Laborator #4 : exerciții de laborator

Sumar:

I) Exerciții cu fișiere de comenzi (partea întâia -- *script*-uri ce efectuează diverse calcule matematice, iterative sau recursive)

a) Exerciții propuse spre rezolvare

b) Exerciții suplimentare, propuse spre rezolvare pentru acasă

II) Exerciții ce presupun corectarea unor greșeli "strecurate" într-un *script* dat

---

I) Exerciții cu fișiere de comenzi (partea întâia -- *script*-uri ce efectuează diverse calcule matematice, iterative sau recursive):

a) *Exerciții propuse spre rezolvare:*

Intrați pe setul de exerciții propuse spre rezolvare, pe care vi-l va indica profesorul de laborator, în timpul laboratorului, și încercați să le rezolvați singuri:

Setul #1 | Setul #2

setul 1

1. [#1: Iterative math]
   Să se scrie un script care primește o secvență de n numere, plus un număr p citit de la tastatură, și efectuează calculul iterativ al sumei cuburilor acelor numere din secvența primită, ce sunt mai mici sau egale cu p.
   (Indicație: secvența de n numere se va prelua ca argumente din linia de comandă prin care se lansează scriptul, sau se va citi cu comanda read, în caz că nu este dată în linia de comandă.)

2. [#2: Recursive math]
   Să se scrie un script care să efectueze calculul recursiv al celui mai mare divizor comun pentru două numere date a și b. Se va folosi algoritmul lui Euclid cu scăderi, sub forma unei funcții recursive.
   (Indicație: valorile pentru a și b se vor prelua ca argumente din linia de comandă prin care se lansează scriptul, sau se vor citi cu comanda read, în caz că nu sunt date în linia de comandă.)

setul 2

1. [#1: Iterative math]
   Să se scrie un script care primește două numere, k și d, și efectuează calculul iterativ al sumei pătratelor primelor k numere naturale divizibile cu d.
   (Indicație: valorile pentru k și d se vor prelua ca argumente din linia de comandă prin care se lansează scriptul, sau se vor citi cu comanda read, în caz că nu sunt date în linia de comandă.)

2. [#2: Recursive math]
   Să se scrie un script care primește un număr n și calculează recursiv, afișând pe stderr, puterile succesive ale lui 3, de la 3^0 până la 3^n, pe baza relației de recurență: $3^k = 3 * 3^{k-1}$, pentru orice $k>0$.
   (Indicație: valoarea pentru n se va prelua ca argument din linia de comandă prin care se lansează scriptul, sau se va citi cu comanda read, în caz că nu este dată în linia de comandă.)

b) *Exerciții suplimentare, propuse spre rezolvare pentru acasă:*

Iată alte câteva exerciții cu fișiere de comenzi, pe care să încercați să le rezolvați singuri în timpul liber, pentru a vă auto-evalua cunoștințele dobândite în urma acestui laborator:

1. [Iterative math #6]
   Să se scrie un script care să calculeze valoarea de rang n din șirul lui Fibonacci.
   (Indicație: valoarea pentru n va fi preluată ca argument la linia de comandă prin care se lansează scriptul, sau se va citi prin comanda read, în caz că nu este dată în linia de comandă.)

2. [Iterative math #7]
   Să se scrie un script care să calculeze A(n,k) (i.e., aranjamente de n luate câte k).
   (Indicație: valorile n și k se vor prelua ca argumente din linia de comandă prin care se lansează scriptul, sau se vor citi prin comanda read, în caz că nu sunt date în linia de comandă.)

3. [Iterative math #8]
   Să se scrie un script care va fi lansat cu 1+N*k+p argumente, unde k și p sunt variabile și necunoscute apriori. Scriptul preia primul argument într-o variabilă N și calculează suma următoarelor argumente luate câte N (deci suma primelor N argumente, apoi a următoarelor N numere ș.a.m.d., iar dacă la sfârșit rămâne un număr mai mic decât N de argumente, va calcula doar suma acestora), afișând valorile sumelor pe ecran și scriindu-le în același timp într-un fișier output.txt.
   Dacă este lansat fără nici un parametru, atunci va cere introducerea unei valori numerice N și apoi a altor N valori numerice (presupunem numere întregi), afișând ca rezultat doar această primă sumă.
   (Indicație: man expr și help shift.)

4. [Iterative math #9]
   Să se scrie un script care să efectueze calculul iterativ al factorialului (i.e., `n! = 1 * 2 * ... * n, pentru n > 0`).
   (Indicație: valoarea pentru n se va prelua ca argument din linia de comandă prin care se lansează scriptul, sau se va citi cu comanda read, în caz că nu este dată în linia de comandă.)

5. [Recursive math #3]
   Să se scrie un script care primește o valoare n ca parametru de intrare (verificați că n > 0) și calculează în manieră recursivă suma cifrelor lui, ce sunt numere prime, iar la finalul calculului va afișa valoarea sumei.
   Exemplul #1: n=1235 --> rezultat: 2+3+5=10;
   Exemplul #2: n=6274 --> rezultat: 2+7=9.
   (Indicație: valoarea pentru n va fi preluată ca argument la linia de comandă prin care se lansează scriptul, sau se va citi prin comanda read, în caz că nu este dată în linia de comandă.)

