Test Perl (3 november 2009)

Instructies

Log, voor zover dat nog niet gebeurd zou zijn, in als *administrator* (wachtwoord *e=mc**2)*.* Je creëert en test jouw oplossingen uit in de map c:\temp\tp. Je hoeft geen oplossing op papier af te geven. Enkel de versie op pc is van belang. **Enkel de bestanden 1.pl, 2.pl en 3.pl in de map c:\temp\tp** worden bij de quotering geraadpleegd. Hiervoor zijn sjablonen beschikbaar. Vul **nu reeds** de eerste lijnen van deze bestanden aan met jouw naam. Eens je de oplossingen effectief uitgewerkt hebt, vul je ook de overige headers van de bestanden in. In deze headers geef je aan in hoeverre jouw antwoord aan de gestelde eisen voldoet. Eventuele tekortkomingen van jouw oplossing geef je aan op de daarvoor voorziene lijn. Verduidelijk de oplossingen met behulp van aanvullende commentaarlijnen. Er zal praktisch **uitsluitend op het resultaat** gequoteerd worden: combinaties van `(,),{,},[,],<,>,->,\,/,@,%,!,#,^,*,+,?` en `$`-tekens die misschien wel *enigszins* op een correct antwoord lijken, maar toch niet het gevraagde resultaat opleveren, zullen geen enkele positieve bijdrage tot de quotering leveren. Lijnen met syntaxfouten worden zondermeer verwijderd. Je mag ook **geen enkele perl module** gebruiken. Na het beëindigen van de test (17u ten laatste) **log je uit**, maar laat je het toestel verder gewoon aanstaan (**geen shutdown uitvoeren** !).

Opgaven

1) De eerste 64 lijnen in de __DATA__ sectie van het bestand ***1.pl*** bevatten een recente top-64 lijst van professionele tennisspelers, lijn per lijn, in volgorde van hun ranking (dd.26/10/09). Het lijnnummer geeft bijgevolg telkens de ranking aan. De daaropvolgende lijnen bevatten een planning om een tennistornooi te organiseren in een systeem met rechtstreekse uitschakeling, waarbij de spelers vooralsnog voorgesteld zijn door een numerieke identificatie tussen 1 en 64. Ontwikkel in het ***1.pl*** bestand perl code die, vertrekkend van deze gegevens, een tornooirooster opstelt. Enerzijds is het hierbij de bedoeling dat de betere spelers elkaar pas in de latere fazes van het tornooi kunnen ontmoeten. Anderzijds moet met een *beperkte willekeur* rekening gehouden worden, zodat verschillende tornooien, die hetzelfde stramien zouden volgen, toch niet volgens strikt hetzelfde schema verlopen. Sorteer daarom de top-64 lijst (in welke vorm je de gesorteerde lijst bijhoudt, heeft geen belang) volgens een samengestelde sleutel met twee componenten:
   - De eerste sleutelcomponent, x, moet gevormd worden door het kleinst mogelijk geheel getal, waarvoor geldt dat 2^x groter of gelijk is aan de ranking van de speler (het lijnnummer in de top-64 lijst): voor de speler met ranking 1 wordt x bijgevolg 0, voor de speler met ranking 2 wordt x 1, voor de spelers met $3 \leq \text{ranking} \leq 4$ geldt $x=2$, voor de spelers met $5 \leq \text{ranking} \leq 8$ geldt $x=3$, ... Deze sleutelcomponent zal er voor zorgen dat de *top-2* spelers elkaar pas in de finale van het tornooi kunnen ontmoeten, willekeurige top-4 spelers elkaar pas in de halve finales, willekeurige top-8 spelers elkaar pas in de kwartfinales, ...
   - De tweede sleutelcomponent, y, moet *at random* bepaald worden, en zorgt hierdoor voor de beperkte willekeur in de sortering.

