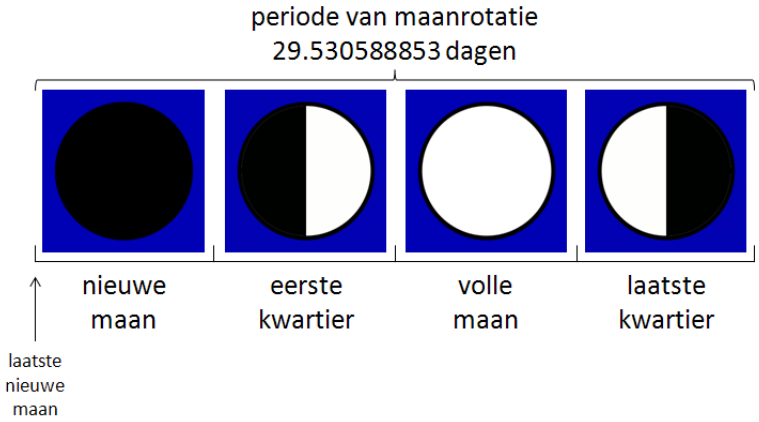
EXAMEN: **Scriptingtalen**

1e Bachelor Informatica
prof. dr. Peter Dawyndt
groep 1

maandag 21-06-2010, 8:30
academiejaar 2009-2010
eerste zittijd

Opgave 1

Onze maan vertoont zogenaamde *schijngestalten*, wat wil zeggen dat ze zich gedurende haar omloop rond de aarde in verschillende gedaanten laat zien. Dit fenomeen wordt veroorzaakt door het feit dat de maan vanuit een andere richting door de zon wordt bestraald dan de richting waarin ze wordt waargenomen. Hierdoor lijkt het alsof de *terminator* (dit is de scheidingslijn van licht en donker) zich over de oppervlakte van de maan verplaatst, waardoor de maan te zien is als een sikkelvormig of rond lichaam. De periode waarin de maan rond de aarde draait wordt over het algemeen onderverdeeld in vier *maanfasen*: nieuwe maan, eerste kwartier, volle maan en laatste kwartier. Als je voor een gegeven datum kunt bepalen hoeveel dagen er zijn verstreken sinds de laatste nieuwe maan, dan kan je de corresponderende maanfase bepalen door te kijken in welke vierde deel van de periode van de maanrotatie dit aantal dagen valt (zie onderstaande figuur).



De maan draait in een baan om de aarde met een periode van $p = 29.530588853$ dagen. Als we weten dat er een nieuwe maan voorkwam op 6 januari 2000, dan kunnen we op basis van onderstaande formule het aantal dagen sinds de laatste nieuwe maan bepalen. We noteren dit aantal dagen als $d_n \in \mathbb{R}$.

$$d_n = d_r - p * \frac{d_r}{p}$$

Hierbij stelt d_r het aantal dagen voor sinds de referentiedatum van 6 januari 2000, en stelt de breuk een gehele deling voor (of met andere woorden $d_n = d_r \bmod p$, waarbij $p \in \mathbb{R}$). Stel dat we bijvoorbeeld de maanfase willen bepalen voor de datum 21/06/2010. Er zijn op deze datum reeds 3819 dagen verstreken sinds 6 januari 2000, waardoor in dit geval $d_n = 9.55$. Aangezien de waarde van d_n in het tweede deel van de vierdelige onderverdeling van de maanperiode valt, correspondeert deze datum dus met de maanfase *eerste kwartier*.
Gevraagd wordt om een `bash` shell script `maanfase` te schrijven. Aan het shell script moeten twee gehele getallen als argument meegegeven worden, die respectievelijk het volgnummer van een maand en een jaartal voorstellen. Indien geen argumenten aan het shell script worden meegegeven, dan wordt de huidige maand en het huidige jaar verondersteld. Het shell script moet één enkele regel teruggeven waarop de maanfasen voor de verschillende dagen van de maand zijn weergegeven. Hierbij

mag verondersteld worden dat elke maand 31 dagen telt. Voorts wordt de maanfase *nieuwe maan* voorgesteld door `@`, *eerste kwartier* door `)`, *volle maan* door `O` en *laatste kwartier* door `(`. Onderstaande sessie illustreert de werking van het shell script `maanfase`.

```
$ maanfase
102030405(6(7(8(9(10(11(12@13@14@15@16@17@18@19)20)21)22)23)24)25)26)270280290300310
$ maanfase 6 2010
102030405(6(7(8(9(10(11(12@13@14@15@16@17@18@19)20)21)22)23)24)25)26)270280290300310
$ maanfase 6 2009
1)2)3)4)5)6)7)8090100110120130140150(16(17(18(19(20(21(22(23@24@25@26@27@28@29@30)31)
```

Hierbij geeft de uitvoer `102030405(6(...` bijvoorbeeld aan dat er een volle maan voorkomt op de eerste dag van de maand, en dat de maan zich in het laatste kwartier bevindt op de zesde dag van de maand. Het shell script moet bovendien aan de volgende voorwaarden voldoen:

1. Naast de programmeerbare filter `sed` mag het shell script geen gebruik maken van andere programmeerbare filters of scriptingtalen zoals `awk`, `perl`, `python`, `ruby`, ...

