NK Programmeren 1996

POLSSTOKHOOGSPRINGEN

Opgave A

Polsstokhoogspringen is een van de meest veeleisende onderdelen van de atletiek. Niet alleen conditie en techniek zijn belangrijk, maar ook gevoel voor tactiek. De belangrijkste spelregel is: wie het hoogst springt wint, maar vaak is het niet zo eenvoudig en wint degene met de beste tactiek. Wanneer meerdere atleten een gelijke hoogte overbruggen, dan is de zojuist genoemde regel niet voldoende. In zo'n geval treedt ''regel 41'' in werking.

Regel 41:

Bij gelijke prestaties beslist het laagste aantal pogingen bij achtereenvolgens:

(a) De laatste geldig gesprongen hoogte;

(b) Alle foutsprongen bij elkaar (tot en met de laatste geldige poging);

(c) Alle sprongen bij elkaar (tot en met de laatste geldige poging).

Wanneer ook deze regel geen beslissing kan brengen dan eindigen atleten gelijk.

Overige wedstrijdregels zijn:

- Er wordt gesprongen op bepaalde, vooraf vastgestelde hoogten;
- Wanneer een deelnemer springt, dan kan de poging of geldig zijn (weergegeven met een 'o') of ongeldig (weergegeven met een 'x'). Wanneer een deelnemer 3 opeenvolgende pogingen ongeldig springt, dan mag deze deelnemer niet meer springen (einde oefening);
- Na elke geldige poging moet de volgende hoogte waarop de deelnemer springt strict groter zijn dan de vorige poging;
- Na elke ongeldige poging moet de volgende hoogte waarop de deelnemer springt groter of gelijk zijn aan de vorige poging;
- Deelnemers mogen op elk moment de wedstrijd beëindigen, dus niet noodzakelijk pas na 3 opeenvolgende ongeldige pogingen;
- Wanneer een deelnemer niet springt op een bepaalde hoogte dan wordt dit weergegeven met een '-'.

Voorbeeld:

Bij de olympische spelen van 1976 (zie voorbeeld invoer) eindigden 3 deelnemers met dezelfde maximaal behaalde hoogte: 5.50. Om uit te maken hoe de medailles verdeeld moesten worden, moest regel 41 dus weer toegepast worden. Om te beginnen regel 41a: er werd gekeken naar het aantal pogingen op de laatst behaalde hoogte (5.50 dus). Alle drie de deelnemers sprongen slechts eenmaal op 5.50, dus bracht regel a geen uitkomst. Vervolgens regel 41b: het totaal aantal foutsprongen in de

wedstrijd, tot en met 5.50. Zowel Slusarski als Kalliomaki hadden 0 ongeldige pogingen. Roberts eenmaal. Roberts werd dus 3de, en regel 41c moest er tenslotte aan te pas komen om de beslissing te brengen: Slusarski had tot en met 5.50m 3 keer ge-sprongen, Kalliomaki 5 maal. Slusarski werd dus eerste, Kalliomaki 2de.

Het is nu jouw taak om de wedstrijdregels van het polsstokspringen te interpreteren en de uitslag van een wedstrijd te geven.

Invoer

De eerste regel van de testinvoer bevat een integer: het aantal runs *R*.

Vervolgens komen na elkaar *R* runs, die bestaan uit:

- Een eerste regel die uitsluitend bestaat uit een rij van de karakters '`-`' (minteken). Deze regel is maximaal 200 karakters lang;
- Een tweede regel die begint met een getal P (het aantal deelnemers), P # 50, een spatie, een rij karakters die begint en eindigt met een '`'`' (tussen de aanhalingstekens staat de naam van de wedstrijd), gevolgd door tenminste één spatie en vervolgens een aantal (*N*, 1 # *N* # 20) getallen tussen 0.00 en 9.99 (twee decimalen nauwkeurig) die de hoogten weergeven waarop de deelnemers mogen springen. De lengte van deze regel is gelijk aan de lengte van de eerste regel en bevat geen spaties aan het einde;
- De derde regel is gelijk aan de eerste regel van de run;
- Hierna volgen *P* regels, die bestaan uit:
    - de naam van de deelnemers (deze begint aan het begin van de regel en wordt be ëindigd met een '`,`' (komma). De maximale lengte van de naam is 20 karakters;
    - hierna volgt een spatie;
    - na de spatie volgen drie hoofdletters, die het land van de deelnemer weergeven;
    - vervolgens staat onder elke hoogte, beginnend in dezelfde kolom als waarin de hoogte in de header wordt gedeclareerd (= tweede regel van de run) de (maximaal 3) resultaten van de deelnemer op deze hoogte. Dit kan zijn:

