

```
ALPHA      := 'a' - 'z'
DIGIT      := '0' - '9'
CHAR       := ALPHA
              DIGIT
WILD       := '#'
              '.'
              '$'
OR-PART    := EXPRESSIE '|' EXPRESSIE
              EXPRESSIE '|' OR-PART
OR         := '(' OR-PART ')'
PART       := WILD
              CHAR
              OR
EXPRESSIE  := PART
              PART EXPRESSIE
```

Hierin zijn drie typen wildcards gedefiniëerd:

- '#' matched 1 van de characters '0'-'9'
- '$' matched 1 van de characters 'a'-'z'
- '.' matched 1 van 'a'-'z', '0'-'9'

De constructie met haakjes en pipe-tekens duidt een 'of'-relatie aan. (a|b|c) betekent dus 'a' of 'b' of 'c'.

Het aantal matches dat gezocht wordt is het aantal verschillende strings dat de Scotty's expressie matched. 'a$#' heeft dus 1*26*10 = 260 mogelijke matches.

Omdat het programma niet al te slim is, worden expressies die in een of-relatie dubbel voorkomen ook dubbel geteld. De expressie (a|$) heeft dus 27 matches.

Invoer — Bestand: E.IN

Eerst een regel met daarop een integer die het aantal runs aangeeft. Vervolgens voor iedere run een regel met een geldige Scotty's expressie van maximaal 2000 karakters.

Uitvoer — Bestand: E.UIT

Voor iedere run een integer die aangeeft hoeveel matches de Scotty's expressie oplevert.

Voorbeeld

Bij de volgende invoer (E.IN):

```
3
..##(a|b)
jshdfsldkflkadjsf39939
(#|$|.)
```

Hoort de uitvoer: (E.UIT):

```
259200
1
72
```

Beschrijving

Vanwege de grote belangstelling voor de studie Informatica en de onvoldoende capaciteit van de opleidingen zal met ingang van het jaar 1999 niet iedereen meer tot de studie Informatica kunnen worden toegelaten.

Naar oud-Hollands gebruik zullen de beschikbare studieplaatsen onder de gegadigden worden verloot. De inlotingskans hangt af van het gemiddelde van de eindexamencijfers voor Wiskunde B en Natuurkunde.

Eerst worden de gegadigden worden verdeeld in 4 groepen:

Groep A : gem • 9
Groep B: 8 • gem < 9
Groep C: 7 • gem < 8
Groep D: gem < 7

Als de kans op inloting voor een kandidaat uit groep D gelijk is aan *X*, dan is die kans *2X* voor een kandidaat uit groep C, *3X* voor een kandidaat uit groep B, en *4X* voor een kandidaat uit groep A. Hierbij geldt dat wanneer een kans volgens deze regel groter dan 1 is, hij gelijk wordt gesteld aan 1.

De kans op inloting voor groep *g* noemen we *k(g)*. Het aantal kandidaten in groep *g* noemen we *n(g)*. Het aantal beschikbare plaatsen noemen we *m*. De bovengenoemde kans *X* kunt U (aannemende dat $n(A) + n(B) + n(C) + n(D) \bullet m$) berekenen uit:

$$n(A)k(A) + n(B)k(B) + n(C)k(C) + n(D)k(D) = m$$

In eerste instantie wordt het aantal studenten uit groep *g* dat wordt toegelaten bepaald op *n(g)k(g)*, naar beneden afgerond op een geheel getal. Vanwege deze afronding kunnen er dan nog plaatsen overschieten; deze worden zodanig toegewezen aan de groepen dat groep A als eerste profiteert, totdat alle kandidaten van deze groep worden toegelaten, daarna groep B, daarna groep C, tenslotte groep D.

De loting is als volgt georganiseerd. Alle gegadigden melden zich centraal aan bij de Informatie Beweer Groep. Aangezien deze aanmelding plaatsvindt ver voor het eindexamen, vindt er nog geen indeling in groepen plaats.

Iedere kandidaat krijgt door het lot een lotingsnummer toegewezen. De lotingsnummers zijn de getallen *1..N*, waarbij N het aantal kandidaten is.

Na afloop van de eindexamens melden de kandidaten hun eindexamencijfers voor wiskunde B en natuurkunde. Op grond daarvan worden de groepen vastgesteld, alsmede het aantal kandidaten van iedere groep dat wordt toegelaten.

In iedere groep heeft een lager inlotingsnummer voorrang boven een hoger inlotingsnummer. Voor iedere groep geldt: het hoogste lotingsnummer van de groep is het hoogste lotingsnummer van alle toegelaten kandidaten van de groep; als geen enkele kandidaat wordt toegelaten dan is het hoogste lotingsnummer gelijk aan 0.

Probleem

In deze programmeeropdracht is de loting al geschiedt en zijn de cijfers bekend. Wat er nog moet gebeuren is dat voor iedere groep het hoogste lotingsnummer wordt bepaald.

Invoer **Bestand: F.IN**

Eerst een regel met het aantal runs. Daarna per run op de eerste twee regels respectievelijk het aantal beschikbare plaatsen *M* en het aantal aangemelde studenten *N*. Er geldt: 0 • *M* • *N* • 10000.

Daarna komen er in de file *N* regels. Iedere regel begint met de naam van de kandidaat, gevolgd door het inlotingsnummer, gevolgd door het cijfer voor wiskunde B, gevolgd door het cijfer voor natuurkunde. Naam en cijfers worden gescheiden door spaties, en de naam bevat geen spaties. De eindexamencijfers zijn gehele getallen, minimaal 1 en maximaal 10.