2. Het shell script moet een functie `jdn` bevatten, waaraan een datum in het formaat `dd/mm/jjjj` of `dd-mm-jjjj` als argument moet meegegeven worden. Deze functie moet als resultaat het Juliaanse dagnummer dat correspondeert met deze datum naar standaard uitvoer uitschrijven. Dit is het aantal dagen dat verstreken is sinds 1 januari van het jaar 4713 voor Christus. Het Juliaanse dagnummer voor een gegeven datum `dd/mm/jjjj` kan als volgt berekend worden:

   $a \leftarrow (14 - \texttt{mm})/12$
   $y \leftarrow \texttt{jjjj} + 4800 - a$
   $m \leftarrow 12 * a - 3 + \texttt{mm}$
   $jdn \leftarrow \texttt{dd} + (153 * m + 2)/5 + 365 * y + y/4 - y/100 + y/400 - 32045$

   Alle delingen stellen hierbij gehele delingen voor. Zo moet de functie voor de datum `21/06/2010` de waarde `2455369` naar standaard uitvoer schrijven. Op basis van deze functie kan het aantal verstreken dagen sinds een referentiedatum makkelijk berekend worden als het verschil tussen de Juliaanse dagnummers van beide datums.

3. Het shell script moet een functie `maanstand` bevatten, waaraan een datum in het formaat `dd/mm/jjjj` of `dd-mm-jjjj` als argument moet meegegeven worden. Deze functie moet als resultaat het symbool dat correspondeert met de maanstand op deze datum naar standaard uitvoer uitschrijven. De manier waarop de maanstand kan berekend worden, staat hierboven beschreven. Zo moet de functie voor de datum `21/06/2010` het symbool `)` dat correspondeert met de maanfase *eerste kwartier* naar standaard uitvoer schrijven.

4. Het shell script moet de geldigheid van de meegegeven argumenten nagaan, namelijk twee argumenten die respectievelijk een geldige maand en een jaartal voorstellen of geen enkel argument. Het shell script moet een gepaste foutboodschap uitschrijven indien niet aan deze voorwaarden voldaan wordt.

Tip: Indien je er niet in slaagt om te werken met `bash` shell functies, dan kan je als alternatief ook externe `bash` shell scripts `jdn` en `maanstand` schrijven. Dit levert evenwel een kleine puntenaftrek voor de delen (2) en (3) op. Indien je ook daar niet in slaagt, dan kan je de rest van deze opgave afwerken door gebruik te maken van de meegeleverde Python scripts `jdn.py` en `maanstand.py`, die je als volgt kunt gebruiken:

```
$ jdn.py 21/6/2010
2455369
$ jdn.py 6/1/2000
2451550
$ maanstand.py 21/6/2010
)
$
```

Tip: Als je gebruik maakt van het commando `bc` om te rekenen met reële getallen, dan zal je merken dat het commando `echo "4/3"|bc` als resultaat de waarde `1` teruggeeft. De waarde die je toekent aan de `bc`-variabele `scale` geeft aan hoeveel cijfers na de komma het resultaat moet bevatten. Deze waarde moet toegekend worden voor het uitvoeren van de berekening en is standaard ingesteld op nul. Het commando `echo "scale=2;4/3"|bc` zal dus de waarde `1.33` teruggeven.

Tip: De GNU versie van het commando `bc` kan ook booleaanse expressies evalueren. Het commando `echo "7>3|bc"` geeft de waarde `1` terug, en het commando `echo "7<3|bc"` geeft de waarde `0` terug. Op `genix` vind je een GNU versie van het commando `bc` met als absolute padnaam `/users/p/pdawyndt/bc`.

Opgave 2

Veronderstel dat de uitvoer van het commando `maanfase <maand> <jaar>` (zie vorige opgave) onmiddellijk gevolgd wordt door de uitvoer van de GNU versie van het commando `cal <maand> <jaar>`, in beide gevallen voor dezelfde maand en hetzelfde jaar. Gevraagd wordt om voor een dergelijke uitvoer een `sed` script `maanfase.sed` te schrijven dat de informatie gebruikt die door het shell script `maanfase` wordt aangeleverd, om de symbolen die de maanfasen voorstellen na de corresponderende dagen te plaatsen in de uitvoer van het commando `cal`. Onderstaande sessie illustreert hoe dit `sed` script kan gebruikt worden om een maandkalender weer te geven die wordt aangevuld met maanfasen.

```
$ m=6;j=2010;(maanfase $m $j; cal $m $j)
1020304O5(6(7(8(9(10(11(12@13@14@15@16@17@18@19)20)21)22)23)24)25)26)27O28O29O30O31O
     June 2010
Su Mo Tu We Th Fr Sa
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30

$ m=6;j=2010;(maanfase $m $j; cal $m $j) | gsed -f maanfase.sed
          June 2010
Sun Mon Tue Wed Thu Fri Sat
          1O  2O  3O  4O  5(
 6(  7(  8(  9( 10( 11( 12@
13@ 14@ 15@ 16@ 17@ 18@ 19)
20) 21) 22) 23) 24) 25) 26)
27O 28O 29O 30O

$ m=6;j=2009;(maanfase $m $j; cal $m $j) | gsed -f maanfase.sed
          June 2009
Sun Mon Tue Wed Thu Fri Sat
     1)  2)  3)  4)  5)  6)
 7)  8O  9O 10O 11O 12O 13O
14O 15( 16( 17( 18( 19( 20(
21( 22( 23@ 24@ 25@ 26@ 27@
28@ 29@ 30)

$
```

Om een verzorgde alignering te behouden bij de weergave van de kalender na het toevoegen van de symbolen die de maanfasen voorstellen, moet het `sed` script de volgende acties ondernemen:

- Vervang drie opeenvolgende spaties door vier spaties op de regels die de dagen van de week weergegeven. Dit zijn immers posities in de kalender waarop geen dagen vallen in de weergegeven maand, en dus moet ook daar een extra spatie toegevoegd worden om de kolommen die in de oorspronkelijke kalender drie karakters breed zijn nu vier karakters breed te maken.
- Voeg vier extra spaties toe aan de eerste regel van de kalender, zodat de maand en het jaar van de weergegeven kalendermaand weer netjes gecentreerd staan.

