

---

---

## EXAMEN: Scriptingtalen

---

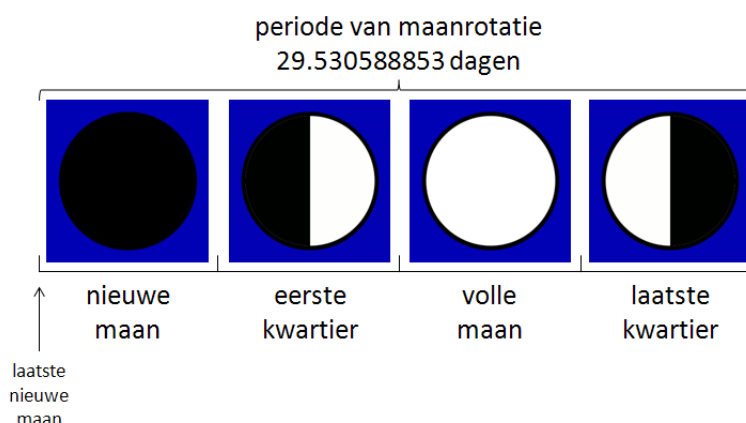
---

1<sup>e</sup> 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 0 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)809010011012013014015(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:

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

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 maanfases 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 maanfases.

```
$ m=6;j=2010;(maanfase $m $j; cal $m $j)
102030405(6(7(8(9(10(11(12@13@14@15@16@17@18@19)20)21)22)23)24)25)26)270280290300310
    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
      10  20  30  40  5(
 6(  7(  8(  9( 10( 11( 12@
13@ 14@ 15@ 16@ 17@ 18@ 19)
20) 21) 22) 23) 24) 25) 26)
270 280 290 300

$ 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)  80  90 100 110 120 130
140 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 maanfases 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 weekday 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
      10  20  30  40  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@
$
```

**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.

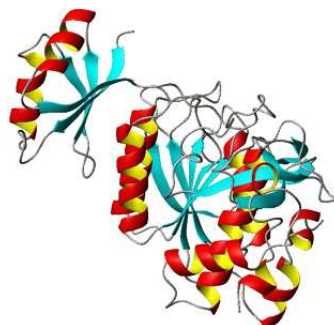
### Opgave 3

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.

Overall 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.

-2	5	-4	2
4	-1	2	1
3	2	-5	2
5	-3	6	1



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

$$\begin{aligned}
 &(-1 \times 2) + (2 \times -5) + (5 \times -3) + (4 \times 3) + (5 \times -1) + \\
 &(-1 \times 2) + (-4 \times 2) + (-5 \times 6) + (1 \times 2) = -58
 \end{aligned}$$

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 doorgegeven — 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 doorgegeven — 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 weergegeven worden door een verticale streep (|) en horizontale schotjes door een reeks van 5 koppelteken (-). 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  $\lambda$  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 gegenereerd 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<sup>e</sup> rij en 56<sup>e</sup> 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).