leuven SWEPSILO

Opgave A - Muis & kaas

In een twee-dimensionale wereld vol driehoeken leeft een muis. Om zich in leven te houden eet het muisje kaas. Nu is er één probleem : een muisje kan niet in een driehoek komen, of juist uit een driehoek komen (als hij in een driehoek begint). Dit houdt in dat hij alleen een stukje kaas te pakken kan krijgen als

a) Het muisje start niet in een driehoek èn
   Het stukje kaas ligt niet in een driehoek,

b) Het muisje start in een driehoek èn
   Het stukje kaas ligt in dezelfde driehoek.

Schrijf een programma dat bepaalt hoeveel stukjes kaas een muisje kan pakken.

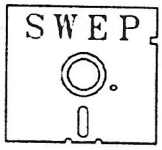
Specificatie : De startpositie van de muis en de posities van de stukjes kaas worden gegeven door coördinaten. De driehoeken worden gegeven door de coördinaten van de hoekpunten. De waarden van de alle coördinaten liggen tussen -25000 en 25000 (gehele getallen). De driehoeken overlappen elkaar niet. De coördinaten van de stukjes kaas en de coördinaten van de muis liggen niet op de rand van een driehoek.

De invoer (A.IN) bestaat uit een aantal runs. Elke run bevat een onbekend aantal regels en ziet er als volgt uit :

- Startcoördinaat van de muis
- Aantal stukjes kaas ($n \leq 500$)
- Coördinaat stukje kaas 1
- Coördinaat stukje kaas 2
- Coördinaat stukje kaas 3
- .....
- Coördinaat stukje kaas n
- Aantal driehoeken ($m \leq 500$)
- Coördinaten driehoek 1
- Coördinaten driehoek 2
- .....
- Coördinaten driehoek m

Het aantal regels is dus in dit geval 3+n+m. De runs worden gescheiden door een *. Na de laatste run staat echter een #.

De uitvoer (A.UIT) bevat alleen voor elke run het aantal stukjes kaas dat een muisje kan bereiken (elk aantal op een eigen regel).



IBM

Voorbeeld

Invoer :

```
100 100
4
105 101
91 101
91 109
100 111
3
95 100 91 103 97 103
102 104 99 108 105 108
92 107 88 110 95 110
*
0 0
4
1 1
2 2
3 3
4 4
1
0 1 1 0 -1 -1
#
```

Uitvoer :

```
3
0
```

SWEP

IBM

/evien swepgilo

Opgave B - Loesje

In café "De Wilde Plakker" is een rechthoekige muur bestemd voor het opplakken van affiches. Nieuwe affiches worden naast, maar soms ook geheel of gedeeltelijk over oude heen geplakt. Na enige tijd zijn sommige oudere affiches deels of totaal onzichtbaar. De vraag die gesteld wordt is de volgende: nadat een reeks van n affiches is geplakt, hoeveel zijn er geheel zichtbaar, hoeveel zijn er deels zichtbaar, en hoeveel zijn er geheel onzichtbaar?

In de invoerfile, (B.IN), staan een aantal reeksen affiches. Elke reeks affiches begint met een regel met een getal (het aantal affiches n), gevolgd door de n affiches zelf, in de volgorde waarin ze geplakt zijn. Elke affiche staat op een nieuwe regel. Een affiche wordt voorgesteld door vier reële getallen, gescheiden door spaties: de x- en y-coördinaat van het middelpunt van de affiche op de muur, en de breedte en de hoogte van de (rechthoekige) affiche. Hierbij "loopt" de x-coördinaat in de breedte. De laatste regel van de invoerfile bevat alleen het getal 0.

In de uitvoerfile (B.UIT) moeten de resultaten van de reeksen komen te staan, elke reeks op een eigen regel. Het resultaat van een reeks bestaat uit 4 getallen, gescheiden door komma's: het totaal aantal affiches n, het aantal geheel zichtbare, het aantal deels zichtbare, en het aantal onzichtbare affiches.