- '`-`' : de deelnemer heeft niet gesprongen op deze hoogte;
- '`o`' : de deelnemer heeft succesvol gesprongen (geldige poging);
- '`x`' : de deelnemer heeft niet succesvol gesprongen (ongeldige poging);
- ' ' : de deelnemer heeft de wedstrijd beëindigd.
    - de resultaten tussen twee opeenvolgende hoogten worden gescheiden door spaties.

Let op dat een deelnemer maximaal drie pogingen heeft per hoogte.

Uitvoer

Per run moet de uitvoer voldoen aan het volgende formaat. Runs worden gevolgd door een lege regel. De eerste regel van de uitvoer van een run bestaat uit het woord '`Uitslag`', gevolgd door een spatie en de naam van de wedstrijd tussen aanhalingstekens ('`'`').

De tweede regel van de run bestaat uit mintekens ('`-`'), en is net zo lang als de derde regel.

De derde regel van een run bevat de strings:

- ‘`##`’ beginnend in kolom 1;
- ‘`NAAM`’ beginnend in kolom 4;
- ‘`LAND`’ beginnend in kolom 24;
- ‘`HOOGTE`’ beginnend in kolom 29;
- ‘`41A`’ beginnend in kolom 37;
- ‘`41B`’ beginnend in kolom 42;
- ‘`41C`’ beginnend in kolom 47.

Vervolgens komen *P* regels met daarin een gesorteerde uitslag, beginnend met de winnaar, die bestaat uit:

- Positie, dit is een geheel getal, rechts uitgelijnd in een veld van breedte 2 dat begint in kolom 1;
- Naam, beginnend in kolom 4;
- Land, beginnend in kolom 24;
- De maximaal behaalde hoogte, beginnend in kolom 29. Als geen hoogte gehaald is, moet ‘`0.00`’ afgedrukt worden;
- Het aantal sprongen volgens regel 41A, beginnend in kolom 37;
- Het aantal sprongen volgens regel 41B, beginnend in kolom 42;
- Het aantal sprongen volgens regel 41C, beginnend in kolom 47.

De laatste regel van een run is gelijk aan de tweede regel van de run en bestaat dus uit een rij mintekens. De volgorde waarin atleten die gelijk eindigen afgedrukt worden is gelijk aan de volgorde van betreffende atleten in de invoer.

Voorbeeldinvoer

```
2

-------------------------------------------------------------------------------

6 ``Olympische spelen 1976``   5.10 5.20 5.25 5.30 5.35 5.40 5.45 5.50 5.55 5.60

-------------------------------------------------------------------------------

Patrick Abada, FRA           -    -    -    XO   -    -    O    -    XXX

Earl Bell, USA               -    O    -    -    O    -    XXO  -    XXX

Wojciech Buciarski, POL      -    O    -    -    XO   -    XO   X    XX

Antti Kalliomaki, FIN        O    -    -    O    -    O    O    O    XXX

David Roberts, USA           -    -    -    -    XO   -    -    O    -    XXX

Tadeusz Slusarski, POL       -    O    -    -    -    O    -    O    XXX

-----------------------------------

2 ``Gelijkspel `96``    2.40 2.50 2.60

-----------------------------------
```

```
Bart den Boer, HOL    X    X    X
Bert den Boer, BEL    -    XXX
```

Voorbeelduitvoer

```
Uitslag ``Olympische spelen 1976``
-------------------------------------------------
## NAAM                LAND HOOGTE  41A  41B  41C
 1 Tadeusz Slusarski   POL  5.50    1    0    3
 2 Antti Kalliomaki    FIN  5.50    1    0    5
 3 David Roberts       USA  5.50    1    1    3
 4 Patrick Abada       FRA  5.45    1    1    3
 5 Wojciech Buciarski  POL  5.45    2    2    5
 6 Earl Bell           USA  5.45    3    2    5
-------------------------------------------------
Uitslag ``Gelijkspel `96``
-------------------------------------------------
## NAAM                LAND HOOGTE  41A  41B  41C
 1 Bart den Boer       HOL  0.00    0    0    0
 1 Bert den Boer       BEL  0.00    0    0    0
-------------------------------------------------
```