6. [Recursive math #4]
   Să se scrie un script care primește o valoare n ca parametru de intrare (verificați că n > 0) și calculează în manieră recursivă inversul numărului obținut prin complementarierea tuturor cifrelor față de 9, iar la finalul calculului va afișa noul număr astfel calculat.
   Exemplul #1: n=13950 --> rezultat: 94068, deoarece 9-0=9, 9-5=4, 9-9=0, 9-3=6, 9-1=8;
   Exemplul #2: n=349 --> rezultat: 56, deoarece 9-9=0, 9-4=5, 9-3=6.
   (Indicație: valoarea pentru n va fi preluată ca argument la linia de comandă prin care se lansează scriptul, sau se va citi prin comanda read, în caz că nu este dată în linia de comandă.)

7. [Recursive math #5]
   Să se scrie un script care primește o valoare n ca parametru de intrare (verificați că n > 0) și calculează în manieră recursivă termenul x_n al șirului $\{x_m\}_{m>=0}$, definit astfel:
   $x_0 = 1,\ x_1 = 2$, respectiv $x_m = x_{m-1} + 4*x_{m-2} + m$, pentru orice $m>=2$.
   La final se va afișa valoarea termenului cerut x_n. Se vor trata toate cazurile de excepție ce pot apare.
   (Indicație: valoarea pentru n va fi preluată ca argument la linia de comandă prin care se lansează scriptul, sau se va citi prin comanda read, în caz că nu este dată în linia de comandă.)

8. [Recursive math #6]
   Să se scrie un script care primește o valoare n ca parametru de intrare (verificați că n > 0) și calculează în manieră recursivă termenul y_n al șirului $\{y_m\}_{m>=0}$, definit astfel:
   $y_0 = 1,\ y_1 = 1$, respectiv $y_m = (m+1)*y_{m-1} + (m+2)*y_{m-2}$, pentru orice $m>=2$.

La final se va afișa valoarea termenului cerut y_n. Se vor trata toate cazurile de excepție ce pot apare.

(Indicație: valoarea pentru n va fi preluată ca argument la linia de comandă prin care se lansează scriptul, sau se va citi prin comanda read, în caz că nu este dată în linia de comandă.)

II) Exerciții ce presupun corectarea unor greșeli "strecurate" într-un script dat:

1. [Exemplul #1 cu erori sintactice și semantice]

Se dă scriptul următor, care se va apela cu o serie de parametri în linia de comandă:

```
#!/bin/noshell
A=1; #B=0;
for a in $(ls /usr/share)
do
  [ -d $a ] && let B++
done
echo #B, $A
```

i) Explicați ce se va afișa pe ecran în urma execuției scriptului.
ii) Corectați eventualele erori existente astfel încât, în urma execuției scriptului corectat, pe ecran să apară afișat numărul subdirectoarelor aflate în directorul `/usr/share` , urmat de numărul parametrilor cu care a fost apelat scriptul.

Show / Hide some suggestions for solving the problem

i) Executați scriptul, identificați eventualele mesaje de eroare ce apar pe ecran și încercați să identificați cauzele care le produc!

ii) Corectați, pe rând, fiecare greșeală identificată în maniera descrisă la pct.i), repetând procedeul până când nu mai apare niciun mesaj de eroare la execuția scriptului.
Apoi încercați să identificați și eventualele erori semantice, i.e. *bug*-uri în cod, adică greșeli strecurate în script care nu cauzează mesaje de eroare la execuția lui, dar datorită cărora scriptul nu face exact ceea ce este specificat în enunț că ar trebui să facă!

2. [Exemplul #2 cu erori sintactice și semantice]

Se dă scriptul următor, care se va apela cu un parametru în linia de comandă:

```
#!/bin/nologin
if[ $# -lt 1] then
    read -p "Dati n: " n
else
    n=$1

prod=0
for k in ${seq 1 n}
do
    prod*=k
done

echo "Fact($n)=prod"
```

i) Explicați ce se va afișa pe ecran în urma execuției scriptului.
ii) Corectați eventualele erori existente astfel încât, în urma execuției scriptului corectat, pe ecran să apară afișată valoarea factorialului pentru numărul primit ca parametru.

Show / Hide some suggestions for solving the problem

i) Executați scriptul, identificați eventualele mesaje de eroare ce apar pe ecran și încercați să identificați cauzele care le produc!

ii) Corectați, pe rând, fiecare greșeală identificată în maniera descrisă la pct.i), repetând procedeul până când nu mai apare niciun mesaj de eroare la execuția scriptului.
Apoi încercați să identificați și eventualele erori semantice, i.e. *bug*-uri în cod, adică greșeli strecurate în script care nu cauzează mesaje de eroare la execuția lui, dar datorită cărora scriptul nu face exact ceea ce este specificat în enunț că ar trebui să facă!