Part II – test programare Seria: 2

a) Specificație (cerințe):

Scrieți o aplicație formată din trei programe C, denumite supervisor.c , worker1.c și worker2+3.c , care să poată fi rulate în mod **cooperativ**, în modul următor (unde *N* și *M* sunt numere întregi pozitive oarecare) :

```
{ sleep N ; ./supervisor ; } &  { sleep M ; ./worker2+3 ; } &
```

Funcționalitățile celor trei programe sunt specificate în cele de mai jos. Se vor folosi doar apeluri POSIX pentru toate operațiile de citire/scriere în/din canalele de comunicație specificate mai jos. Se vor trata în mod corespunzător toate erorile survenite la apelurile de funcții POSIX.

1. Programul supervisor.c va implementa funcționalitatea descrisă în specificația următoare:

Programul va primi un argument în linia de comandă, ce reprezintă calea (absolută sau relativă) către un fișier existent pe disc, numit "input_data.txt". Acest fișier conține o secvență de numere întregi, în format textual.
i) În funcția principală a programului, acesta va testa existența unui argument primit în linia de comandă și va face inițializările necesare pentru a putea trimite informații procesului **worker1** printr-un canal anonim (prin comunicații unu-la-unu) și, respectiv, pentru a putea primi rezultate de la procesul **worker3** printr-un obiect de memorie partajată – o mapare nepersistentă cu nume, creată cu funcția *shm_open*. De asemenea, tot în funcția principală a programului, programul va crea un proces fiu, iar în fiul creat va starta, printr-un apel exec adecvat, programul **worker1**.

ii) Într-o funcție separată, apelată din funcția main, programul va citi pe rând, unul câte unul, fiecare număr din acel fișier "input_data.txt" și-l va converti la reprezentarea în format binar a numerelor întregi, iar rezultatul conversiei îl va transmite către procesul **worker1** prin acel canal anonim, folosind apeluri POSIX (așadar, folosind reprezentarea binară a numerelor întregi, le veți transmite prin mesaje de lungime constantă).

iii) Într-o altă funcție separată, apelată din funcția main, programul va citi, folosind apeluri POSIX, numerele transmise lui de către procesul **worker3** (descris mai jos) prin intermediul acelei mapări nepersistente cu nume și le va aduna. Rezultatul acestei adunări va fi convertit la reprezentarea textuală în baza 10 a numerele întregi și apoi se va calcula suma cifrelor acestei reprezentări, afișând-o la final pe ecran.

2. Programul worker1.c va implementa funcționalitatea descrisă în specificația următoare:

i) În funcția main, va face inițializările necesare pentru a putea primi informații de la procesul **supervisor** printr-un canal anonim (prin comunicații unu-la-unu). De asemenea, tot în funcția main, va face inițializările necesare pentru a putea trimite informații către procesul **worker2**, printr-un canal fifo (prin comunicații unu-la-unu).

ii) Într-o funcție separată, apelată din funcția main, programul va citi din acel canal anonim, folosind apeluri POSIX, fiecare număr transmis lui de către procesul **supervisor** (prin mesaje de lungime constantă, folosind reprezentarea binară a numerelor întregi), și va verifica dacă acesta este număr nedivizibil cu 3, caz în care îl înmulțește cu 3, sau este număr divizibil cu 3, caz în care îl păstrează neschimbat; în ambele cazuri, valoarea astfel obținută va fi transmisă mai departe către procesul **worker2,** prin acel canal canal fifo (se va păstra reprezentarea în binar pentru numărul respectiv). De asemenea, programul va contoriza într-o variabilă *contor_inmultiri* câte operații de înmulțire cu 3 a efectuat pe parcursul procesării tuturor numerelor primite de la procesul **supervisor** și va transmite la final această valoare către procesul **worker2,** prin acel canal canal fifo.

3. Programul worker2+3.c va implementa funcționalitatea descrisă în specificația următoare:

i) În funcția main, va crea un obiect de memorie partajată – o mapare nepersistentă anonimă. De asemenea, tot în funcția principală a programului, programul va crea un proces fiu, iar în fiul creat va executa funcția **worker3**. Iar tatăl va executa apoi funcția **worker2**.

ii) În procesul tată / funcția **worker2** se vor face inițializările necesare pentru a putea primi date de la procesul **worker1**, prin acel canal fifo (prin comunicații unu-la-unu). De asemenea, se va pregăti pentru a putea trimite date către procesul fiu / funcția **worker3**, prin acea mapare nepersistentă anonimă. Apoi se vor citi, rând pe rând, fiecare număr transmis de către procesul **worker1** prin acel canal fifo, folosind apeluri POSIX (așadar, folosind reprezentarea binară a numerelor întregi, veți transmite mesaje de lungime constantă), și va calcula pătratul acelui număr. Rezultatul acestui calcul de ridicare la pătrat îl va transmite către procesul fiu / funcția **worker3**, prin intermediul acelei mapări nepersistente anonime (se va păstra reprezentarea în binar pentru numere întregi).