NK Programmeren 1996

RUN-LENGTH ENCODING

Opgave B

Om het verzenden van data over een netwerk te versnellen wordt vaak gebruik gemaakt van compressietechnieken. Immers, het tijdverlies dat ontstaat door een hoeveelheid data te comprimeren (en bij aankomst weer te decomprimeren) wordt ruimschoots goedgemaakt door de tijd die gewonnen wordt bij het versturen van een kleiner datapakket. Als de te versturen pak-ketjes gemiddeld kleiner worden, kan ook meer data tegelijkertijd over een netwerk verstuurd

worden. De bandbreedte wordt zo als het ware vergroot.

Echter, het loslaten van een compressie-algoritme op een hoeveelheid data heeft niet altijd het gewenste effect, vermindering van de hoeveelheid data. Voor elk algoritme bestaan namelijk bepaalde categorieën invoer die een even grote of zelfs grotere uitvoer tot gevolg heeft. Immers, als er een algoritme zou zijn waarvoor dit niet geldt, dan kun je voor elke invoer de (kleinere) uitvoer van dit algoritme weer als invoer gebruiken, wat weer een kleinere uitvoer als gevolg heeft, enzovoorts.

Een compressietechniek die toegepast kan worden als de te versturen data voor het grootste deel uit nullen en slechts af en toe een een bestaat, is run-length encoding. Bij deze techniek worden niet steeds al deze nullen één voor één verzonden, maar wordt er een getal verzonden dat aangeeft hoeveel nullen er volgen tot de volgende een (of het einde van het bericht).

Een voorbeeld:

invoerstring: 00010000000000000010000110

gecodeerde data: 3 14 4 0 1

te versturen (3 bits/getal): 011 111 111 000 100 000 001

Het versturen van de gecodeerde getallen in groepjes van n, in dit geval 3, bits verdient nog enige uitleg. Als een getal (in bovenstaand voorbeeld 14) niet past in n bits, wordt het in meerdere groepjes van n verstuurd. Om het versturen van grotere getallen mogelijk te maken, wordt als een groepje bits uitsluitend uit enen bestaat, het erop volgende groepje erbij opgeteld. Het einde van het getal wordt dan gemarkeerd door het eerste groepje dat niet uitsluitend uit enen bestaat. In dit geval is dit een groepje van 3 nullen. Deze horen dus nog bij hetzelfde getal! Op deze manier wordt 14 (7+7+0) gevormd.

Een probleem is nog het kiezen van het aantal bits n, waarin deze getallen steeds verzonden worden. Bij een te grote of te kleine n wordt het te versturen pakketje onnodig lang. Bijvoorbeeld:

n=2 11 00 11 11 11 11 10 11 01 00 01 (22 bits)

n=3 011 111 111 000 100 000 001 (21 bits)

n=4 0011 1110 0100 0000 0001 (20 bits)

n=5 00011 01110 00100 00000 00001 (25 bits)

In dit geval is het optimaal om de getallen in groepjes van n=4 bits te versturen. Bij elke andere waarde voor n moet een langer datapakket verstuurd worden, langer vaak dan het oorspronke-lijke bericht!

Merk op, dat als een invoerstring met een 1 begint of op een 1 eindigt, er als eerste respectieve-lijk laatste getal een extra 0 moet worden verzonden. De string ‘1000001’ wordt zo bijvoorbeeld gecodeerd in de getallen ‘0 5 0’.

Het is nu jouw taak om, gegeven een bepaalde invoerstring bestaande uit enen en nullen, het minimum aantal bits te geven waarin die string verstuurd kan worden, als gebruik gemaakt wordt van de run-length encoding compressietechniek, alsmede het aantal bits n \xb3 1 waarin de getallen in dit geval gegroepeerd worden.

Invoer

De invoer begint met een regel waarop een getal m staat. Er volgen nu m regels, elk een testcase bevattend. Deze regels bevatten een invoerstring bestaande uit uitsluitend de karakters ‘`0`’ en ‘`1`’, met een lengte \xb3 1 en # 1000.

Uitvoer

Druk, voor elke testcase, de minimale lengte en de optimale n op een regel af, gescheiden door een spatie. Als voor meerdere n dezelfde minimale lengte kan worden verkregen, moet de kleinste n gegeven worden.

