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:

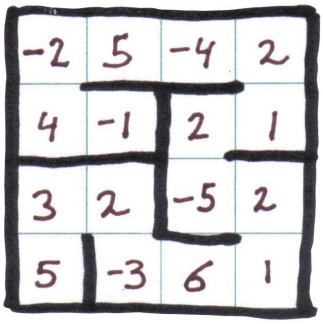
```
$ max.py SEND MORE MONEY
5
$ opmaak.py SEND MORE MONEY
  SEND
+ MORE
------
=MONEY
$ controleer_som.py 10 20 30 40 50 150
Correcte som!!
$
```

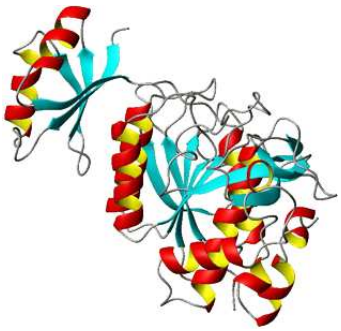
Tip: Indien je de vorige opgave (nog) niet hebt opgelost, dan kan je gebruik maken van de meegeleverde Python scripts `reduceer.py` en `vervang.py`. Deze kunnen op de volgende manier gebruikt worden ter vervanging van de `sed` scripts.

```
$ 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

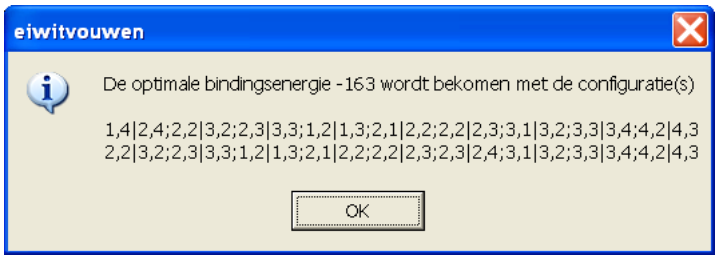
`2,2|2,3;3,2|3,3;4,1|4,2;2,1|3,1;1,2|2,2;2,2|3,2;1,3|2,3;3,3|4,3;2,4|3,4`

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.