- Dagen van de week worden afgekort tot drie letters in plaats van twee. Zorg ervoor dat dit ook blijft werken indien het commando `cal` aangeroepen wordt met de optie `-s`, die de kalender weergeeft met een gekozen weekdag als eerste dag van de week (bv. optie `-s1` geeft maandag in plaats van zondag weer als eerste dag van de week). Dit wordt hieronder geïllustreerd.

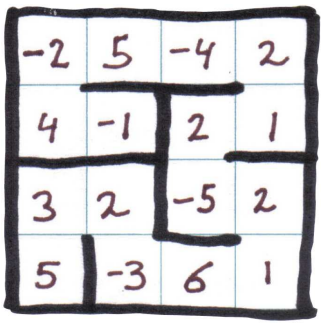
```
$ m=6;j=2010;(maanfase $m $j; cal -s1 $m $j) | gsed -f maanfase.sed
          June 2010
Mon  Tue  Wed  Thu  Fri  Sat  Sun
     1O   2O   3O   4O   5(   6(
 7(   8(   9(  1O(  11(  12@  13@
14@  15@  16@  17@  18@  19)  2O)
21)  22)  23)  24)  25)  26)  27O
28O  29O  3OO

$
```

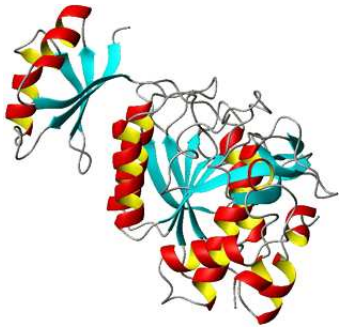
Tip: Op `genix` is het commando met de absolute padnaam `/users/p/pdawyndt/cal` een GNU versie van het commando `cal`.

Tip: Indien je de vorige opgave (nog) niet hebt opgelost, dan kan je gebruik maken van de meegeleverde bestanden `juni2009.txt` en `juni2010.txt` die de samengevoegde uitvoer van de commando's `maanfase` en `cal` bevatten, respectievelijk voor de maanden juni 2009 en juni 2010. Zorg er in ieder geval voor dat je `sed` script correct werkt voor deze invoerbestanden.

Opgave 3

Een berucht probleem in de moleculaire biologie is het vouwen van eiwitten. Een eiwitmolecuul bestaat uit een keten van aminozuren die zichzelf spontaan opvouwt tot een driedimensionale vorm die energetisch het gunstigst is. Deze bindingsenergie hangt af van de onderlinge positie van alle aminozuren. Dit levert zoveel mogelijkheden op dat zelfs de modernste supercomputers hier nog op stuklopen.
We gaan een eiwitmolecuul simuleren dat zich in twee dimensies opvouwt. De begintoestand is een vierkant rooster, waarin elk aminozuur een positieve of negatieve lading heeft. Het opvouwen gebeurt door schotjes tussen de hokjes te plaatsen, zodanig dat een keten door het rooster ontstaat die alle hokjes omvat. De uiteinden van het molecuul hoeven niet aan de rand van het rooster te liggen.
Overal waar een schotje tussen twee hokjes staat, is het molecuul gevouwen en raken de aminozuren elkaar. Als tussen een hokje met lading -3 en een hokje met lading 4 een schotje staat, dan levert dit een bindingsenergie $-3 \times 4 = -12$ op. Een aminozuur kan aan meerdere aminozuren raken, en levert dan met elke buur bindingsenergie op. Uiteraard kan, vanwege de mintekens, bindingsenergie zowel positief als negatief zijn.





Bij wijze van voorbeeld staat hierboven een 4×4 rooster waarin een eiwit met 16 aminozuren is opgevouwen tot een configuratie met bindingsenergie gelijk aan

$$(-1 \times 2) + (2 \times -5) + (5 \times -3) + (4 \times 3) + (5 \times -1) + (-1 \times 2) + (-4 \times 2) + (-5 \times 6) + (1 \times 2) = -58$$

Gevraagd wordt om een `awk` script `eiwitvouwen.awk` te schrijven waaraan twee bestanden als argument moeten meegegeven worden:

- Het eerste bestand bevat een $n \times n$ rooster dat de begintoestand van een eiwit aangeeft. Elke rij van het rooster staat op een afzonderlijke regel, en de getallen op elke rij worden van elkaar gescheiden door spaties.
- Het tweede bestand bevat alle mogelijke vouwconfiguraties voor een $n \times n$ rooster. Elke vouwconfiguratie staat op een afzonderlijke regel, en moet als volgt geïnterpreteerd worden. Stel dat we de rijen en kolommen van het rooster nummeren vanaf 1, dan kunnen de schotjes uit de configuratie van het voorbeeld op de volgende manier voorgesteld worden:

  `2,2|2,3;3,2|3,3;4,1|4,2;2,1|3,1;1,2|2,2;2,2|3,2;1,3|2,3;3,3|4,3;2,4|3,4`

  Hierbij wordt een schotje tussen de twee hokjes (r_1, k_1) en (r_2, k_2) voorgesteld als $r_1,k_1|r_2,k_2$ (waarbij r en k respectievelijk staan voor rij en kolom), en worden verschillende schotjes van elkaar gescheiden door middel van een puntkomma (`;`). De volgorde waarin de schotjes worden opgesomd is hierbij van geen belang.