Opmerkingen:

1. Elke reeks begint op een schone, lege muur. De affiches worden alle evenwijdig aan de zijden van de muur geplakt. Over de maten van de muur hoeft u zich geen zorgen te maken.
2. De affiches worden nooit "stotend" geplakt: er is altijd een (wellicht kleine) overlap, of ze hebben niets met elkaar gemeen.
3. Het aantal posters n voldoet aan 1 <= n <= 50. De x- en y-coördinaten liggen tussen 0 en 1000.

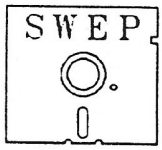
Voorbeeld (waarbij de derde poster van de eerste reeks deels over de eerste ligt):

Invoer :

```
3
10 10 4 4
40.1 40 2.5 2.645
8.2 8 4 4
2
20 20 5 5
20 20 6 6
0
```

Uitvoer :

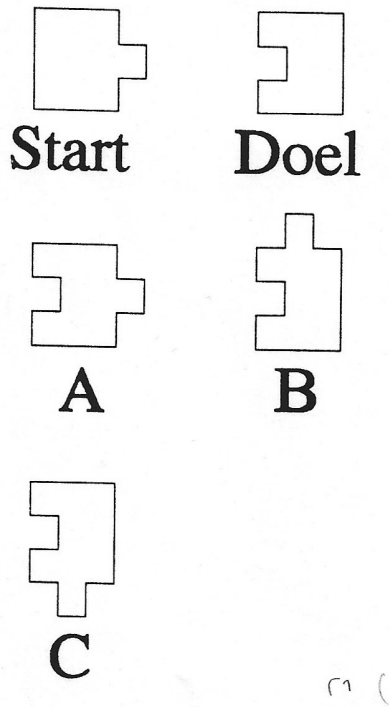
```
3,2,1,0
2,1,0,1
```

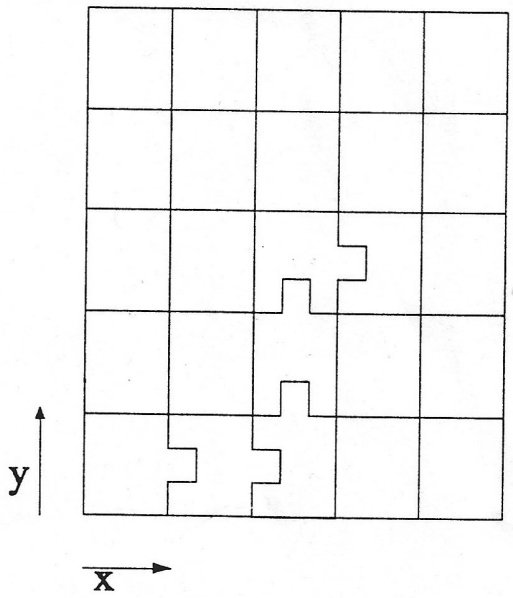


IBM

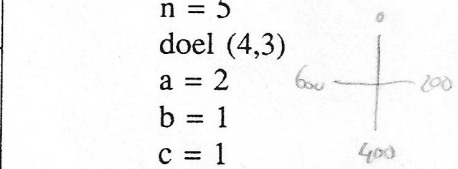
Opgave C - Slang

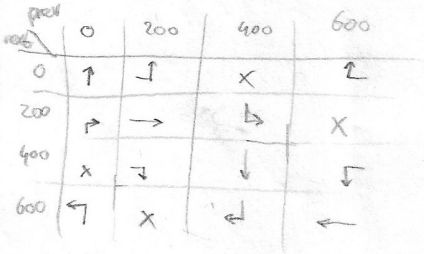
Een twee-dimensionale variant van de slang van Rubik bestaat uit vierkante stukjes van 1 bij 1 centimeter welke van uitsparingen en uitstulpingen voorzien zijn als bij legpuzzels. Er zijn 5 soorten stukjes, geheten Start, Doel, A, B en C (zie tekening). Van Start en Doel is elk 1 stukje voorradig, van A a stuks, van B b en van C c stuks. De stukjes liggen met de bovenkant boven en mogen niet omgekeerd worden (zodat stukjes B en C echt verschillen). Uiteraard passen uitstulpingen en uitsparingen precies in elkaar. De stukjes A zijn niet van elkaar te onderscheiden; dit zelfde geldt voor de stukjes B en C.

