Oefeningenexamen Informatica: Januari 2015

1. We wensen te werken met faculteiten. Ter herinnering: een faculteit **n!** is het product van alle natuurlijke getallen van 1 tot en met n. VB: 3! = 1*2*3 = 6

    a. Maak de functie **facIteratief** aan waarbij je een factulteit n interatief oplost en het resultaat wordt teruggegeven.

    b. Maak de functie **facRecursief** waarbij je een faculteit n recursief oplost en het resultaat wordt teruggegeven. Ter herinnering: een recursieve functie roept zichzelf steeds opnieuw op met andere functiewaarden.

    c. Maak de functie **facLimiet** waarbij je een willekeurige limiet kan meegeven. Deze functie hoort de hoogste faculteit te berekenen waarvan het resultaat kleiner is of gelijk aan de doorgegeven limiet. Geef als resultaat zowel n als n! terug waarbij de limiet ligt op 1000000.

2. We willen geheimen communiceren door gebruik te maken van een eenvoudige encryptie techniek: het Caesar cijfer. Bij dit cijferschrift worden alle letters van het alfabet een bepaald aantal karakters geroteerd ten opzichte van elkaar. Bij een rotatie van 1 veranderd een 'A' dus in een 'B'. Na het laatste karakter (vb: 'Z') komt logischerwijze weer het eerste karakter (vb: 'A'). Voorbeeld met een verschuiving van 2: "caesar" -> "ecguct"

    a. Maak een lijst alfabet die de alle hoofdletters, kleine letters, cijfers en enkele speciale karakters bevat met behulp van volgende code:
    from string import letters,digits
    alfabet = list(letters)+list(digits) + [' ','.',',','!','?']

    Om één enkel karakter te vercijferen kan je de index van dit karakter opzoeken in het alfabet, bij deze index de verschuiving optellen en dan kijken welk karakter er die positie staat.

    b. Maak een functie **vercijfer** waaraan je onvercijferde tekst kan meegeven alsook het aantal posities die geroteerd worden. Als resultaat wordt uiteraard de vercijferde tekst teruggegeven.

    c. Maak een nieuwe functie **ontcijfer** waaraan we een vercijferde tekst meegeven alsook het aantal rotaties waarmee is vercijferd. Als resultaat moet de onversleutelde tekst teruggegeven worden. Hergebruik reeds bestaande code indien mogelijk.

    d. Vercijfer de volgende tekst: "Hallo Wereld! Dit is maar een korte test om vercijfering, ontcijfering en IO te testen." Gebruik voor de vercijfering de reeds gemaakte functie met een rotatie van 13. Schrijf vervolgens deze vercijferde tekst weg naar "versleuteld.txt".

    e. Lees nu het bestand "versleuteld.txt" gemaakt in stap d. en ontcijfer deze inhoud door gebruik te maken van de functie uit stap c. Print het resultaat af.

3. We willen functies om binaire getallen om te zetten in decimale getallen en vice versa.

Het getal 5240 in decimale vorm kan je schrijven als:

$$(5240)_{10} = 5 \cdot 1000 + 2 \cdot 100 + 4 \cdot 10 + 0 \cdot 1 = 5 \cdot 10^3 + 2 \cdot 10^2 + 4 \cdot 10^1 + 0 \cdot 10^0$$

Binaire getallen werken net hetzelfde maar tellen in basis 2 in plaats van basis 10. Bijvoorbeeld:

$$(100110)_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = (38)_{10}$$

i. Schrijf een functie **binair_naar_decimaal(b)** die een string b aanneemt die een binair getal voorstelt. Je functie berekent de decimale waarde van dit getal en geeft deze terug als integer. Let op: je functie moet werken ongeacht hoeveel nullen er vooraan in de binaire string staan. Zowel de strings '11' als '011' als '0011' etc. zijn gelijk aan het getal 3.

ii. Schrijf een functie **decimaal_naar_binair(d)** die een integer d aanneemt die een decimaal getal voorstelt. Je functie zet dit getal om in zijn binaire waarde en geeft deze als een string terug.Het algoritme hiervoor wordt hieronder in pseudocode gegeven:

```
zolang getal groter is als 0:
  bit = getal modulo 2
  getal = getal div 2
  schrijf de nieuwe bit links bij de vorige berekende bits
```

div is de deling met gehele getallen (8 div 2 = 4 en 9 div 2 = 4)

4. We willen een Julia fractal tekenen. Hiervoor moeten we rekenen met complexe getallen. We gaan een complex getal z =a + i*b voorstellen door het tupel (a,b).

i. Schrijf functies **comlexe_som(z1,z2)** en **complex_product(z1,z2)** om tupels die een complex getal voorstellen te kunnen optellen en vermenigvuldigen. Schrijf ook een functie **complexe_norm** die de norm van het complex getal berekent en teruggeeft. Deze functies werken wiskundig gezien als volgt:
   a. $(a_1 + i * b_1) + (a_2 + i * b_2) = (a_1 + a_2) + i * (b_1 + b_2)$
   b. $(a_1 + i * b_1) * (a_2 + i * b_2) = (a_1 * a_2 - b_1 * b_2) + i * (a_1 * b_2 + a_2 * b_1)$
   c. $|a + i * b| = \sqrt{a^2 + b^2}$

ii. Schrijf een functie **linspace(begin,eind,aantal_stappen)** die een lijst van float getallen teruggeeft van begin tot en met eind in aantal_stappen stappen. Dus bijvoorbeeld linspace(0,1,5) zou de lijst [0.0,0.25,0.50,0.75,1.0] teruggeven.

iii. Schrijf een functie **Julia_fractaal(N,zc)** die de fractaal berekent en teruggeeft in een NxN matrix:
   a. Gebruik de linspace functie om 2 lijsten x en y aan te maken van -1 tot +1 in in N stappen.
   b. Maak een matrix Julia van NxN met allemaal nullen.
   c. Voor elk paar x[k] en y[l], start vanaf het complex getal x[k] + i*y[l] en itereer de volgende formule:

$$z_{nieuw} = z_{oud} * z_{oud} + z_c$$

   waarbij z_c is een constant complex getal is dat als argument aan de functie wordt meegegeven. Itereer tot de norm van z_{nieuw} groter wordt dan 5 of totdat het aantal iteraties gelijk wordt aan 255. Schrijf het aantal iteraties weg naar Julia[l][k].

Wanneer je met volgende pylab commando's de fractaal visualiseert zou je voor N = 600 en zc = -0.70176 – i*0.3842 de volgende figuur moeten krijgen:

```
from pylab import figure,imshow,show

figure()
imshow(Julia_fractaal(600, (-0.70176,-0.3842)), cmap='hsv')
show()
```