**Part II – test programare Seria: 1**

**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, pe fiecare linie, câte o tripletă de numere reale în format textual, separate prin spații: "a b c".
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 canal fifo (prin comunicații unu-la-unu). 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, una câte una, fiecare linie din acel fișier “input_data.txt” și va converti fiecare tripletă citită la reprezentarea în format binar a numerelor reale, iar rezultatul conversiei (i.e., cele trei numere reale a,b,c în format binar) îl va transmite către procesul **worker1** prin acel canal anonim, folosind apeluri POSIX (așadar, folosind reprezentarea binară a numerelor reale, veți transmite mesaje de lungime constantă).

iii) Într-o altă funcție separată, apelată din funcția main, programul va citi, folosind apeluri POSIX, fiecare cvintet de 5 numere reale a,b,c,P,S transmis lui de către procesul **worker3** (descris mai jos) prin acel canal canal fifo și, dacă P!=0, afișează pe ecran mesajul: tripleta a,b,c reprezintă lungimile laturilor unui triunghi ce are perimetrul P și aria S, iar dacă P=0, afișează pe ecran mesajul: tripleta a,b,c nu poate reprezenta lungimile laturilor unui triunghi. De asemenea, va contoriza câte triplete a,b,c pot reprezenta lungimile laturilor unui triunghi și câte nu pot, afișând la final pe ecran un mesaj adecvat cu cele două totaluri.

**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 obiect de memorie partajată – o mapare nepersistentă cu nume, creată cu funcția *shm_open*.

ii) Într-o funcție separată, apelată din funcția main, programul va citi din acel canal anonim, folosind apeluri POSIX, fiecare tripletă a,b,c transmisă lui de către procesul **supervisor** (prin mesaje de lungime constantă, folosind reprezentarea binară a numerelor reale), și va verifica dacă cele trei numere reale a,b,c ar putea fi lungimile laturilor unui triunghi (i.e., condițiile: a,b,c > 0 și suma oricăror două să fie mai mică decât al treilea). Rezultatul acestei verificări va fi transmis mai departe către procesul **worker2,** prin intermediul acelei mapări nepersistente cu nume, sub forma a 4 numere a,b,c,r , unde a,b,c sunt cele trei numere reale, iar r este o valoare întreagă egală cu 0, dacă nu formează triunghi, respectiv 1, dacă formează triunghi (se va păstra reprezentarea în binar pentru toate cele 4 numere).

**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 **worker2**. Iar tatăl va executa apoi funcția **worker3**.

ii) În procesul fiu / funcția **worker2** se vor face inițializările necesare pentru a putea primi date de la procesul **worker1**, prin acea mapare nepersistentă cu nume descrisă în specificația programului **worker1**. De asemenea,

se va pregăti pentru a putea trimite date către procesul tată / funcția **worker3**, prin acea mapare nepersistentă anonimă. Apoi se vor citi, rând pe rând, fiecare cvartet de forma a,b,c,r transmis de către procesul **worker1** prin intermediul acelei mapări nepersistente cu nume și, dacă r=1, atunci va calcula valoarea p = semi-perimetrul triunghiului cu laturile a,b,c, iar dacă r=0, atunci va seta p = 0. Rezultatul acestui calcul îl va transmite, sub forma a 4 numere a,b,c,p către procesul tată / funcția **worker3**, prin intermediul acelei mapări nepersistente anonime (se va păstra reprezentarea în binar pentru toate cele 4 numere).

iii) În procesul tată / funcția **worker3**, se vor face inițializările necesare pentru a putea trimite rezultate către procesul **supervisor** prin acel canal fifo (prin comunicații unu-la-unu). De asemenea, se va pregăti pentru a putea primi date de la procesul fiu / funcția **worker2**, prin acea mapare nepersistentă anonimă. Apoi se vor citi, rând pe rând, fiecare cvartet de forma a,b,c,p transmis de către procesul fiu / funcția **worker2,** prin intermediul acelei mapări nepersistente anonime**,** și se va procesa informația citită astfel: dacă p!=0, atunci se calculează perimetrul P = 2*p , precum și aria triunghiului, după formula lui Heron: S = sqrt(p*(p-a)*(p-b)*(p-c)) , iar dacă p=0, atunci se va asigna P=0 și S=0. Apoi se va transmite rezultatul acestei procesări, sub forma a 5 numere reale a,b,c,P,S , către procesul **supervisor,** prin acel canal fifo, folosind apeluri POSIX (așadar, folosind reprezentarea binară a numerelor reale, veți transmite mesaje de lungime constantă).

**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 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 concurență) a configurării și utilizării canalului fifo pentru comunicație cu procesul worker3: **2p**
– 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: **6 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 tipul *race-condition* sau *segmentation fault*) a transmiterii informațiilor procesate către procesul worker2 prin acea *mapare nepersistentă cu nume*, în funcția descrisă la punctul ii) din specificația dată: **2.5p**
– 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 tipul *race-condition* sau *segmentation fault*) a recepționării informațiilor de la procesul worker1 prin acea *mapare nepersistentă cu nume*, în funcția descrisă la punctul ii) din specificația dată: **2.5p**
– 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 concurență) a configurării și utilizării canalului fifo pentru comunicație cu procesul supervisor: **2p**