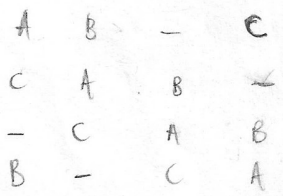




n = 5
doel (4,3)
a = 2
b = 1
c = 1







Verder hebben we de beschikking over een n bij n schaakbord met vakjes van 1 bij 1 centimeter. Gegeven worden x- en y-coördinaat van een vakje, en a, b, en c. Gevraagd wordt het aantal verschillende aaneengesloten slangen, die beginnend in (1,1) (links onderaan op het bord; hier komt stukje Start) uitkomen in het gegeven vakje (waar stukje Doel ligt). Uiteraard mag er niets over de rand uitsteken en moeten alle uitsparingen gevuld zijn. Voorbeeld: zie tekening.

De invoerfile (C.IN) bevat voor elk "probleem" een regel met de grootte van het bord, gevolgd door een regel met achtereenvolgens x- en y-coördinaat van het doel, a, b en c (gescheiden door spaties). De laatste regel van de invoerfile bevat alleen het getal 0. De uitvoerfile C.UIT moet voor elk probleem het gevraagde aantal bevatten (1 getal per regel).

Opmerkingen:

1. Let op de in het plaatje aangegeven richtingen van x- en y-as (dit in verband met de stukjes B en C).
2. Restricties: 2 <= n <= 10, a + b + c <= 20, a >= 0, b >= 0, c >= 0.
3. Het doel verschilt altijd van het startpunt (1,1).
4. Een slang hoeft niet alle stukjes A, B en C op te maken.

Voorbeeld:

Invoer :
```
5
4 3 2 1 1
8
2 2 3 3 3
0
```

Uitvoer :
```
3
10
```

Voorronde NK gi 2/2 C

Opgave D - Anagrammen

Deze opgave betreft het zoeken van anagrammen in een bestand met woorden. Een anagram is een woord dat ontstaat uit een ander woord door de letters van dat andere woord door elkaar te husselen. Zo is bijvoorbeeld 'poets' een anagram van 'stoep' en 'spreektaal' van 'speelkaart'.
De bedoeling is nu om anagrammen te vinden in een lijst met woorden.

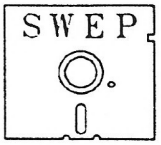
- Invoer (D.IN)

De invoer bevat een aantal lijsten met woorden. Elke lijst begint met een getal dat het aantal woorden in die lijst aangeeft. Vervolgens komen de woorden, die alle alleen uit lower-case letters ('a', ..., 'z') bestaan. Elk woord staat op een aparte regel en is maximaal 10 letters lang (en minimaal 1 letter). Het aantal woorden per lijst uit de invoer is altijd ≥ 0 en ≤ 250.

- Gewenste uitvoer (D.UIT)

Voor elke lijst met woorden uit de invoer dient de bijbehorende uitvoer te bestaan uit een gegroepeerde rij woorden, waarbij anagrammen bij elkaar staan. Tussen twee groepen anagrammen moet steeds een asterisk ('*') staan. Woorden die geen anagrammen hebben mogen niet in de uitvoer komen te staan. (In onderstaand voorbeeld heeft het woord 'feest' geen anagrammen, en komt dus niet in de uitvoer voor.) De onderlinge volgorde in de uitvoer van woorden binnen een groep van anagrammen is niet van belang (in onderstaand voorbeeld mogen 'poes', 'pose' en 'soep' ook in een andere volgorde staan). Ook de volgorde waarin de groepen anagrammen in de uitvoer staan is niet belangrijk (in het voorbeeld kan het rijtje anagrammen 'mok', 'kom' ook na het rijtje 'poes', 'pose', soep' komen). Een woord mag meer dan één keer in een lijst voorkomen. Een woord is een anagram van zichzelf, dus het moet dan ook even zo veel keer in de bijbehorende groep anagrammen terecht komen.