Op basis van de gegeven invoerbestanden moet het `awk` script de optimale configuraties bepalen. De optimale configuraties zijn die configuraties waarvoor de bindingsenergie minimaal is. Het is dus best mogelijk dat er meerdere optimale configuraties zijn. Onderstaande sessie geeft aan hoe het `awk` script moet gebruikt worden, en hoe de uitvoer door het script moet opgemaakt worden.

```
$ awk -f eiwitvouwen.awk begintoestand1.txt configuraties_4.txt
De optimale bindingsenergie -86 wordt bekomen met de volgende configuratie(s):

+-----+-----+-----+-----+
| -2      5 | -4      2 |
+     +     +     +     +
|  4  | -1      2 |  1  |
+     +-----+-----+     +
|  3  |  2     -5 |  2  |
+     +     +-----+     +
|  5     -3 |  6      1 |
+-----+-----+-----+-----+
$ awk -f eiwitvouwen.awk begintoestand2.txt configuraties_4.txt
De optimale bindingsenergie -163 wordt bekomen met de volgende configuratie(s):

+-----+-----+-----+-----+
| -4      7 |  9      8 |
+     +     +     +-----+
| 10  | -7  |  8      0 |
+     +-----+-----+     +
| -5  |  7     -2 |  0  |
+     +     +     +     +
|  6      3 |  0      5 |
+-----+-----+-----+-----+

+-----+-----+-----+-----+
| -4      7 |  9      8 |
+     +     +     +     +
| 10  | -7  |  8  |  0  |
+     +-----+-----+     +
| -5  |  7     -2 |  0  |
+     +     +     +     +
|  6      3 |  0      5 |
+-----+-----+-----+-----+
$
```

Het `awk` script `eiwitvouwen.awk` moet de volgende twee functies bevatten:

- Een functie `bindingsenergie` die voor een gegeven configuratie van schotjes — die onder de vorm van een tekenreeks aan de functie wordt doorgeven — de corresponderende bindingsenergie teruggeeft. Zo moet de functie voor de tekenreeks die correspondeert met de configuratie uit bovenstaand voorbeeld de waarde -58 teruggeven.
- Een functie `toon_configuratie` die een gegeven configuratie van schotjes — die onder de vorm van een tekenreeks aan de functie wordt doorgeven — uitschrijft naar standaard uitvoer. Hierbij mag verondersteld worden dat aminozuren een lading l hebben die gelegen is in het interval $-9 \leq l \leq 10$. Vertikale schotjes moeten weergeven worden door een vertikale streep (`|`) en horizontale schotjes door een reeks van 5 koppeltekens (`-`). Op de plaatsen binnen het rooster waar geen schotjes voorkomen, moeten spaties gezet worden.

Opgave 4

Gegeven is een Excel werkmap `wereldbeker2010.xlsm` waarin reeds twee werkbladen werden aangemaakt. Op het werkblad `invulformulier` kan het scoreverloop van de wedstrijden op de FIFA wereldbeker voetbal 2010 in Zuid-Afrika worden ingevuld. Dit werkblad is voorzien van gepaste Excel formules, zodat landen die na de groepsfase tegen elkaar uitkomen automatisch worden ingevuld. Het werkblad `rangschikking` bevat de FIFA rangschikking van de verschillende landen bij aanvang van het wereldkampioenschap. Gevraagd wordt om VBA programmacode te schrijven die een voorspelling maakt wie de toekomstige wereldkampioen zal worden, rekening houdend met de sterkte van de verschillende landenploegen. Hiervoor ga je als volgt te werk:

1. Schrijf een functie `randpoisson` in VBA die een willekeurig natuurlijk getal teruggeeft dat gegenereerd werd op basis van een Poissonverdeling met gemiddelde waarde $\lambda \in \mathbb{R}$. De waarde van λ moet als parameter aan deze functie doorgegeven worden. Onderstaande pseudocode geeft aan hoe een dergelijke Poisson verdeelde willekeurige waarde kan geproduceerd worden:

   stel $L \leftarrow e^{-\lambda}$, $k \leftarrow 0$ en $p \leftarrow 1$
   do:
   $k \leftarrow k + 1$
   genereer uniform verdeeld willekeurig getal $u \in [0, 1[$
   $p \leftarrow p \times u$
   while $p > L$
   return $k - 1$

2. Schrijf een functie `score` in VBA die de eindstand van een wedstrijd tussen twee landenploegen l_1 en l_2 simuleert, rekening houdend met de sterktes van beide ploegen. De namen van de twee landen die tegen elkaar uitkomen moeten als argument aan de functie doorgegeven worden, en de functie moet als resultaat een tekenreeks teruggeven waarbij het aantal gescoorde doelpunten van beide landen wordt gescheiden door een koppelteken (bv. `4-2`). De simulatie van de wedstrijd gebeurt op de volgende manier:

   (a) Het aantal gescoorde doelpunten d is een willekeurig waarde die genereerd wordt op basis van een Poissonverdeling met $\lambda = 2.3$. Tijdens het voorgaande wereldkampioenschap voetbal in 2006 werden immers gemiddeld 2.3 doelpunten per wedstrijd gescoord.