   Lees vervolgens de *roostergegevens* lijn per lijn in, vervang hierbij elke numerieke identificatie door de speler met dat volgnummer in de gesorteerde lijst, en schrijf de lijn uit. De lege lijnen en streepjeslijnen moet je ook reproduceren. Voor een top-32 speler moet de verwijzing tussen ronde haakjes (naar de nationaliteit) aangevuld worden met zijn ranking (het lijnnummer in de oorspronkelijke top-64 lijst), van elkaar gescheiden door een komma. Zorg er voor dat alle lijnen zowel links als rechts mooi gealligneerd blijven. Je moet bijgevolg een analoog resultaat bekomen als volgende voorbeeldoutput (kolommen sequentieel te lezen):

```
Roger Federer (SUI,1)                    Evgeny Korolev (RUS)
David Ferrer (ESP,18)                   Thomaz Bellucci (BRA)
Nicolas Almagro (ESP,28)                   Jose Acasuso (ARG)
Gael Monfils (FRA,15)                      Marc Gicquel (FRA)

Radek Stepanek (CZE,14)                 Fabrice Santoro (FRA)
Stanislas Wawrinka (SUI,21)             Benjamin Becker (GER)
Mikhail Youzhny (RUS,25)                      Dudi Sela (ISR)
Juan Martin Del Potro (ARG,5)          Janko Tipsarevic (SRB)
-------------------------------------------------------------
Andy Roddick (USA,7)                       Andreas Seppi (ITA)
Lleyton Hewitt (AUS,20)                    Andreas Beck (GER)
Juan Monaco (ARG,30)                      Fabio Fognini (ITA)
Marin Cilic (CRO,13)                   Horacio Zeballos (ARG)

Tommy Robredo (ESP,16)                       Ivo Karlovic (CRO)
Philipp Kohlschreiber (GER,24)          Maximo Gonzalez (ARG)
Ivan Ljubicic (CRO,29)                  Feliciano Lopez (ESP)
Novak Djokovic (SRB,3)                   Victor Hanescu (ROU)
...                                                      ↵
```

```
...↵
Andy Murray (GBR,4)                        Marcos Baghdatis (CYP)
Tommy Haas (GER,17)                  Martin Vassallo Arguello (ARG)
Tomas Berdych (CZE,19)                     Daniel Koellerer (AUT)
Robin Soderling (SWE,10)                    Albert Montanes (ESP)

Fernando Verdasco (ESP,9)                    Jurgen Melzer (AUT)
Viktor Troicki (SRB,31)                    Dmitry Tursunov (RUS)
Sam Querrey (USA,26)                            Mardy Fish (USA)
Nikolay Davydenko (RUS,6)                   Olivier Rochus (BEL)
-------------------------------------------------------------------
Jo-Wilfried Tsonga (FRA,8)                       John Isner (USA)
James Blake (USA,23)                           Igor Andreev (RUS)
Jeremy Chardy (FRA,32)                      Julien Benneteau (FRA)
Fernando Gonzalez (CHI,11)                       Simon Greul (GER)

Gilles Simon (FRA,12)                Guillermo Garcia-Lopez (ESP)
David Nalbandian (ARG,27)                        Pablo Cuevas (URU)
Juan Carlos Ferrero (ESP,22)                      Jan Hernych (CZE)
Rafael Nadal (ESP,2)                    Paul-Henri Mathieu (FRA)
```

2) De __DATA__ sectie van het bestand ***2.pl*** bevat, in *xml* formaat, informatie in verband met de Belgische voetbalcompetitie in eerste klasse. Uit een *voetballer* element, bijvoorbeeld,

```
<voetballer id="THOMASBUFFEL" nationaliteit="BEL">
  <naam>Thomas Buffel</naam>
  <foto>THOMASBUFFEL.jpg</foto>
  <positie>Middenvelder</positie>
  <werkgevers>
    <werkgever naam="CER">2009</werkgever>
    <werkgever naam="GNK">2009</werkgever>
  </werkgevers>
</voetballer>
```

kun je voor elke speler, steeds in dezelfde volgorde, de *nationaliteit*, *naam*, *positie* en *ploegidentificatie* halen. Hou voor dit laatste enkel rekening met het laatste *werkgever* element in het *voetballer* element. Het verband tussen de *ploegidentificatie* en de *ploegnaam* kun je halen uit de diverse *club* elementen, bijvoorbeeld,