In de uitvoer dient elke uitvoerrij, behorende bij een lijst van woorden uit de invoer, te worden gevolgd door een hekje ('#').



Voorbeeld.

Invoer :

```
6
poes
mok
feest
soep
pose
kom
5
een
twee
nee
een
drie
2
groot
klein
-1
```

Uitvoer :

```
mok
kom
*
poes
pose
soep
#
een
nee
een
#
#
```

Voorronde NK gi 2/2 D

levier swergilo

Opgave E - Een straat

We hebben een straat met vier huizen, genummerd 2, 4, 6 en 8. Verder zijn gegeven vier personen (geïdentificeerd door hun nationaliteit) en vier honden (geïdentificeerd door hun soortnaam). Over deze personen en honden hebben we slechts gedeeltelijke informatie, zoals:

Op nummer 6 woont een retriever
De Chinees heeft een spaniel
Op nummer 4 woont een Duitser
De Italiaan heeft geen poedel

Ook hebben we enige informatie over buurrelaties:

De Duitser woont naast de Italiaan
De poedel woont naast de dog

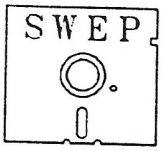
In de straat wonen geen twee mensen met dezelfde nationaliteit of met dezelfde soort hond. Bij ieder huis hoort een nationaliteit (een bewoner) en een hond (elke persoon heeft een hond). Op grond van de gegeven informatie willen we concluderen welke persoon en welke hond in elk van de vier huizen wonen. De informatie hoeft niet volledig te zijn (dan zijn er meerdere oplossingen mogelijk) en kan ook strijdig zijn (dan zijn er geen oplossingen mogelijk).

- Invoer (E.IN)

De invoerfile is een tekstfile, bestaande uit een blok waarin vier soorten honden en vier nationaliteiten staan. Dit blok bestaat uit twee regels: de eerste bevat vier door een spatie gescheiden woorden die de hondesoorten geven en de tweede idem dito voor de nationaliteiten.

Vervolgens komen enkele blokken met informatie zoals boven genoemd. Voor ieder van deze blokken moet worden nagegaan welke persoon met welke hond in elk van de huizen woont. Een blok informatie begint met een getal, dat het aantal regels informatie aangeeft. Vervolgens de informatie zelf. Deze informatie is als volgt gecodeerd:

CODE	BETEKENIS
+ 6 retriever	Op nummer 6 woont een retriever
+ CHINEES spaniel	De Chinees heeft een spaniel
+ 4 DUITSER	Op nummer 4 woont een Duitser
- ITALIAAN poedel	De Italiaan heeft geen poedel
- 2 DUITSER	Op nummer 2 woont geen Duitser
- 6 dog	Op nummer 6 woont geen dog
x DUITSER ITALIAAN	De Duitser woont naast de Italiaan
x poedel dog	De poedel woont naast de dog
x DUITSER retriever	De Duitser woont naast de retriever



Elke regel informatie is dus van de vorm <op> <cons> <cons>, waarin <op> een operator is uit de verzameling {+, -, x}, en <cons> een constante: een integer (huisnummer), een karakterstring in hoofdletters (nationaliteit) of een karakterstring in kleine letters (hondesoort). U mag er van uitgaan dat een blok hooguit 20 regels informatie bevat. In buurrelaties waarin een nationaliteit en een hondesoort voorkomen wordt altijd eerst de nationaliteit gegeven. In de '+-relaties' geldt altijd: nummer voor nationaliteit voor hondesoort.

De invoer wordt afgesloten met -1.

- Gewenste uitvoer (E.UIT)

De uitvoer bestaat voor elk blok met informatie uit een regel met het aantal mogelijke oplossingen. In het geval dat er precies een oplossing is, moet deze eveneens in de uitvoer komen te staan (zie onderstaand voorbeeld). De uitvoer behorend bij de verschillende blokken dient te worden gescheiden door een blanco regel.