Voorbeeldinvoer

```
2

00010000000000000010000110

1000000000000000001
```

Voorbeelduitvoer

```
20 4

15 3
```

NK Programmeren 1996

FLIPPO-RAADSELS

Opgave C

Fabrikanten van zoutjes op aardappelbasis stellen alles in het werk om de aandacht van het pu-bliek te trekken en zo hun omzet te vergroten. Zo begon chipsproducent Smoky op een gegeven moment kleine platte schijfjes, de zogenaamde ‘‘flippo’s’’, toe te voegen aan zijn zakken chips. Deze flippo’s en dus ook de zakken Smoky chips vonden gretig aftrek bij het jeudig deel van het Nederlandse chipseterscontingent, uiteraard tot groot genoegen van de fabrikant. Echter, om de aandacht van de chipsconsumentjes vast te houden moest Smoky meer en meer flippo’s bij zijn zakken chips leveren, wat weer extra kosten met zich meebracht.

Om ook het wat meer intellectueel gerichte chipspubliek te stimuleren om vooral zoutjes van fabrikant Smoky te kopen, werd besloten om flippo’s met ‘‘raadsels’’ te introduceren: een spelletje waarbij gegeven vier getallen, het getal 24 moest worden gevormd met behulp van de vier operatoren +,-,* en /. Deze raadsels waren soms behoorlijk moeilijk en zetten de chipseter aan tot puzzelen. Tijdens het puzzelen werd uiteraard chips genuttigd.

Langzaam maar zeker echter werd de chipseter meer en meer getraind in het oplossen van de flippo-raadsels, zodat er weer minder chips gegeten werd tijdens het puzzelen. Slimme marke-ting-medewerkers van Smoky bedachten echter een nieuw, moeilijker soort flippo-raadsels, die weer een stijgende lijn in de chipsafzet teweeg moeten gaan brengen.

Door een ouderwets gevalletje van bedrijfsspionage is grote concurrent Crits echter achter deze plannen gekomen. Het plan is nu, om verdere afkalving van het marktaandeel te voorkomen, een diskette in iedere zak met Crits chips te stoppen met daarop een programma dat de nieuwe flippo-raadsels kan oplossen voor de chips-consument. Het gevolg hiervan zal zijn, dat deze om de diskette in handen te krijgen, ook Crits chips zal gaan kopen en bovendien dat het geen uit-daging meer is om deze raadsels op te lossen, waardoor elke chipseter met onmiddelijke ingang weer over zal schakelen op Crits chips, daar deze natuurlijk het lekkerst smaken (en kraken).

Crits vraagt nu aan iedere kundige programmeur, om het benodigde programma te schrijven. Dit programma moet het volgende flippo-raadsel oplossen:

Gegeven een oneindige voorraad van het getal N. Hoeveel van deze getallen zijn nodig om het getal M te verkrijgen, waarbij gebruik mag worden gemaakt van de operatoren +, -, *, div en mod?

De operatoren +, - en * zijn de normale optelling, aftrekking en vermenigvuldiging. A div B (A,B > 0) is het quotient van A en B, indien nodig om een geheel getal te verkrijgen naar beneden

afgerond. A mod B (A,B > 0) is de rest na deling van A door B.

Bijvoorbeeld, als N = 3 en M = 4, is het antwoord op het raadsel 3. Er zijn dus minimaal 3 drieën nodig om het getal 4 te maken: (3 + (3 div 3) = 4).

Verder is Crits er ook achter gekomen, dat 0 # N # 9, 0 # M # 10000 en dat geen enkele subexpressie van de berekening een waarde kleiner dan -10000 of groter dan 10000 mag hebben, dit om het de chipspuzzelaar niet al te moeilijk te maken. In bovenstaand voorbeeld bevat de berekening drie maal de subexpressie ‘3’ (met waarde 3) en eenmaal de subexpressie ‘3 div 3’ (met waarde 1).

Invoer

De eerste regel van de invoer bevat een getal K. Hierna volgen K regels. Elk van die regels bevat de getallen N en I (0 # I # 50), gevolgd door I getallen Mi. Elk paar (N, Mi) vormt een testcase.