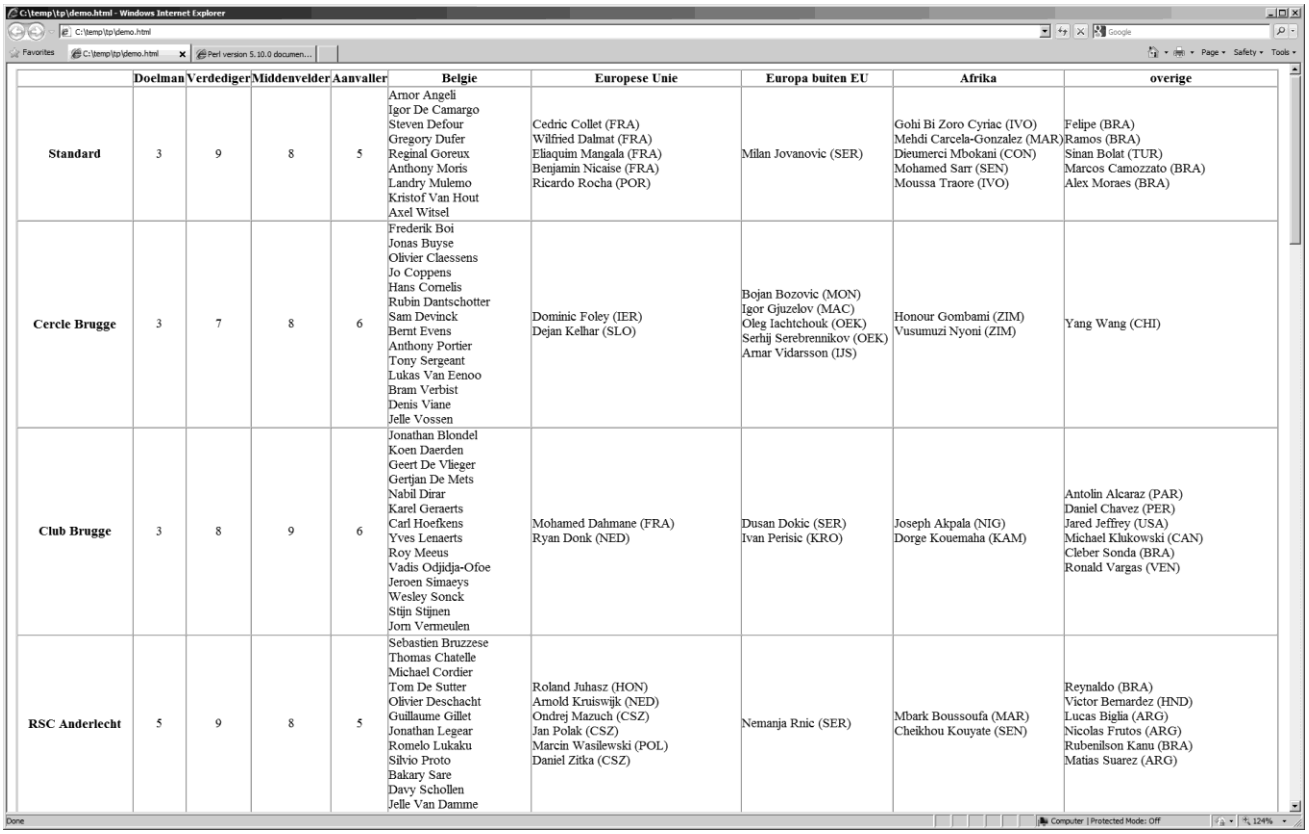
```
<club id="GNT">
  <naam>AA Gent</naam>
  <clubkleuren>Blauw-Wit</clubkleuren>
  <oprichtingsdatum>10/12/1891</oprichtingsdatum>
  <logo>gent.jpg</logo>
  <website>www.kaagent.be</website>
  <stadion>
    <naam>Jules Ottenstadion</naam>
    <capaciteit>12919</capaciteit>
    <adres>
      <straat>Bruiloftstraat</straat>
      <nummer>42</nummer>
      <gemeente>
        <naam>Gentbrugge</naam>
        <postcode>9050</postcode>
      </gemeente>
    </adres>
  </stadion>
</club>
```