Voorbeeld:

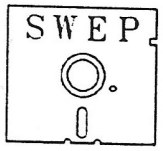
Invoer :

```
dog retriever poedel spaniel
CHINEES DUITSER ITALIAAN HONGAAR
1
+ 4 DUITSER
6
+ 8 HONGAAR
+ 2 dog
x DUITSER dog
x ITALIAAN HONGAAR
- 6 spaniel
+ HONGAAR poedel
-1
```

Uitvoer :

```
144

1
2 : CHINEES dog
4 : DUITSER spaniel
6 : ITALIAAN retriever
8 : HONGAAR poedel
```



Levien Sweigelo

Opgave F - Patience

Om aan de sleur van de werkgroepen Inleiding Pascal te ontkomen speelt Blaise een nieuw één-persoons kaartspel. Een aantal kaarten wordt met de beeldzijde naar boven op een rijtje uitgelegd. Het is nu de bedoeling om, volgens van te voren opgestelde spelregels, alle kaarten weg te halen. Deze spelregels schrijven voor hoe een rijtje naast elkaar liggende kaarten vervangen mag worden door een ander, korter, rijtje kaarten. Bijvoorbeeld: Aas, Aas mag vervangen worden door Heer; Aas, Heer mag vervangen worden door een Heer; Aas, Aas, Aas mag vervangen worden door Twee, Vrouw; Twee, Vrouw, Heer mag geheel weggenomen worden. Deze regels kunnen per spelletje verschillen.
Beginnend met Aas, Aas, Aas, Aas, Aas, Aas kunnen met deze spelregels de kaarten weggenomen worden via Twee, Vrouw, Aas, Aas, Aas; dan Twee, Vrouw, Aas, Heer; dan Twee, Vrouw, Heer; waarna we deze drie kaarten weg kunnen halen.
Als de spelregels eenmaal vaststaan is het natuurlijk niet altijd mogelijk om alle kaarten weg te halen. (In ons voorbeeld kunnen we bijvoorbeeld alleen met de kaarten Twee, Vrouw, Heer en Aas aan de slag.) De bedoeling is dan ook om een programma te schrijven dat, bij gegeven spelregels en beginstand, bepaalt of alle kaarten weggenomen kunnen worden.

Het invoerfile F.IN beschrijft een aantal spelen, dat wil zeggen de regels en beginposities. De kaarten worden weergegeven met de symbolen 2,...,9,T (voor de tien), en B,V,H en A. De beschrijving van elk spel begint met een regel met de beginpositie. Daarna volgt een regel met het aantal spelregels, gevolgd door de spelregels zelf. Elke spelregel staat op een aparte regel, en is van de vorm "oorspronkelijk"->"nieuw". De spelen zèlf worden gescheiden door een *. Na het laatste spel volgt echter een #.

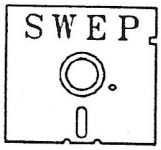
Het spel uit ons voorbeeld wordt dus beschreven door:

```
AAAAAA
4
AA->H
AH->H
AAA->2V
2VH->
```

De uitvoer F.UIT bevat voor elk van de spelen een regel met daarop JA als het spel tot succes leidt en NEE als niet alle kaarten van de beginstand volgens de regels weggenomen kunnen worden.

Opmerkingen:

- We mogen uitgaan van maximaal 15 verschillende regels per spel. Elk spel start met maximaal 10 uitgelegde kaarten.
- Elke spelregel zorgt dat het aantal kaarten met ten minste één vermindert.
- Het aantal kaarten van elke soort wordt niet, zoals in één set kaarten, beperkt tot vier. Er mag in de beginstand een willekeurig aantal van elke soort voorkomen.



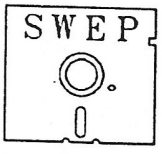
Voorbeeld:

Invoer:

```
AAAAAA
4
AA->H
AH->H
AAA->2V
2VH->
*
HAAAAAAAA
*2 2
A->
AA->A
#
```

Uitvoer:

```
JA
NEE
```



IBM