   Tip: Indien je er (nog) niet in geslaagd bent om de functie `randpoisson` te implementeren, dan kan je (voorlopig) als aantal doelpunten bijvoorbeeld een uniform willekeurig getal $0 \leq d \leq 8$ genereren.

(b) Zoek het aantal punten p_1 en p_2 van beide landen l_1 en l_2 op in het werkblad `rangschikking` (derde kolom). Zorg er hierbij voor dat dit werkblad niet noodzakelijk het actieve werkblad hoeft te zijn.

(c) Simuleer het scoren van d doelpunten door voor elk doelpunt een uniform verdeeld willekeurig getal $u \in [0, 1[$ te generen. Als $u < \frac{p_1}{p_1+p_2}$ dan heeft land l_1 gescoord, anders heeft land l_2 gescoord.

3. Schrijf een subprocedure `pronostiek` in VBA. Deze procedure moet eerst en vooral een kopie maken van het werkblad `invulformulier` binnen de werkmap `wereldbeker2010.xlsm` (die als de actieve werkmap mag beschouwd worden). Daarna moet de subprocedure op het gekopieerde invulformulier voor alle wedstrijden van het wereldkampioenschap een score invullen die gegenereerd werd volgens de procedure die hierboven werd omschreven. De gesimuleerde eindstanden van de wedstrijden uit de groepsfase komen in kolommen 7 en 8 van het invulformulier terecht. Voor de wedstrijden die gespeeld worden na de groepsfase moeten de gesimuleerde eindstanden worden ingevuld in kolommen 56 en 57. Omdat elke wedstrijd na de groepsfase een winnaar moet opleveren, moet je voor deze wedstrijden vermijden dat er een gelijkspel gesimuleerd wordt. Tenslotte moeten alle cellen met de scores van de zeven wedstrijden die door de wereldkampioen (zie 47^{e} rij en 56^{e} kolom) gespeeld werden een groene achtergrondkleur krijgen met RGB-waarde (153,255,102).

   Tip: De cellen van het invulformulier waarin een aantal doelpunten moet ingevuld worden, hebben een achtergrondkleur met RGB-waarde (255,255,153).

EXAMEN: **Scriptingtalen**

1e Bachelor Informatica
prof. dr. Peter Dawyndt
groep 2

maandag 21-06-2010, 14:00
academiejaar 2009-2010
eerste zittijd

Opgave 1

Een wereldkampioenschap voetbal begint steeds met een groepsfase, waarbij de deelnemende landen over een aantal groepen verdeeld worden. Het aantal groepen en het aantal ploegen per groep varieert van kampioenschap tot kampioenschap, maar elke groep krijgt steeds een uniek label toegekend (meestal een letter uit het alfabet). Binnen elke groep speelt elk land eenmaal tegen elk ander land. Gevraagd wordt om een `awk` script `overzicht.awk` te schrijven waarmee op basis van een lijst van gespeelde wedstrijden een overzicht kan opgemaakt worden van de stand binnen elke groep. In dit overzicht moeten de groepen alfabetisch gesorteerd op hun label weergegeven worden.
De lijst van gespeelde wedstrijden is opgeslagen in een bestand waarin voor elke wedstrijd op een afzonderlijk regel de volgende informatie gegeven wordt: *i*) thuisploeg, *ii*) uitploeg, *iii*) eindstand en *iv*) groepslabel. De verschillende informatievelden van een wedstrijd worden gescheiden door komma's en bij de eindstand wordt het aantal gescoorde doelpunten van beide ploegen gescheiden door een koppelteken (`-`). Het bestand met de gespeelde wedstrijden kan verder ook lege regels of commentaarregels die starten met een hekje (`#`) bevatten. Deze moeten door het `awk` script genegeerd worden. Onderstaande sessie geeft aan hoe een voorbeeldbestand `wereldbeker2010.txt` met gespeelde wedstrijden er uitziet, hoe het `awk` script `overzicht.awk` moet gebruikt worden, en hoe de uitvoer moet opgemaakt worden.

```
$ cat wereldbeker2010.txt | grep France
Uruguay,France,1-2,A
France,Mexico,3-0,A
South Africa,France,1-1,A
$ gawk -f overzicht.awk wereldbeker2010.txt | head -n20
                        GROEP A
+-----------------+-----+-------------------------+-----+
|                 |   P |   W   L   D   F   A   S | Pts |
+-----------------+-----+-------------------------+-----+
|          France |   3 |   2   0   1   6   2   4 |   7 |
|         Uruguay |   3 |   1   1   1   3   2   1 |   4 |
|    South Africa |   3 |   1   1   1   3   4  -1 |   4 |
|          Mexico |   3 |   0   2   1   1   5  -4 |   1 |
+-----------------+-----+-------------------------+-----+

                        GROEP B
+-----------------+-----+-------------------------+-----+
|                 |   P |   W   L   D   F   A   S | Pts |
+-----------------+-----+-------------------------+-----+
|          Greece |   3 |   3   0   0   3   0   3 |   9 |
|       Argentina |   3 |   2   1   0   3   2   1 |   6 |
|         Nigeria |   3 |   1   2   0   2   3  -1 |   3 |
|     South Korea |   3 |   0   3   0   0   3  -3 |   0 |
+-----------------+-----+-------------------------+-----+

$
```