die ondermeer ook de *capaciteiten* van de stadions bevatten. Ontwikkel in het ***2.pl*** bestand perl code die op basis van deze gegevens, zonder extra perl modules te gebruiken, en zo efficiënt mogelijk gebruik makend van reguliere expressies, een tweedimensionale tabel in HTML formaat produceert, met volgende specificaties:

- De eerste rij toont de aard van de inhoud van de corresponderende kolommen. De informatie op deze rij moet uit de beschikbare gegevens afgeleid worden, en mag bijgevolg niet hardgecodeerd worden !
- De volgende rijen tonen, één rij per ploeg, informatie in verband met elke voetbalploeg. Noch het aantal ploegen, noch de namen ervan mogen hardgecodeerd worden. De rijen moeten gesorteerd worden, in dalende volgorde, op de *capaciteit* van het stadion van de overeenkomstige ploeg.
- De eerste kolom bevat telkens de *ploegnaam*.
- De volgende kolommen tellen het aantal spelers die de ploeg voor elke specifieke *positie* in dienst heeft. Het aantal *posities* (en het corresponderende aantal kolommen) mag opnieuw niet hardgecodeerd worden. De onderlinge sortering van deze *positie* kolommen doet er niet toe.
- De cellen van de laatste kolommen geven de namen van de spelers van die ploeg, in categorieën, afgeleid uit de *nationaliteit* van de speler. De categorie waarin een speler met een specifieke nationaliteit moet ondergebracht worden, kan afgeleid worden uit de *landcodes* hash, die in het ***2.pl*** bestand ter beschikking is gesteld. Indien deze hash voor een bepaalde nationaliteit geen categorie oplegt, dan moet de speler ondergebracht worden in een *overige* categorie. De onderlinge sortering van de diverse *categorie* kolommen is niet van belang, behalve dat de *overige* categorie de laatste kolom moet innemen. Binnen elke cel moeten de spelers op verschillende lijnen, gesorteerd op hun achternaam, opgesomd worden. Ter vereenvoudiging mag je veronderstellen dat de achternaam onmiddelllijk na de eerste spatie in de spelersnaam begint. Behalve voor spelers uit de *Belgie* categorie, moet de naam aangevuld worden met (tussen haakjes) de code van hun land van herkomst.

Het script moet een analoge uitvoer bekomen als opgeslagen in het bestand *\temp\tp\demo.html* . Indien je deze uitvoer in een webbrowser bekijkt, dan moet je een resultaat krijgen zoals getoond op de volgende pagina.

Test Perl (3 november 2009)