Uitvoer **Bestand: F.UIT**

Per run dient uw programma 4 regels te produceren, met op iedere regel een geheel getal. De eerste regel bevat het maximale inlotingsnummer voor groep A, de tweede dat voor groep B, de derde dat voor groep C en de vierde dat voor groep D.

Voorbeeld

Bij de volgende invoer (F.IN):

```
2
1 2
Jan 1 9 8
Tom 2 6 7
6 10
JandeGroot 2 5 6
PetraJanssen 3 6 7
PieterdeBoer 4 7 8
KlaasVaak 8 9 9
PeterPan 7 5 7
JeroenvanGelderen 9 6 8
StijndeBekker 10 8 8
LutgerKunst 5 7 7
EelcoBrolman 6 9 9
EdwinWoudt 1 10 9
```

Hoort de uitvoer: (F.UIT):

```
1
0
0
0
8
10
5
0
```

Beschrijving

Docent G. Emakzucht onderwijst verscheidene groepen studenten op de Technische Universiteit Lutjebroek. Iedere TUL-student heeft een uniek Student Identiteits Nummer (SIN). Een SIN is een natuurlijk getal kleiner dan 1.000.000.

De docent heeft het wat moeilijk met het onthouden van deze nummers en daarom zoekt hij een andere manier om de studenten te identificeren.

Hij heeft hierbij gekozen om bij iedere groep een zo klein mogelijke 'magische' modulus *m* (0<*m*<=MaxSIN), zodanig dat binnen de groep de SINs gereduceerd modulo m uniek zijn.

Als de SIN's in de groep de volgende zijn: 27721, 1, 73169, 19601, 25568, 30317, dan kan de docent alles modulo 61 doen, zodat de SIN's toch verschillend worden, maar gemakkelijker te onthouden, immers:

27721 modulo 61 = 27
1 modulo 61 = 1
73169 modulo 61 = 30
19601 modulo 61 = 25
25568 modulo 61 = 9
30317 modulo 61 = 0

Probleem

De docent heeft nu wel een manier bedacht om de SIN's simpeler te maken, maar hij moet nog wel steeds met de hand wat het 'magische' getal *m* is. Dus de opdracht is: schrijf een programma dat de docent dit werk uit handen neemt.

Invoer **Bestand: G.IN**

Eerst een regel met het aantal runs, dan per run een regel met het aantal SIN's. Daarna de SIN's zelf, één per regel. Natuurlijk zijn alle SIN's in een run verschillend.

Uitvoer **Bestand: G.UIT**

Voor elke testset een regel met daarop de kleinst mogelijke *m*, waarbij alle SIN's modulo *m* nog steeds verschillend zijn.

Voorbeeld

Bij de volgende invoer (G.IN):

```
1
6
27721
1
73169
19601
25568
30317
```

Hoort de uitvoer: (G.UIT):

```
61
```

Beschrijving

Aangezien de systeembeheerders Ronald Hoi en Fred Eendevlees weinig te doen hebben, hebben ze een spelletje bedacht om de tijd te verdrijven: ‘The Last Coin’.

Bij dit spel leggen ze beiden een aantal munten op tafel en maken ze er stapeltjes van één of twee munten van. Nu hebben ze steeds de mogelijkheid om een geheel stapeltje weg te nemen, of in het geval van een stapeltje van twee, daar één van af te pakken. Diegene het laatste muntje pakt (The Last Coin) heeft gewonnen.

Probleem

Aangezien Fred Eendenvlees een programma heeft geschreven om dit spel zo goed mogelijk te spelen kan Ronald Hoi natuurlijk niet achterblijven.

Maar ja, Ronald Hoi is wel eens een beetje lui, dus heeft hij aan de NKP commissie gevraagd of die niet zo’n programma konden schrijven. De NKP commissie had hier alleen niet zo’n zin in, dus vandaar dat het nu een NKP opgave is geworden.

Om precies te zijn is de opdracht: Schrijf een programma dat in een gegeven stelling bepaalt of de positie gewonnen of verloren is en in het geval van een gewonnen stelling de winnende zet verteld.

Invoer **Bestand: H.IN**

De invoer bestaat eerst uit een regel met het aantal runs, dan per run een regel met het aantal stapeltjes (maximaal 100) en dan een regel met precies dit aantal aan ‘1’-tjes en ‘2’-tjes.

Uitvoer **Bestand: H.UIT**

In het uitvoerbestand moet per run één regel weggeschreven worden.

Als de stelling verloren is, dan moet er een regel met ‘`Verloren`’ in het uitvoerbestand komen.

Als de stelling gewonnen is, dan moet er een regel met ‘De winnende zet is: ’ gevolgd door de positie van het stapeltje waarvanaf men iets pakt en het aantal munten dat men eraf pakt.

Als er meerdere mogelijkheden zijn voor de winnende zet, dan moet die mogelijkheid gegeven worden die voor de positie van het stapeltje het laagste nummer oplevert.

De nummering van de posities van de stapeltjes begint bij 1 voor het in de invoerfile meest ‘linkse’ stapeltje.

Voorbeeld

Bij de volgende invoer (H.IN):

```
3
3
111
10
2221122112
3
112
```

Hoort de uitvoer: (H.UIT):

```
De winnende zet is: 1 1
Verloren
De winnende zet is: 3 2
```