Test bash (15 december 2010)

Log, voor zover dat nog niet gebeurd zou zijn, in als *root* (wachtwoord *rufemisi)*. Je creëert en test jouw oplossingen uit in de map /root/tb. Voor de antwoorden vul je de bestanden *phi.bash*, *dirusage.bash* en *roos.bash* aan, die reeds **in deze map** aanwezig zijn. Enkel deze bestanden worden bij de quotering geraadpleegd. Alle deelvragen worden praktisch **uitsluitend op het resultaat** gequoteerd: combinaties van `(,),{,},[,],<,>,\,|,/,!,*,@,#,^` en `$`-tekens die misschien wel *enigszins* op een correct antwoord lijken, maar toch niet het gevraagde resultaat opleveren, zullen geen positieve bijdrage tot de quotering leveren. **Lijnen code met syntaxfouten** worden zondermeer verwijderd. Vul **nu reeds** de eerste lijnen van de drie scriptbestanden aan met jouw naam. Vul na het beantwoorden van de vraag de *status* lijnen aan met de aanduiding in hoeverre jouw antwoord aan de gestelde deelvraag voldoet. Verduidelijk de oplossingen met behulp van aanvullende commentaarlijnen. Na het beëindigen van de test (17u00 ten laatste) **log je uit**, maar laat je het toestel verder gewoon aanstaan (**geen shutdown uitvoeren !**).

Behalve alle bash *interne* faciliteiten mag je een beroep doen op volgende *externe* Linux commando's: ***cat***, ***cp***, ***cut***, ***factor***, ***find***, ***grep***, ***head***, ***mktemp***, ***mv***, ***nl***, ***printf***, ***pwd***, ***rm***, ***sort***, ***tail***, ***tee***, ***uniq***, ***wc*** en ***xargs***. Alle andere externe commando's (~~*ls*~~ en ~~*du*~~ bijvoorbeeld) zijn niet toegelaten !

Opgaven

1) De **factor** opdracht laat toe om van alle gehele getallen, die als parameters opgegeven worden, de ontbinding in priemfactoren (priemfactorisatie) te achterhalen. Enkele voorbeelden:

```
[root@… tb]# factor 6188 9405 9702 9973 91665
6188: 2 2 7 13 17
9405: 3 3 5 11 19
9702: 2 3 3 7 7 11
9973: 9973
91665: 3 3 3 5 7 97
```

/4 Ontwikkel een **phi.bash** script dat, ondermeer op basis van de output van de **factor** opdracht (die je moet gebruiken, zonder die zelf te implementeren), voor alle gehele getallen ***n*** die je als parameters aan het script meegeeft, de φ-functie van Euler (ook *totient* functie genoemd) op **twee alternatieve manieren** berekent:

1. Tel het aantal gehele getallen, kleiner dan ***n***, die geen gemeenschappelijke priemfactoren met ***n*** hebben. Voer deze (naïve) berekening zo efficiënt mogelijk uit: ook de performantie zal geëvalueerd worden.
2. Vermenigvuldig ***n*** met factoren $(p_i-1)/p_i$, voor elk van de **verschillende** priemfactoren p_i van ***n***:

$$n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_r}\right),$$

Als uitvoer genereer je, zoals dit hieronder geïllustreerd wordt, voor elke invoerparameter een lijn, die het getal ***n*** zelf en de resultaten van beide φ-berekeningen (die uiteraard gelijk moeten zijn) toont:

```
[root@… tb]# bash phi.bash 6188 9405 9702 9973 91665
6188: 2304 = 2304
9405: 4320 = 4320
9702: 2520 = 2520
9973: 9972 = 9972
91665: 41472 = 41472
```

2) Ontwikkel een **dirusage.bash** script dat, van alle **onmiddellijke submappen** van een map, waarvan je de relatieve of absolute padnaam als enige parameter aan het script meegeeft, volgende informatie toont (één netjes geformatteerde lijn per submap, gesorteerd op de submapnaam): de **padnaam**, het **aantal reguliere bestanden** (willekeurig diep in de hiërarchie onder de submap), en het **totale aantal bytes** die al deze bestanden innemen. Toon eveneens een laatste regel, die dezelfde informatie toont voor de map zelf. Indien je het script aanroept zonder parameter, dat moet je de huidige werkmap als vertrekpunt nemen.