La final va citi valoarea *contor_inmultiri* transmisă lui de către procesul **worker1** și o va transmite mai departe către procesul fiu / funcția **worker3**.

iii) În procesul fiu / funcția **worker3**, se vor face inițializările necesare pentru a putea trimite rezultate către procesul **supervisor** prin acea mapare nepersistentă cu nume descrisă în specificația programului **supervisor**. De asemenea, se va pregăti pentru a putea primi date de la procesul tată / funcția **worker2**, prin acea mapare nepersistentă anonimă. Apoi va citi, rând pe rând, fiecare număr transmis lui transmis de către procesul tată / funcția **worker2,** prin intermediul acelei mapări nepersistente anonime**,** și se va procesa informația citită astfel: va calcula suma tuturor numerelor primite, mai puțin ultimul (i.e., acea valoare *contor_inmultiri*). Apoi va transmite această sumă, precum și valoarea *contor_inmultiri*, către procesul **supervisor,** prin intermediul acelei mapări nepersistente cu nume.

Recomandare:

Mai întâi, desenați de mână pe o foaie de hârtie o schemă cu **arhitectura aplicației**: ierarhia de procese, mijloacele de comunicație între procese și sensul de transmitere a informației prin aceste mijloace de comunicație (precum ați văzut în exemplele de diagrame realizate de mână, atașate la unele exerciții rezolvate în suporturile online de laborator).
Schema vă va ajuta să vă clarificați mai bine cerințele specificate în enunțul problemei și, de asemenea, vă va ajuta la partea de implementare!
De asemenea, citiți mai întâi și baremul asociat acestui subiect.

b) Baremul pentru cele trei programe:

1. Baremul pentru programul supervisor.c (total: **7.5 puncte**)

– implementarea corectă (și fără *bug*-uri de concurență) a configurării și utilizării canalului anonim pentru comunicație cu procesul worker1: **2p**
– implementarea corectă (și fără *bug*-uri de tipul *race-condition* sau *segmentation fault*) a recepționării informațiilor de la procesul worker3 prin acea *mapare nepersistentă cu nume*, în funcția descrisă la punctul iii) din specificația dată: **2.5p**
– implementarea corectă a operațiilor fork și, respectiv, exec descrise la punctul i) din specificația dată: **1p**
– implementarea corectă a cerințelor de procesare a datelor de intrare și, respectiv, a rezultatelor primite: **0.5p + 1p**
– tratarea tuturor erorilor la apelurile de funcții POSIX: **0.5p**

2. Baremul pentru programul worker1.c (total: **5.5 puncte**)

– implementarea corectă (și fără *bug*-uri de concurență) a configurării și utilizării canalului anonim pentru comunicație cu procesul supervisor: **2p**
– implementarea corectă a cerinței de procesare a datelor primite: **1p**
– implementarea corectă (și fără *bug*-uri de concurență) a configurării și utilizării canalului fifo pentru comunicație cu procesul worker2: **2p**
– tratarea tuturor erorilor la apelurile de funcții POSIX: **0.5p**

3. Baremul pentru programul worker2+3.c (total: **12 puncte**)

– implementarea corectă a creării unei mapări anonime și, respectiv, a unui proces fiu (inclusiv apelarea celor două funcții worker2 și worker3): **0.5p + 0.5p**
– tratarea tuturor erorilor la apelurile de funcții POSIX: **0.5p**
3.i) în procesul fiu / funcția worker2:
– implementarea corectă (și fără *bug*-uri de concurență) a configurării și utilizării canalului fifo pentru comunicație cu procesul worker1: **2p**
– implementarea corectă a cerinței de procesare a datelor primite: **1p**
– implementarea corectă (și fără *bug*-uri de tipul *race-condition* sau *segmentation fault*) a transmiterii informațiilor procesate către procesul worker3 prin acea *mapare nepersistentă anonimă*, în funcția descrisă la punctul ii) din specificația dată: **2p**
3.ii) în procesul tată / funcția worker3:
– implementarea corectă (și fără *bug*-uri de tipul *race-condition* sau *segmentation fault*) a recepționării informațiilor de la procesul worker2 prin acea *mapare nepersistentă anonimă*, în funcția descrisă la punctul iii) din specificația dată: **2p**
– implementarea corectă a cerinței de procesare a datelor primite: **1p**
– implementarea corectă (și fără *bug*-uri de tipul *race-condition* sau *segmentation fault*) a transmiterii informațiilor procesate către procesul supervisor prin acea *mapare nepersistentă cu nume*, în funcția descrisă la punctul iii) din specificația dată: **2.5p**