Het `awk` script moet een functie `toon_groep` bevatten, die voor een gegeven groep (waarvan het label als argument aan de functie wordt doorgegeven) een opgemaakt overzicht van de groepsstatistieken uitschrijft naar standaard uitvoer. Hierbij worden voor elk land uit de groep de volgende statistieken weergegeven: *i*) aantal gespeelde wedstrijden (`P`), *ii*) aantal gewonnen wedstrijden (`W`), *iii*) aantal verloren wedstrijden (`L`), *iv*) aantal wedstrijden die op een gelijkspel geëindigd zijn (`D`), *v*) aantal gescoorde doelpunten (`F`), *vi*) aantal tegendoelpunten (`A`), *vii*) doelsaldo (`S`) en *viii*) aantal punten

(`Pts`). Het doelsaldo is het aantal gescoorde doelpunten min het aantal tegendoelpunten. Voor elke gewonnen wedstrijd krijgt een ploeg drie punten, en voor elk gelijkspel één punt. Bij de weergave moeten de landen binnen een groep gesorteerd worden eerst volgens dalend aantal punten, en daarna volgens dalend doelsaldo. Verder moet de opmaak er uitzien zoals in bovenstaand voorbeeld wordt weergegeven, waarbij een tabel wordt getekend en het groepslabel gecentreerd boven de tabel wordt geplaatst.

Opgave 2

Bij wijze van voorbereiding op de volgende opgave wordt gevraagd om twee `sed` scripts te schrijven die de volgende opdrachten uitvoeren:

1. Het script `reduceer.sed` moet uit een gegeven bestand enkel de gebruikte hoofdletters overhouden, en deze ontdubbeld en in alfabetische volgorde naar standaard uitvoer uitschrijven. Onderstaande sessie illustreert hoe dit `sed` script kan gebruikt worden.

```
$ cat puzzel1.txt
  SEND
+ MORE
------
=MONEY
$ cat puzzel1.txt | gsed -f reduceer.sed
DEMNORSY
$ cat puzzel2.txt
 VENI
+VIDI
-----
=VICI
$ cat puzzel2.txt | gsed -f reduceer.sed
CDEINV
$
```

2. Het script `vervang.sed` moet de eerste regel van een gegeven bestand gebruiken om vervangingen door te voeren op de rest van het bestand. De interpretatie van de eerste regel bestaat erin om elk karakter op een oneven positie te vervangen door het karakter dat erop volgt. De eerste regel van een bestand dat door dit `sed` script moet verwerkt worden, moet dus altijd een even aantal karakters bevatten. Als de eerste regel bijvoorbeeld de string `D7E5M1N600R8S9Y2` bevat, dan moet elke letter `D` in de rest van het bestand vervangen worden door het cijfer `7`, elke letter `E` door het cijfer `5`, enzoverder. Karakters die reeds vervangen werden, mogen geen tweede keer vervangen worden, omdat anders een oneindige reeks vervangingen kan ontstaan. Als de eerste regel bijvoorbeeld de string `0112` bevat, dan moet elk cijfer `0` vervangen worden door het cijfer `1`, maar moet dit cijfer vervolgens niet opnieuw worden vervangen door het cijfer `2`. Uiteraard moet waar in de oorspronkelijke tekst het cijfer `1` voorkwam, dit wel worden vervangen door het cijfer `2`. Alle karakters die niet in de vervangingsreeks voorkomen mogen gewoon blijven staan. Onderstaande sessie illustreert hoe dit `sed` script kan gebruikt worden.

```
$ (echo D7E5M1N600R8S9Y2; cat puzzel1.txt) | gsed -f vervang.sed
  9567
+ 1085
------
=10652
$ cat puzzel3.txt
 1234
+5678
-----
=6912
```

```
$ (echo 01122334455667788990; cat puzzel3.txt) | gsed -f vervang.sed
 2345
+6789
-----
=7023
$
```

Opgave 3

Gevraagd wordt om een `bash` shell script `rekensom` te schrijven. Aan dit shell script moeten vier argumenten meegegeven worden, waarvan de eerste drie argumenten uitsluitend bestaan uit hoofdletters en het laatste argument enkel uit cijfers. Het shell script moet nagaan of de som van de eerste twee woorden gelijk is aan het derde woord. Daarvoor moet het shell script de cijferreeks gebruiken die als het vierde argument werd doorgegeven, om daarmee alle letters uit de woorden te vervangen door cijfers. Dit gebeurt door de alfabetisch eerste letter uit de drie woorden te vervangen door het eerste cijfer uit de reeks, de alfabetisch tweede letter door het tweede cijfer, enzoverder. Het shell script moet de som van de oorspronkelijke woorden en de som waarin de letters werden vervangen door cijfers volgens een vaste opmaak uitschrijven naar standaard uitvoer, samen met een boodschap die aangeeft of de som correct is of niet. Onderstaande sessie illustreert de werking van het shell script `rekensom`.

```
$ rekensom SEND MORE MONEY 75160892
  SEND
+ MORE
------
=MONEY

  9567
+ 1085
------
=10652

Correcte som!!
$ rekensom VENI VIDI VICI 123456
 VENI
+VIDI
-----
=VICI

 6354
+6424
-----
=6414

Foutieve som!!
$
```