Uitvoer

Voor iedere testcase (N, M) moet de volgende regel worden afgedrukt:

Er zijn X `getallen nodig.`

Waarbij X het minimaal aantal N-en is dat nodig is om volgens bovenstaande regels het getal M te vormen. Als het niet mogelijk is het getal M te vormen uit louter N-en, moet het volgende worden afgedrukt:

```
Onmogelijk.
```

Voorbeeldinvoer

```
1

3 3 4 5 6
```

Voorbeelduitvoer

```
Er zijn 3 getallen nodig.

Er zijn 4 getallen nodig.

Er zijn 2 getallen nodig.
```

NK Programmeren 1996

STAARTDELING

Opgave D

Iedereen herinnert zich wel de vervelende, soms ellenlange staartdelingen die je vroeger op school moest maken. Tegenwoordig heeft iedereen een zakjapanner waarop een eenvoudige deling in nanoseconden kan worden uitgevoerd, maar nog steeds vormen staartdelingen een verplicht onderdeel van het rekenonderwijs.

Om het leed voor de huidige basisscholers enigszins te verzachten, wordt nu aan jou gevraagd een programma te schrijven dat een staartdeling uitprint, zodat zij dit nare werkje niet meer hoe-ven te doen.

Invoer

De eerste regel van de invoer bevat een getal N. Hierna volgen N regels die ieder een testgeval bevatten. Een testgeval bestaat uit twee getallen n en m, 1 # m # n # 30000. n is de teller van de deling, m de noemer.

Uitvoer

Voor ieder testgeval moet de bijbehorende staartdeling worden afgedrukt, in het formaat zoals dat in de voorbeelden gebruikt is, gevolgd door een lege regel.

Voorbeeldinvoer

```
3

99 99

100 3

100 10
```

Voorbeelduitvoer

```
99/99\1

   99

   --

    0

3/100\33

   9
```

```
  --
   10
    9
   --
    1
10/100\10
   10
   --
    00
     0
    --
     0
```

NK Programmeren 1996

PRIEMWOESTIJNEN

Opgave E

Een priemgetal is een positief getal dat alleen deelbaar is door zichzelf en door het getal 1. De verdeling van de priemgetallen onder de positieve getallen is uiterst grillig. Er is wel een asymptotische benadering van het aantal priemgetallen in de verzameling {2, .., n}: P(n) \xbb x / ln(x)

(1896: Hadamard/De la Vallee Poussin), maar dat zegt niets over het exacte aantal priemgetallen in een gegeven segment van de positieve getallen.

Er zijn situaties waar twee priemgetallen zeer dicht tegen elkaar aan liggen, de zogenaamde priemtweelingen. Een priemtweeling is een paar (p, p+2) waarbij p en p+2 beide priem zijn. Hoeveel priemtweelingen er zijn is nog steeds niet bewezen, maar alles wijst erop dat het er oneindig veel zijn.

Omgekeerd zijn er lange segmenten van de natuurlijke getallen die geen enkel priemgetal bevatten; [1330 .. 1360] is zo'n segment. Een dergelijk segment noemen we een priemwoestijn.

Gevraagd wordt een programma te schrijven dat bij invoer van twee natuurlijke getallen x en y als uitvoer het kleinste getal a \xb3 x oplevert dat voldoet aan: het segment [a .. a+y] is een priemwoestijn.

Er geldt: 1 # x # 20000; 0 # y < 50; het resulterende segment [a .. a+y] is een deelsegment van [1 .. 20000].

Invoer

De invoer begint met een getal N, het aantal testgevallen. Hierna volgen N regels die ieder twee getallen x en y bevatten.

Uitvoer

Voor ieder testgeval dient de volgende regel uitgevoerd te worden:

[a .. b] is de eerste priemwoestijn.

waarbij a en b=a+y door de juiste getallen vervangen moeten worden.

Voorbeeldinvoer

```
2

1000 9
```

```
1000 19
```

Voorbeelduitvoer

```
[1070 .. 1079] is de eerste priemwoestijn.

[1130 .. 1149] is de eerste priemwoestijn.
```

NK Programmeren 1996

TAXI

Opgave F

Beschouw de volgende situatie: Op een ronde weg rijden 10 taxi's met een snelheid van 60 km/uur, allen rijden in dezelfde richting. De taxi's zijn genummerd van 1 tot en met 10. De lengte van de weg is precies 30 km en om de kilometer is een taxi-standplaats. Deze standplaatsen zijn in de rijrichting genummerd van 1 tot en met 30. Om middernacht 0:00 uur start taxi i op standplaats 3i-2.

