Test practic la SO – varianta nr. 1

Realizați o aplicație formată din următoarele patru componente cooperante, definite în continuare, care vor fi dispuse în sistemul de fișiere conform ierarhiei de mai jos:

```
.
├── main.sh
├── validators
│   └── validatorASCII.c
│   └── default.sh
└── xprocesator
    └── procesator.sh
```

1. Să se scrie un program C, numit "validatorASCII.c", conform specificației următoare:
Programul va primi un argument în linia de comandă; în caz contrar se va termina cu codul de eroare 2. Programul va considera acel argument un nume de fișier și va verifica dacă are drept de read asupra sa; în caz contrar se va termina cu cod de eroare 3. Altfel, programul va afișa pe ieșirea stdout, folosind o funcție din biblioteca stdio, un mesaj formatat astfel: numele fișierului primit ca argument, separatorul ':' , numărul de caractere litere mari din fișier, separatorul ':' , numărul de caractere litere mici din fișier, și terminat cu caracterul '\n'. Apoi programul își va încheia execuția cu codul de terminare 0.

2. Să se scrie un script bash, numit "main.sh", conform specificației următoare:
Scriptul va primi un argument în linia de comandă și va face următoarele verificări, în ordinea următoare:
i) va verifica dacă în directorul în care se află "main.sh", există un subdirector numit "xprocesator" ce conține un script numit "procesator.sh" și că are permisiunea de execuție a acestuia, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 1;
ii) va verifica dacă în directorul în care se află "main.sh", există un subdirector numit "validators" ce conține un program numit "validatorASCII.c" și că are permisiunea de citire a acestuia, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 2;
iii) va verifica dacă argumentul primit reprezintă un director existent și asupra căruia are permisiune de citire, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 3.
Apoi scriptul "main.sh" va invoca scriptul "procesator.sh", transmițându-i în linia de comandă, argumentul pe care l-a primit el în linia de comandă, și la final va afișa codul de terminare a execuției acestuia.

3. Să se scrie un al doilea script bash, numit "procesator.sh", conform specificației următoare:
Scriptul va primi un argument în linia de comandă; în caz contrar se va termina cu codul de exit 2.
Mai întâi scriptul va verifica dacă în subdirectorul "validators" există un script numit "default.sh" și că are permisiunea de execuție a acestuia, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 1. De asemenea, va valida că argumentul primit este un director asupra căruia are permisiuni de citire și de traversare (*executable*). În caz negativ, se va termina cu codul 3.

Apoi scriptul "procesator.sh" va parcurge, folosind o structură repetitivă, toate intrările directe din acel director și, pentru fiecare intrare care este subdirector, scriptul "procesator.sh" se va apela recursiv pe sine însuși, cu acea intrare ca parametru în linia de comandă. În schimb, dacă acea intrare este un fișier de tip normal, va apela scriptul “default.sh” cu acea intrare ca argument și redirectând outputul standard și cel de eroare, în modul *append*, într-un fișier denumit ”results.txt”. (Atenție: așadar, aici trebuie să implementați o recursie explicită!)

4. Să se scrie un al treilea script bash, numit "default.sh", conform specificației următoare:
Scriptul va primi un argument în linia de comandă; în caz contrar, va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul de exit 2.
Mai întâi scriptul va verifica dacă în subdirectorul "validators" (același în care se află și el) se mai află și un fișier numit "validatorASCII" (executabilul obținut prin compilare din programul "validatorASCII.c"), iar în caz negativ va apela compilatorul gcc pentru a-l produce și, doar dacă vor fi erori la compilare, va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul de exit 3.
Apoi scriptul "default.sh" va apela, printr-un pipeline, executabilul ”validatorASCII” cu input calea către fișierul respectiv, iar în outputul produs de executabil va înlocui aparițiile caracterului ”:” cu caracterul “>”.

Barem de corectare

Observații:
- programul C și cele trei scripturi trebuie plasate într-o ierarhie de directoare conform celor descrise în enunțul problemei (și apelate în mod corespunzător poziției lor în ierarhia respectivă).
- conform specificației date, programul C poate folosi biblioteca standard de C doar pentru afișarea rezultatelor; interacțiunea cu fișierul (aflarea tipului, lungimii, etc.) se va face folosind doar API-ul POSIX.
- pentru implementarea parcurgerii recursive se va respecta specificația dată pentru scriptul "procesator.sh".
- **dacă nu respectați toate condițiile precedente** (de exemplu, dacă plasați vreunul dintre cele trei fișiere apelate pornind de la “main.sh” într-un alt director, sau dacă folosiți comenzi precum find sau ls -R pentru parcurgerea recursivă în cadrul "procesator.sh"), atunci vom evalua corectitudinea rezolvării, dar **punctajul total acordat va fi înjumătățit**.
- fiecare punctaj din barem se acordă integral, sau deloc.

1. Baremul pentru programul "validatorASCII.c":
a) 0.5p + 0.5p – testarea numărului de argumente și, respectiv, a dreptului de citire asupra fișierului.
b) 1p + 1p – calculul numărului de majuscule și de minuscule din conținutul fișierului și, respectiv, afișarea rezultatului pe ieșirea stdout, conform specificației date.
c) 1p – tratarea tuturor erorilor posibile, conform specificației date.
d) 1p – programul se compilează fără erori și fără warning-uri.
Total: 5p

2. Baremul pentru scriptul "main.sh":
a) 1p – implementarea corectă, conform specificației date, a verificării descrise la punctul i) din specificație.
b) 1p – implementarea corectă, conform specificației date, a verificării descrise la punctul ii) din specificație.
c) 1p - implementarea corectă, conform specificației date, a verificării descrise la punctul iii) din specificație.
d) 1p + 0.5p – invocarea corectă a scriptului "procesator.sh" și, respectiv, afișarea codului de exit, conform specificației date.
Total: 4.5p

3. Baremul pentru scriptul "procesator.sh":
a) 0.5p + 1p – implementarea corectă, conform specificației date, a verificărilor referitoare la argumentul primit.
b) 1p – implementarea corectă, conform specificației date, a verificării referitoare la scriptul "default.sh".
c) 1p – parcurgerea iterativă a intrărilor directe din directorul primit ca argument.
d) 1p – apelarea recursivă a scriptului pentru fiecare intrare care este director.
e) 1p + 1p – apelul corect al scriptului “default.sh” pentru intrările ce sunt fișiere normale, inclusiv cu operațiile de redirectare scrise în mod corect.
Total: 6.5p

4. Baremul pentru scriptul "default.sh":
a) 0.5p – implementarea corectă, conform specificației date, a verificării referitoare la argumentul primit.
b) 0.5p + 1p – implementarea corectă, conform specificației date, a verificării referitoare la programul executabil "validatorASCII" și apelul compilatorului, dacă este cazul, conform specificației date.
c) 1p – invocarea corectă a programului executabil "validatorASCII", conform specificației date.
d) 1p – apelul comenzii potrivite, în cadrul pipeline-ului, pentru înlocuirea caracterului ':' cu caracterul ‘>’ , conform specificației date.
Total: 4p