1. Două procese (P1, P2) folosesc o zonă de memorie partajată, rulează într-un sistem preemptiv și execută secvența de cod de mai jos. Știind că valoarea inițială indicată de pointerul **counter** este 0 și că acesta indică în zona de memorie partajată, care este valoarea indicată de pointerul **counter** la finalul execuției celor două procese?

P1	P2
`if (*counter == 0)` `  (*counter)++;`	`if (*counter == 0)` `  (*counter)++;`

2. Pe un sistem dual processor cu 128MB de RAM și swap de 256MB rulează un sistem Linux. Câte procese se pot găsi la un moment dat în starea **RUNNING**, **READY** respectiv **WAITING**?

3. Un sistem folosește TLB în care fiecare intrare conține trei câmpuri (pid, frame, pagină virtuală). Care este avantajul acestei implementări față de o implementare care conține doar două câmpuri (frame, pagină virtuală)?

4. Descrieți în pseudocod cum se poate determina dacă stiva crește de la adrese mari la adrese mici sau invers.

5. Câte pagini fizice vor ocupa stivele celor două procese rezultate în urma apelului **fork** exact înainte de apelul **exit**? Apelul **fork** se întoarce cu succes.

```
int main(void)
{
        char buf[3 * PAGE_SIZE];
        buf[0] = 'a';
        buf[PAGE_SIZE] = 'z';
        fork();
        buf[0] = 'b';
        exit(EXIT_SUCCESS);
}
```

6. În urma rulării secvenței de cod de mai jos fișierul **a.txt** conține șirul **222313**. După fiecare apel **write** actualizați următorul vector (cursor fd1, cursor fd2, cursor fd3, conținut fișier). Exemplu: inainte de primul write vectorul va fi (0, 0, 0, '').

```
fd1 = open("a.txt", O_RDWR | O_CREAT | O_TRUNC, 0644);
fd2 = open("a.txt", O_RDWR | O_CREAT | O_TRUNC, 0644);
fd3 = dup(fd1);

write(fd1, "1", 1);
write(fd2, "2", 1);
write(fd3, "3", 1);

pid = fork();


/* continuat pe coloana a doua */
```

```
switch (pid) {
case 0:
        write(fd1, "1", 1);
        write(fd2, "2", 1);
        write(fd3, "3", 1);
        break;
default:
        wait(&status);
        write(fd1, "1", 1);
        write(fd2, "2", 1);
        write(fd3, "3", 1);
        break;
}
```

1. Două procese (P1, P2) folosesc o zonă de memorie partajată, rulează într-un sistem preemptiv și execută secvența de cod de mai jos. Știind că valoarea inițială indicată de pointerul **counter** este 0 și că acesta indică în zona de memorie partajată, care este valoarea indicată de pointerul **counter** la finalul execuției celor două procese?

P1	P2
`if (*counter == 0)` `  (*counter)++;`	`if (*counter == 0)` `  (*counter)++;`

2. Pe un sistem dual processor cu 128MB de RAM și swap de 256MB rulează un sistem Linux. Câte procese se pot găsi la un moment dat în starea **RUNNING**, **READY** respectiv **WAITING**?

3. Un sistem folosește TLB în care fiecare intrare conține trei câmpuri (pid, frame, pagină virtuală). Care este avantajul acestei implementări față de o implementare care conține doar două câmpuri (frame, pagină virtuală)?

4. Descrieți în pseudocod cum se poate determina dacă stiva crește de la adrese mari la adrese mici sau invers.

5. Câte pagini fizice vor ocupa stivele celor două procese rezultate în urma apelului **fork** exact înainte de apelul **exit**? Apelul **fork** se întoarce cu succes.

```
int main(void)
{
        char buf[3 * PAGE_SIZE];
        buf[0] = 'a';
        buf[PAGE_SIZE] = 'z';
        fork();
        buf[0] = 'b';
        exit(EXIT_SUCCESS);
}
```

6. În urma rulării secvenței de cod de mai jos fișierul **a.txt** conține șirul **222313**. După fiecare apel **write** actualizați următorul vector (cursor fd1, cursor fd2, cursor fd3, conținut fișier). Exemplu: inainte de primul write vectorul va fi (0, 0, 0, '').

```
fd1 = open("a.txt", O_RDWR | O_CREAT | O_TRUNC, 0644);
fd2 = open("a.txt", O_RDWR | O_CREAT | O_TRUNC, 0644);
fd3 = dup(fd1);

write(fd1, "1", 1);
write(fd2, "2", 1);
write(fd3, "3", 1);

pid = fork();


/* continuat pe coloana a doua */
```

```
switch (pid) {
case 0:
        write(fd1, "1", 1);
        write(fd2, "2", 1);
        write(fd3, "3", 1);
        break;
default:
        wait(&status);
        write(fd1, "1", 1);
        write(fd2, "2", 1);
        write(fd3, "3", 1);
        break;
}
```

În conformitate cu ghidul de etică al Catedrei de Calculatoare, declar că nu am copiat la această lucrare. De asemenea, nu am ajutat și nu voi ajuta pe nimeni să copieze la această lucrare.

Nume:..... Grupă: .....

Semnătură: .....

În conformitate cu ghidul de etică al Catedrei de Calculatoare, declar că nu am copiat la această lucrare. De asemenea, nu am ajutat și nu voi ajuta pe nimeni să copieze la această lucrare.

Nume:..... Grupă: .....

Semnătură: .....