Wanneer een lege taxi een standplaats passeert waar een groep passagiers staat te wachten dan neemt de taxi deze passagiers mee. Een groep kan slechts bij een van de andere 29 standplaatsen uitstappen. De tijdsduur van een rit in minuten is het aantal haltes dat gereden moet worden (taxi's rijden 60 km/uur) plus een minuut per vervoerde passagier (voor in- en uitstappen). Een rit voor 3 personen van halte 1 naar halte 6 duurt dus 5+3 = 8 minuten.

Wanneer meerdere groepen mensen bij een halte staan te wachten wordt de groep die het langste wacht meegenomen. Nooit wordt meer dan één groep per keer vervoerd in een taxi. Wanneer een taxi stilstaat bij een halte om een groep af te leveren, of wanneer een taxi langs een tot dan toe lege halte rijdt, en op precies hetzelfde moment arriveert een nieuwe groep bij dezelfde halte dan wordt de nieuwe groep meteen meegenomen. Als twee taxi's op hetzelfde moment langs een halte komen, dan stopt de taxi met het laagste nummer. Als er meerdere groepen mensen staan, wordt de groep die het eerste gearriveerd is door de taxi met het laagste nummer mee-genomen. Per tijdseenheid arriveert maximaal één groep mensen bij een halte. Jouw taak is gedurende een dag de ritten van de taxi's bij te houden en de situatie weer te geven wanneer alle passagiers vervoerd zijn.

Invoer

De invoer begint met een geheel getal, het aantal runs R. Hierna volgt R maal een run, die begint met een regel met daarop een geheel getal: het aantal ritten T. Daarna volgen T regels die bestaan uit een tijd weergegeven in uren en minuten (`uu:mm`), dit is het moment waarop een groep arriveert. Daarna het aantal personen in de groep (elke groep bestaat uit tenminste 1 persoon), gevolgd door de begin en eindhalten van de rit. Alle ritten vinden op dezelfde dag plaats en zijn gesorteerd op de tijd waarop groepen arriveren. De dag begint om 00:00 en eindigt om 23:59. Voor het einde van de dag zijn ook alle ritten afgelopen.

Uitvoer

De uitvoer per run bestaat uit twee regels, gevolgd door een lege regel.

De eerste regel bevat de tekst

Op uu:mm `zijn alle passagiers vervoerd`

waarbij uu:mm het tijdstip is waarop de laatste groep afgeleverd is. De uren (uu) en minuten (mm) moeten weergegeven worden in velden van breedte 2, waar nodig aangevuld met een 0 (Bijvoorbeeld: zes over zes in de ochtend is: 06:06). De tweede regel begint met de tekst:

```
De posities van de taxi's zijn:
```

gevolgd een spatie en daarna door de posities van de taxi's (in volgorde van taxi-nummer) op het moment dat de laatste rit is afgelopen. De posities worden gescheiden door een spatie.

Voorbeeldinvoer

```
2

1

00:00 10 30 1

4

08:11 3 12 8

08:26 2 7 10

08:27 1 7 12

08:33 5 11 26
```

Voorbeelduitvoer

```
Op 00:13 zijn alle passagiers vervoerd

De posities van de taxi's zijn: 14 17 20 23 26 29 2 5 8 1

Op 08:53 zijn alle passagiers vervoerd

De posities van de taxi's zijn: 21 27 29 26 6 9 12 15 18 21
```

NK Programmeren 1996

TEGEL

Opgave G

Harry is zijn badkamer aan het betegelen. Hiervoor gebruikt hij tegels in drie kleuren: rood, geel en blauw. Het zijn echter geen gewone tegels, het zijn speciale tegels waarvan zowel de voor als

de achterkant gebruikt kan worden. Dit is handig, omdat de voor en achterkant van een tegel verschillende kleuren kunnen hebben. Zo heeft hij bijvoorbeeld tegels die zowel voor als achter geel zijn, maar ook tegels die aan de ene kant blauw en aan de andere kant rood zijn.

Alle mogelijke kleurencombinaties komen voor, dus in totaal heeft hij 6 typen tegels (RR, GG, BB, RG, RB, BG). Wanneer hij bijna klaar is met betegelen telt hij nog een keer zijn voorraad tegels. Hij heeft er nog 12, en hij heeft ook nog precies 12 open plekken in de muur, om precies te zijn een rand van 1x12 tegels. Dan begint hij zich af te vragen hoeveel verschillende patronen hij nog kan maken met zijn tegels. Kan jij dit probleem voor het oplossen?