	Doelman	Verdediger	Middenvelder	Aanvaller	Belgie	Europese Unie	Europa buiten EU	Afrika	overige
Standard	3	9	8	5	Arnor Angeli Igor De Camargo Steven Defour Gregory Dufer Reginal Goreux Anthony Moris Landry Mulemo Kristof Van Hout Axel Witsel	Cedric Collet (FRA) Wilfried Dalmat (FRA) Eliaquim Mangala (FRA) Benjamin Nicaise (FRA) Ricardo Rocha (POR)	Milan Jovanovic (SER)	Gohi Bi Zoro Cyriac (IVO) Mehdi Carcela-Gonzalez (MAR) Dieumerci Mbokani (CON) Mohamed Sarr (SEN) Moussa Traore (IVO)	Felipe (BRA) Ramos (BRA) Sinan Bolat (TUR) Marcos Camozzato (BRA) Alex Moraes (BRA)
Cercle Brugge	3	7	8	6	Frederik Boi Jonas Buyse Olivier Claessens Jo Coppens Hans Cornelis Rubin Dantschotter Sam Devinck Bernt Evens Anthony Portier Tony Sergeant Lukas Van Eenoo Bram Verbist Denis Viane Jelle Vossen	Dominic Foley (IER) Dejan Kelhar (SLO)	Bojan Bozovic (MON) Igor Gjuzelov (MAC) Oleg Iachtchouk (OEK) Serhij Serebrennikov (OEK) Arnar Vidarsson (IJS)	Honour Gombami (ZIM) Vusumuzi Nyoni (ZIM)	Yang Wang (CHI)
Club Brugge	3	8	9	6	Jonathan Blondel Koen Daerden Geert De Vlieger Gertjan De Mets Nabil Dirar Karel Geraerts Carl Hoefkens Yves Lenaerts Roy Meeus Vadis Odjidja-Ofoe Jeroen Simaeys Wesley Sonck Stijn Stijnen Jorn Vermeulen	Mohamed Dahmane (FRA) Ryan Donk (NED)	Dusan Dokic (SER) Ivan Perisic (KRO)	Joseph Akpala (NIG) Dorge Kouemaha (KAM)	Antolin Alcaraz (PAR) Daniel Chavez (PER) Jared Jeffrey (USA) Michael Klukowski (CAN) Cleber Sonda (BRA) Ronald Vargas (VEN)
RSC Anderlecht	5	9	8	5	Sebastien Bruzzese Thomas Chatelle Michael Cordier Tom De Sutter Olivier Deschacht Guillaume Gillet Jonathan Legear Romelo Lukaku Silvio Proto Bakary Sare Davy Schollen Jelle Van Damme	Roland Juhasz (HON) Arnold Kruiswijk (NED) Ondrej Mazuch (CSZ) Jan Polak (CSZ) Marcin Wasilewski (POL) Daniel Zitka (CSZ)	Nemanja Rnic (SER)	Mbark Boussoufa (MAR) Cheikhou Kouyate (SEN)	Reynaldo (BRA) Victor Bernardez (HND) Lucas Biglia (ARG) Nicolas Frutos (ARG) Rubenilson Kanu (BRA) Matias Suarez (ARG)

3) De __DATA__ sectie van het perl script ***3.pl*** bevat, in *LDAP Data Interchange* formaat, een opsomming van de beschikbare klassen in het *Active Directory* domein van onze laboklassen, en hun belangrijkste eigenschappen. De informatie voor de diverse klassen wordt weergegeven in *paragrafen*, blokken lijnen, telkens van elkaar gescheiden door een lege lijn, bijvoorbeeld voor de *group* klasse:

```
                …
                dSCorePropagationData: 16010101000000.0Z
lege lijn →
                dn: CN=Group,CN=Schema,CN=Configuration,DC=iii,DC=hogent,DC=be
                …
                subClassOf: top
                …
                auxiliaryClass: posixGroup
                auxiliaryClass: msSFU30PosixGroup
                objectClassCategory: 1
                lDAPDisplayName: group
                …
                systemMustContain: groupType
                systemAuxiliaryClass: mailRecipient
                systemAuxiliaryClass: securityPrincipal
                …
                dSCorePropagationData: 16010101000000.0Z
lege lijn →
                dn: CN=Cross-Ref,CN=Schema,CN=Configuration,DC=iii,DC=hogent,DC=be
                …
```

Uit elke *paragraaf* voor een specifieke klasse kan (niet noodzakelijk in volgorde) volgende interessante informatie gehaald worden:

- De (enige) lijn beginnend met `lDAPDisplayName:` bepaalt de *naam* van de klasse.
- De (enige) lijn beginnend met `objectClassCategory:` bepaalt het *type* van de klasse, gesymboliseerd door een cijfer (0, 1, 2 of 3).