Het shell script moet bovendien aan de volgende voorwaarden voldoen:

1. Naast de programmeerbare filter `sed` mag het shell script geen gebruik maken van andere programmeerbare filters of scriptingtalen zoals `awk`, `perl`, `python`, `ruby`, ...
2. Het shell script moet zoveel mogelijk gebruik maken van de scripts `reduceer.sed` en `vervang.sed` uit de vorige opgave.
3. Het shell script moet een functie `max` bevatten, die de maximale lengte naar standaard uitvoer uitschrijft van alle argumenten die aan de functie meegegeven worden. Zo moet de functie voor de argumenten `SEND MORE MONEY` de waarde `5` naar standaard uitvoer schrijven. Indien geen argumenten worden meegegeven aan de functie, dan moet de waarde `0` uitgeschreven worden.

4. Het shell script moet een functie `opmaak` bevatten, waaraan drie argumenten moeten meegegeven worden. De functie moet deze argumenten op afzonderlijke regels naar standaard uitvoer schrijven, waarbij de argumenten rechts worden uitgelijnd binnen evenveel posities als de lengte van het langste argument. Het eerste argument moet voorafgegaan worden door een extra spatie, het tweede argument door een plusteken (`+`) en het derde argument door een gelijkteken (`=`). Tussen de regels waarop het tweede en het derde argument staan, moet een regel met een reeks koppeltekens (`-`) uitgeschreven worden, met een lengte die één langer is dan de lengte van het langste argument. Zo moet de functie de volgende uitvoer genereren voor de argumenten `SEND`, `MORE` en `MONEY`:

```
  SEND
+ MORE
------
=MONEY
```

5. Het shell script moet een functie `controleer_som` bevatten, waaraan $n \geq 2$ getallen als argument moeten meegegeven worden. Deze functie moet de tekst "`Correcte som!!`" naar standaard uitvoer schrijven als de som van de eerste $n-1$ getallen gelijk is aan het laatste getal. Anders moet de tekst "`Foutieve som!!`" uitgeschreven worden. Zo moet de functie bijvoorbeeld voor de argumenten `10`, `20`, `30`, `40`, `50` en `150` de tekst "`Correcte som!!`" uitschrijven.

6. Het shell script moet de geldigheid van de argumenten nagaan: vier argumenten waarvan de eerste drie enkel uit hoofdletters bestaan en het vierde enkel uit cijfers. Bovendien moet de lengte van het vierde argument gelijk zijn aan het aantal verschillende letters in de eerste drie argumenten. Het shell script moet een gepaste foutboodschap uitschrijven indien niet aan deze voorwaarden voldaan wordt.

Tip: Indien je er niet in slaagt om te werken met `bash` shell functies, dan kan je als alternatief ook externe `bash` shell scripts `max`, `opmaak` en `controleer_som` schrijven. Dit levert evenwel een kleine puntenaftrek voor de delen (3), (4) en (5) op. Indien je ook daar niet in slaagt, dan kan je de rest van deze opgave afwerken door gebruik te maken van de meegeleverde Python scripts `max.py`, `opmaak.py` en `controleer_som.py`, die je als volgt kunt gebruiken:

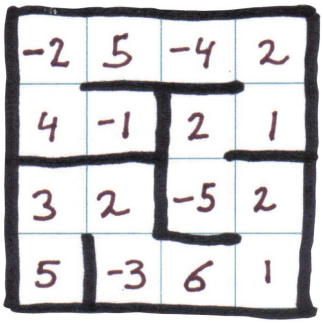
```
$ max.py SEND MORE MONEY
5
$ opmaak.py SEND MORE MONEY
  SEND
+ MORE
------
=MONEY
$ controleer_som.py 10 20 30 40 50 150
Correcte som!!
$
```

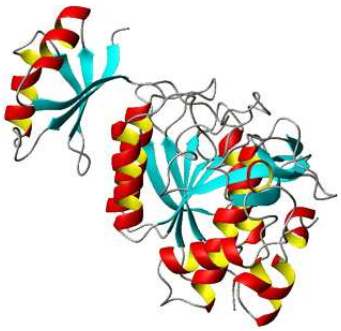
Tip: Indien je de vorige opgave (nog) niet hebt opgelost, dan kan je gebruik maken van de meegeleverde Python scripts `reduceer.py` en `vervang.py`. Deze kunnen op de volgende manier gebruikt worden ter vervanging van de `sed` scripts.

```
$ cat puzzel1.txt | reduceer.py
DEMNORSY
$ (echo D7E5M1N6O0R8S9Y2; cat puzzel1.txt) | vervang.py
  9567
+ 1085
------
=10652
$
```

Opgave 4

Een berucht probleem in de moleculaire biologie is het vouwen van eiwitten. Een eiwitmolecuul bestaat uit een keten van aminozuren die zichzelf spontaan opvouwt tot een driedimensionale vorm die energetisch het gunstigst is. Deze bindingsenergie hangt af van de onderlinge positie van alle aminozuren. Dit levert zoveel mogelijkheden op dat zelfs de modernste supercomputers hier nog op stuklopen.
We gaan een eiwitmolecuul simuleren dat zich in twee dimensies opvouwt. De begintoestand is een vierkant rooster, waarin elk aminozuur een positieve of negatieve lading heeft. Het opvouwen gebeurt door schotjes tussen de hokjes te plaatsen, zodanig dat een keten door het rooster ontstaat die alle hokjes omvat. De uiteinden van het molecuul hoeven niet aan de rand van het rooster te liggen.
Overal waar een schotje tussen twee hokjes staat, is het molecuul gevouwen en raken de aminozuren elkaar. Als tussen een hokje met lading -3 en een hokje met lading 4 een schotje staat, dan levert dit een bindingsenergie $-3 \times 4 = -12$ op. Een aminozuur kan aan meerdere aminozuren raken, en levert dan met elke buur bindingsenergie op. Uiteraard kan, vanwege de mintekens, bindingsenergie zowel positief als negatief zijn.





