

Nume și grupă: _____

1. (a) (5p) Ce este un proces *zombie*?
(b) (5p) Scrieți o secvență scurtă de cod și arătați când un proces devine *zombie*.
2. Fie următoarea secvență

```
for (i=0; i<3; i++) {  
    fork();  
    pthread_create();  
    fork();  
}
```


(a) (5p) Câte procese și fire de execuție sunt create? La numărare, ce presupunere ați făcut legată de `fork()`?
(b) (5p) Desenați arborescența proceselor și firelor de execuție create. Etichetați cu P procesele și cu T firele de execuție.
3. Considerați problema filosofilor și soluția propusă mai jos pentru $n \in \mathbb{N}$ filosofi așezați la masă.

```
do {  
    wait(chopstick[i]);  
    wait(chopstick[(i+1)%n]);  
    /* ... */  
    signal(chopstick[i]);  
    signal(chopstick[(i+1)%n]);  
} while (true);
```

Această soluție permite apariția fenomenului de *deadlock*.

- (a) (5p) Modificați soluția ridicând asimetric bețișoarele: filosofi impari ridică întâi bețișorul din stânga, cei pari pe cel din dreapta. Arătați că nu mai apare fenomenul.
 - (b) (10p) Arătați dacă noua soluție satisface cele trei proprietăți: exclusivitate mutuală, progres și timp finit de așteptare.
4. Fie o matrice $A \in \mathbb{N}^{10 \times 10}$ ținută contiguu în memorie pe linii și fie un sistem în care avem 3 *frame*-uri disponibile. În acest sistem într-o pagină încap 10 întregi, iar programele P1 și P2 de mai de jos încap în totalitate într-o pagină.

P1:

```
for (i = 0; i < 10; i++)  
    for (j = 0; j < 10; j++)  
        A[i][j] = 0;
```

P2:

```
for (j = 0; j < 5; j++)  
    for (i = 0; i < 5; i++)  
        A[i][j] = 0;
```

- (a) (5p) Cum arată programul și datele repartizate pe pagini?
- (b) (5p) Folosind algoritmul LRU, care este programul eficient? De ce?
- (c) (5p) Cum arată diagramele Gantt pentru P1 și P2?