- De lijn beginnend met `subClassOf:` bepaalt de onmiddellijke *ouderklasse*. Elke klasse heeft één enkele ouderklasse, behalve de *top* klasse: die heeft, als *root* in de klassehiërarchie, geen ouderklasse.
- Lijnen beginnend met `mustContain:` of `systemMustContain:` bepalen de eventuele *verplichte attributen* van de klasse. Elke klasse kan hetzij geen, één of meerdere verplichte attributen bezitten.
- Lijnen beginnend met `auxiliaryClass:` of `systemAuxiliaryClass:` bepalen de eventuele *hulpklassen* van de klasse. Elke klasse kan karakteristieken overerven van hetzij geen enkele, één of meerdere hulpklassen.

Verwerk de relevante gegevens van de __DATA__ sectie in een datastructuur. Na de constructie van de datastructuur mag je de gegevens van de __DATA__ sectie niet opnieuw inlezen ! Construeer hierna stapsgewijs een antwoord op telkens dezelfde vraag (vat een volgende stap pas aan, nadat de vorige succesvol is afgerond):

Stap 1

Genereer een op naam gesorteerde lijst (één lijn per klasse) van alle klassen die verplichte attributen bezitten, met vermelding van deze verplichte attributen (eveneens gesorteerd op naam, met | als scheidingsteken). Je moet bijgevolg 97 lijnen uitvoer bekomen, waaronder:

```
group: groupType
person: cn
top: instanceType|nTSecurityDescriptor|objectCategory|objectClass
```

Stap 2

Vooraleer je een antwoord op dezelfde vraag produceert, pas je eerst de datastructuur zodanig aan dat de verzameling verplichte attributen van elke klasse uitgebreid wordt met de (eventuele) verplichte attributen van de (eventuele) *hulpklassen* waarvan de klasse karakteristieken overerft. Je moet nu 99 lijnen uitvoer bekomen, waaronder (wijzigingen zijn onderlijnd):

```
group: cn|groupType|objectSid|sAMAccountName
person: cn
remoteMailRecipient: cn
top: instanceType|nTSecurityDescriptor|objectCategory|objectClass
user: cn|objectSid|sAMAccountName
```

Stap 3

Vooraleer je een antwoord op opnieuw dezelfde vraag produceert, pas je eerst de datastructuur zodanig aan dat de verzameling verplichte attributen van elke klasse uitgebreid wordt met de (eventuele) verplichte attributen van alle *opeenvolgende ouderklassen* in de klassehiërachie (de *user* klasse bijvoorbeeld erft achtereenvolgens de eigenschappen over van *organizationalPerson*, *person* en *top*). Zorg ervoor dat een attribuut slechts éénmaal optreedt in de lijst van verplichte attributen van een specifieke klasse. Je moet nu 244 lijnen uitvoer bekomen, waaronder (wijzigingen zijn onderlijnd):

```
group: cn|groupType|instanceType|nTSecurityDescriptor|objectCategory|objectClass|objectSid|sAMAccountName
person: cn|instanceType|nTSecurityDescriptor|objectCategory|objectClass
remoteMailRecipient: cn|instanceType|nTSecurityDescriptor|objectCategory|objectClass
top: instanceType|nTSecurityDescriptor|objectCategory|objectClass
user: cn|instanceType|nTSecurityDescriptor|objectCategory|objectClass|objectSid|sAMAccountName
```

Stap 4

Beperk in deze stap de uitvoer tot de (74) klassen die tegelijkertijd aan volgende eigenschappen voldoen:

- de *objectClassCategory* staat ingesteld op 1 (er kunnen dan *objectinstanties* van gecreëerd worden),
- de klasse is *geen ouderklasse* van een andere klasse,
- de klasse heeft *top* niet als ouderklasse.