/6 Indien je het script bijvoorbeeld aanroept met */opt* als parameter, dan moet je een analoge output (inclusief formattering !) verkrijgen als hieronder en op de keerzijde geïllustreerd:

```
[root@… tb]# bash dirusage.bash /opt
/opt/apache-ant-1.7.1        1634      40652621
/opt/apache-tomcat-7.0.2      525      10336929
/opt/bash-4.0                1301      29661974
. . .
```

```
. . .
/opt/eclipse                1645     122411884
/opt/emacs-23.1             4085     214879675
/opt/jdk1.6.0_16           22964     583503476
/opt/perldoc-html            938      22745158

/opt                       33094    1161771277
```

3) Het *roos.html* bestand tekent een aantal datapunten in het browservenster. Het *origineel.html* bestand bevat een exacte kopie, die kan gebruikt worden om tijdens het testen de oorspronkelijke toestand te herstellen. Ontwikkel een **roos.bash** script dat in eerste instantie het *roos.html* bestand zodanig aanpast, dat ook de *convex omhullende veelhoek* van de verzameling datapunten getoond wordt. Volg hierbij volgende procedure:

a. Elke lijn die de *canvas* **arc** methodes oproept, tekent één specifiek punt. Bepaal uit dergelijke lijnen de **coördinaten van alle datapunten**.
b. **Sorteer de datapunten** op hun *abscis* (***x***), en bij gelijke abscis, op hun *ordinaat* (***y***).
c. Bepaal een lijst van datapunten, $\mathcal{L}_{\text{boven}}$, die het **bovenste gedeelte van de convex omhullende** voorstelt:
/10
   - Initialiseer $\mathcal{L}_{\text{boven}}$ met de eerste twee datapunten in de in a. vermelde sorteervolgorde.
   - Voeg iteratief de overige datapunten *i* één voor één achteraan de lijst $\mathcal{L}_{\text{boven}}$ toe, en voer hierbij telkens volgende procedure $\mathcal{P}_i$ uit: Zolang de lijst minstens drie datapunten bevat, noem je de laatste drie ervan A, B en C. Indien $\delta = (y_C - y_A)*(x_B - x_A) - (x_C - x_A)*(y_B - y_A)$ groter dan nul is, dan stop je de procedure $\mathcal{P}_i$ voor het datapunt *i* onmiddellijk. Indien echter $\delta \leq 0$, dan verwijder je punt B uit $\mathcal{L}_{\text{boven}}$, en herhaal je de procedure op de nieuwe rij van laatste drie datapunten (voor zover die bestaat).

d. Bepaal een tweede lijst van datapunten, $\mathcal{L}_{\text{onder}}$, die nu het **onderste gedeelte van de convex omhullende** voorstelt, door precies dezelfde werkwijze toe te passen, zoals beschreven in c. voor $\mathcal{L}_{\text{boven}}$, doch nu door de datapunten **in de omgekeerde sorteervolgorde** te behandelen.
e. Vul het *roos.html* **bestand aan met javascript code** om de convex omhullende veelhoek van de datapunten te tekenen. Voeg die code toe net voor de `// Teken Punten` commentaarlijn. Je kan in het *doel.html* bestand, dat het uiteindelijke resultaat voorstelt, verifiëren hoe die moet gebeuren:
   - De lijn `ctx.beginPath();ctx.fillStyle="rgb(255,255,0)";ctx.moveTo(....,....)` stelt de kleur van de veelhoek in op *geel*, en verplaatst de focus naar het eerste punt van $\mathcal{L}_{\text{boven}}$ (het eerste punt in de sorteervolgorde).
   - De daaropvolgende lijnen `ctx.lineTo(....,....)` tekenen de opeenvolgende lijnstukken van de convex omhullende veelhoek, door elk punt van $\mathcal{L}_{\text{boven}}$, en vervolgens elk punt van $\mathcal{L}_{\text{onder}}$ af te lopen, tot uiteindelijk het eerste punt in de sorteervolgorde opnieuw bereikt wordt.
   - De laatste extra lijn, `ctx.fill();ctx.stroke()`, is vereist om de convex omhullende veelhoek effectief te tekenen en in te kleuren.

Nadat je deze procedurestappen a-e volledig en succesvol geïmplementeerd hebt, voeg je een volgende procedurestap toe:

f. Verwijder alle datapunten, die deel uitmaken van de convex omhullende veelhoek (hetzij in de lijst $\mathcal{L}_{\text{boven}}$, hetzij in de lijst $\mathcal{L}_{\text{onder}}$) uit de oorspronkelijke verzameling datapunten.

Je moet dan de procedurestappen b-f **iteratief** toepassen: telkens je een convex omhullende veelhoek berekend en getekend hebt, verwijder je de punten ervan uit de oorsponkelijke verzameling, en herhaal **je de volledige procedure op de resterende datapunten**. Om het mooie visuele effect, dat in *doel.html* gedemonstreerd wordt, te bereiken, moet je ervoor zorgen dat de opeenvolgende convex omhullende veelhoeken met een **steeds intensievere roodschakering** ingekleurd worden. Je kan dit eenvoudig bereiken door de tweede parameter van de `"rgb(255,....,0)"` methode (cfr. stap e.) telkens met 17 eenheden te verminderen: de eerste convex omhullende wordt dan met `"rgb(255,255,0)"` (geel) getekend, de tweede met `"rgb(255,238,0)"`, ..., de vijftiende met `"rgb(255,17,0)"`, de zestiende met `"rgb(255,0,0)"` (felrood).

Stop de iteratie nadat er 16 convex omhullende veelhoeken getekend werden, of indien er uiteindelijk minder dan 3 datapunten overblijven.