Part II – test programare Seria: 1

a) Specificație (cerințe):

i) Scrieți un program C care să poată fi rulat în mod **cooperativ**. Acesta va primi ca și argumente de intrare calea către un fișier BINAR (posibil inexistent) și calea către un director.
Alternativ, programul va suporta invocarea cu argumentul "-o", caz în care va afișa textual conținutul fișierului binar primit ca parametru.

Structura fișierului binar este următoarea: o secvență de înregistrări consecutive (inițial este secvența vidă), unde fiecare înregistrare este formată din două valori, prima de tipul *unsigned short* și a doua de tipul *unsigned int* (stocate în formatul de reprezentare internă/binară a numerelor întregi), prima valoare reprezintă o mască de biți și a doua un contor, a cărui semnificație este descrisă mai jos.

Programul va parcurge RECURSIV directorul dat (parcurgând în mod EXPLICIT subarborele de fișiere determinat de acesta). Pentru fiecare fișier (de orice tip, inclusiv director) găsit prin această parcurgere, programul va extrage masca de biți ce descrie permisiunile asociate acelui fișier și va incrementa în fișierul binar contorul (reprezentând numărul de apariții) corespunzător acelei măști. Dacă fișierul binar specificat nu conține o înregistrare corespunzătoare acelei măști de biți, atunci programul va adăuga o nouă înregistrare la finalul acestuia, ce va conține acea mască de biți și contorul, inițializat în mod adecvat, asociat măștii.

Accesul la fișierul binar va fi realizat în manieră **sincronizată**. Cooperarea mai multor instanțe ale programului rulate în paralel trebuie să fie eficientă, ceea ce implică punerea de blocaje în mod eficient, adică pe o durată minimală și pe porțiunea minimală din fișierul binar.

Un posibil exemplu de rulare:

```
./count_rwx perm.bin dir1 & ./count_rwx perm.bin dir2 & ./count_rwx perm.bin dir3 &
```

Se vor folosi doar apeluri POSIX pentru toate operațiile cu fișiere, cu excepția doar a operației de afișare pe ecran a rezultatului final obținut în fișierul binar, pentru care puteți folosi apeluri de funcții din biblioteca libc/stdio.
Se vor trata în mod corespunzător toate erorile survenite la apelurile de funcții POSIX.

ii) Scrieți un script bash de automatizare, care va face următoarele:
a) Mai întâi scriptul va citi un număr P(=numărul de instanțe), apoi va inițializa fișierul binar (resetare la zero), și eventual va compila sursa C (dacă nu găsește executabilul).
b) Apoi va lansa în execuție un job SPMD, prin care să se execute în paralel P instanțe ale programului de la pct.i), având ca parametri același fișier binar și câte un director diferit pentru fiecare dintre cele P instanțe.
c) Iar după terminarea execuției celor P instanțe scriptul va apela programul de la pct.i) cu parametrul "-o" și numele fișierului binar, pentru a afișa rezultatele obținute în fișierul binar.

b) Baremul pentru program:
– implementarea corectă a parcurgerii recursive explicite: 2p
– implementarea corectă a cerințelor de procesare, creare și actualizare a fișierului binar: 5p
– implementarea mecanismelor de sincronizare pentru corectitudinea cooperării jobului SPMD: 3p
– implementarea în manieră eficientă a sincronizării: 2p
– implementarea corectă a procesării opțiunii "-o": 1p
– tratarea tuturor erorilor la apelurile de funcții POSIX: 1p
Baremul pentru script:
– implementarea corectă a cerinței a) formulate pentru script: 1.5p
– implementarea corectă a cerinței b) formulate pentru script: 1p
– implementarea corectă a cerinței c) formulate pentru script: 1p