1. Ce valori se afiseaza la rularea codului C de mai jos ?
Cate procese au aparut in total (incluzand si procesul initial) ?
Desenati schitat arborescenta acestor procese, notand in dreptul fiecacarul proces valoarea afisata de el.
Justificati raspunsurile.

```
int k = 10;
if( fork() && !fork()) --k; else ++k;
printf("%d\n", k);
```

2. Rutina de tratare a intreruperii de ceas de pe un anumit calculator are nevoie de 2msec (incluzand supraincarcarea pentru comutarea proceselor) per tact de ceas. Ceasul genereaza tacti la frecventa de 70 Hz. Ce procent din capacitatea procesorului este dedicat ceasului ? Justificati raspunsul (aratati calculul).

3. Consideram urmatoarea varianta a solutiei lui Peterson de obtinere a excluderii mutuale (fata de varianta originala au fost interschimbate instructiunle marcate cu /***/):

```
        /* date partajate */
#define FALSE 0
#define TRUE 1
int turn;
int interested[2] = {FALSE, FALSE};

 /* proceduri prezente in fiecare proces */
void enter_region(int process){ /* procesul este 0 sau 1 */
  int other;
  other = 1-process;
  turn = process;            /***/
  interested[process] = TRUE;     /***/
  while(turn==process && interested[other]==TRUE);
}
void leave_region(int process) {
  interested[process]=FALSE;
}
```

Aratati printr-un scenariu de executie (o secventa de executare intercalata a celor doua procese aleasa de planificator) ca aceasta varianta este incorecta (cele doua procese se pot afla simultan in regiunea critica).

4. Scrieti o functie in limbajul C:

```
int nr(char *fis);
```

care inlocuieste in fisierul 'fis' literele mici de la 'a' la 'j' cu respectiv cifrele de la '0' la '9' (restul caracterelor raman nemodificate); f
functia returneaza numarul inlocuirilor facute sau -1 in caz de eroare.
se va opera direct pe fisierul respectiv (in particular, nu se vor folosi vectori sau fisiere auxiliare). Se vor indica fisierele header necesare.
de exemplu, daca fisierul continea initial:
.... la final, va contine ....
si functia va returna 3

5. Scrieti o functie in limbajul C:

int arh(char *director);

care creeaza in directorul curent un fisier obisnuit (regular) avand ca nume numele proprietarului real al procesului curent si continand o arhica a fisierelor obisnuite din directorul 'director' care ofera dreptde citire si scriere pentru proprietar si au fost actualizate in ora curenta din sistem.
Fisierul arhiva va contine specificatorul directorului 'director' (secventa de caractere terminata cu '\0'), apoi, pentru fiecare fisier arhivat:
- numele fisierului (secventa de caractere terminata cu '\0')
- dimensiunea fisierului (long, in format intern);
- continutul fisierului (secventa de caractere).
Functia returneaza numarul fisierelor arhivate sau -1 in caz de eroare.
La iesirea din functie, directorul si fisierele vor fi inchise.
Se vor indica fisierele header necesare.
Se poate presupune ca toti specificatorii de fisier au maxim 256 caractere.

6. Scrieti o functie :

int myalarm(unsigned long microsec, int sig);

care generalizeaza apelul "alarm()", in sensul ca se programeaza primirea unui semnal oarecare "sig" (nu doar SIGALRM), dupa un numar oarecare "microsec" de microsecunde (nu doar secunde intregi).
In acest scop, functia va efectua urmatoarele:
- daca "microsec" este != 0, genereaza un proces copil, care doarme "microsec" microsecunde cu "usleep()", apoi trimite parintelui semnalul "sig" si se termina cu "exit()"; pe de alta parte parintele continua executarea functiei, care retine intr-o variabila locala statica PID-ul copilului si returneaza imediat (nu blocheaza procesul); /
- daca exista un proces copil generat la un apel anterior (PID-ul sau este retinut in variabila locala statica), mai intai il omoara trimitandu-i semnalul SIGKILL;
- daca "microsec" este 0, functia se rezuma la omorarea eventualului proces copil generat la un apel anterior;
- functia returneaza 0 la succes si -1 la esec.
- se vor indica fisierele header necesare.