Invoer

De invoer begint met een regel met daarop een geheel getal R. Dit getal geeft het aantal runs aan. Vervolgens komt R maal een run die bestaat uit 12 regels. Elke regel bestaat uit twee karakters (‘`r`’, ‘`g`’ of ‘`b`’), die de kleuren op een tegel weergeven.

Uitvoer

Voor elke run moet de uitvoer bestaan uit de regel

`Het aantal mogelijke vloerpatronen is` *X*

waarbij *X* vervangen dient te worden door het aantal mogelijk patronen dat met de 12 tegels van de run gemaakt kan worden.

Voorbeeldinvoer

```
2
rg
rr
rr
rr
rr
rr
rr
rr
rr
rr
```

```
rr
rr
gr
gr
gr
gr
gr
gr
rg
rg
rg
rg
rg
rg
```

Voorbeelduitvoer

```
Het aantal mogelijke vloerpatronen is 13
Het aantal mogelijke vloerpatronen is 4096
```

NK Programmeren 1996

EEN ZICHZELF BESCHRIJVENDE RIJ

Opgave H

Een rij wordt als volgt geconstrueerd. De rij bestaat uit blokken van één of twee enen of tweeën. Een blok is een langste aaneengesloten deelrij van gelijke cijfers. Verder geldt voor de rij dat de waarde van het i-de element in de rij de lengte van het i-de blok in de rij bepaalt.

De rij start met een 1, een blok met lengte één. Dus het volgende blok moet beginnen met een 2. Dus hebben we 1 2. De 1 in de rij betekent dat het eerste blok een blok met lengte één is, de twee betekent dat het tweede blok een blok van lengte twee is. Dit geeft 1 2 2. De derde twee betekent dat het derde blok lengte 2 heeft, dus hebben we 1 2 2 1 1. Het vierde blok heeft lengte één, evenzo het vijfde blok. Dus hebben we 1 2 2 1 1 2 1. Dit spelletje kunnen we blijven herhalen: de waarde van het i-de element in de rij bepaalt de lengte van het i-de blok in de rij:

1 2 2 1 1 2 1 2 2 1 2 2 1 1 2 1 1 2 2 . . .

De vraag die T. Sellke voor deze rij stelt is de volgende: Is het aantal enen in de rij ‘asymptotisch’ gelijk aan het aantal tweeën?

De vraag waarin wij echter geïnteresseerd zijn is de volgende:

Gegeven M ($1 \# M \# 10.000.000$), hoeveel enen bevinden zich in de rij ter lengte M?

Implementatie in Pascal en C

Voor deze opgave dient niet een volledig programma, maar slechts een functie hiervan

geschreven te worden. Deze functie heeft een parameter m, en retourneert het aantal enen in de rij ter lengte m.

De interface met het juryprogramma is als volgt:

- in Pascal: `FUNCTION pascalsellke (m: longint): longint`
- in C: `unsigned long Csellke (unsigned long m)`

Voorbeeld voor Pascal

`pascalsellke(10)` moet als resultaat 5 teruggeven.

`pascalsellke(18)` moet als resultaat 9 teruggeven.

Voorbeeld voor C

`Csellke(10)` moet als resultaat 5 teruggeven.

`Csellke(18)` moet als resultaat 9 teruggeven.

Geheugenruimte

De geheugenruimte die u kunt claimen is beperkt (8 Kbytes).

U mag alleen gebruik maken van lokale variabelen:

In C:

```
unsigned long Csellke (unsigned long m)
{
        eentype i;
        eentype j[eenrange];
        /* programma tekst */
}
```

In Pascal:

```
FUNCTION pascalsellke (m: longint): longint;
VAR                                     i: eentype;
        j: ARRAY[eenrange] OF eentype;
begin
        { programma tekst }
end;
```

Het gebruik van de heap is niet toegestaan.

Bijvoorbeeld: aanroepen van procedures, functies:

- In Pascal: new, dispose, mem
- In C: malloc, calloc, alloc, free

is niet toegestaan.

In alle gevallen waarin meer dan 8Kbytes aan geheugenruimte voor data geclaimd wordt, wordt de opgave afgekeurd.

Lengte van de programmatekst

De ingeleverde programmacode mag niet meer dan 100 regels van 80 karakters bevatten.