Als voorbeeld staat hierboven een 4×4 rooster, waarin een eiwit met 16 aminozuren is opgevouwen tot een configuratie met bindingsenergie gelijk aan

$$(-1 \times 2) + (2 \times -5) + (5 \times -3) + (4 \times 3) + (5 \times -1) + \\ (-1 \times 2) + (-4 \times 2) + (-5 \times 6) + (1 \times 2) = -58$$

De gegeven Excel werkmap `eiwitvouwen.xlsm` bevat werkbladen `begintoestand1` en `begintoestand2`, die in de linkerbovenhoek de begintoestand van een vierkant 4×4 rooster bevatten. Werkblad `begintoestand1` bevat het rooster dat hierboven ook als voorbeeld gebruikt werd. Daarnaast bevat de werkmap ook een werkblad `configuraties` dat een lijst van alle mogelijke configuraties van schotjes voor een 4×4 rooster bevat. Elke vouwconfiguratie staat op een afzonderlijke regel, en moet als volgt geïnterpreteerd worden. Stel dat we de rijen en kolommen van het rooster nummeren vanaf 1, dan kunnen de schotjes uit de configuratie van het voorbeeld op de volgende manier worden voorgesteld onder de vorm van een tekenreeks

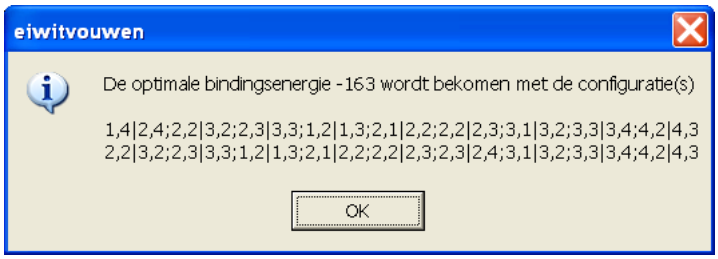
`2,2|2,3;3,2|3,3;4,1|4,2;2,1|3,1;1,2|2,2;2,2|3,2;1,3|2,3;3,3|4,3;2,4|3,4`

Hierbij wordt een schotje tussen de twee hokjes (r_1, k_1) en (r_2, k_2) voorgesteld als $r_1, k_1 | r_2, k_2$ (waarbij r en k respectievelijk staan voor rij en kolom), en worden verschillende schotjes van elkaar gescheiden door middel van een puntkomma (`;`). De volgorde waarin de schotjes worden opgesomd is hierbij niet van belang. Gevraagd wordt:

1. Schrijf een functie `bindingsenergie` in VBA die voor een gegeven configuratie van schotjes de corresponderende bindingsenergie teruggeeft. De tekenreeks die de configuratie voorstelt en de naam van het werkblad waarop het rooster staat moeten als argumenten aan de functie meegegeven worden. Zo moet de functie voor de tekenreeks die correspondeert met de voorbeeldconfiguratie (waarvan het rooster op werkblad `begintoestand1` staat) de waarde -58 teruggeven.

Zorg ervoor dat de functie ook werkt indien het Excel werkblad `configuratie` niet het actieve werkblad is.

2. Schrijf een subprocedure `optimale_bindingsenergie` die in een boodschapvenster de lijst van optimale configuraties weergeeft voor een gegeven 4×4 rooster, waarvan de naam van het werkblad waarop het rooster staat als argument wordt meegegeven aan de subprocedure. De optimale configuraties zijn die configuraties waarvoor de bindingsenergie zo klein mogelijk is. Het is dus best mogelijk dat er meerdere optimale configuraties zijn. Zorg ervoor dat het boodschapvenster eruit ziet zoals hieronder wordt aangegeven (let hierbij op het gebruikte icoontje, de weergave van de boodschap en de titeltekst van het boodschapvenster).



3. Schrijf een subprocedure `toon_configuratie` in VBA die een gegeven configuratie van schotjes grafisch weergeeft op een nieuw werkblad. De tekenreeks die de configuratie voorstelt en de naam van het werkblad waarop het rooster staat moeten als argumenten aan de functie meegegeven worden. De procedure moet eerst een kopie maken van het werkblad waarop het rooster staat, en moet daarna op dit gekopieerd werkblad de gepaste dikke lijnen trekken die de schotjes voorstellen uit de tekenreeks die de configuratie voorstelt.

	A	B	C	D
1	-2	5	-4	2
2	4	-1	2	1
3	3	2	-5	2
4	5	-3	6	1

Pas nu ook de subprocedure `optimale_bindingsenergie` aan, door daarin de subprocedure `toon_configuratie` aan te roepen om daarmee elke optimale configuratie ook grafisch op een afzonderlijk werkblad weer te geven.