Adapost animale

Într-un adăpost există **C** animale, **I** îngrijitori, un paznic și un administrator.

Atât îngrijitorii cât și animalele sunt caracterizați de un ID și un Nume. ID-ul este un număr întreg, iar Numele este generat aleator de thread-ul Main.

Toate animalele trăiesc într-o curte comună.

Rolul îngrijitorilor este să hrănească animalele la intervale de timp aleatorii, **It**, (**It** este cuprinse între **X** și **Y** ms; fiecare îngrijitor își generează propriul **It** la începutul execuției), iar rolul animalelor este să consume hrana primită. Fiecare animal se hrănește la un interval fix de **Ct** ms. Îngrijitorii nu le pot hrăni pe fiecare independent, ci punând în curte porții de mâncare în spații special amenajate (există N spații pentru maxim N porții). Îngrijitorii nu oferă niciodată mâncare dacă există în curte hrană neconsumată. Fiecare îngrijitor depune la un moment dat un număr aleator de porții de hrană, număr cuprins între **1** și **N**. Fiecare animal consumă odată o singură porție de hrană.

Pentru monitorizarea activității, fiecare îngrijitor, după oferirea hranei, își notează într-un *Registru unic* (o listă sau orice altă structură de date potrivită)

[*id_ingrijitor*, *nr_porții*, *timp*, *"oferit"]*.

De fiecare dată când un animal mănâncă, senzorul pe care îl are inserat în piele înscrie în același *Registru unic* următoarea înregistrare*:*

[id_animal, *1*, *timp* ,*"consumat"*]

Administratorul iterează *Registrul unic* la un interval **At** ms și scrie într-un fișier toate înregistrările din acesta și încheie o scriere prin adăugarea pe unui rând nou cu mesajul "VERIFICAT: " urmat de: *nr porții oferite* și *nr porții consumate* de animale până în momentul iterării. La ultima iterare, verifică dacă bilanțul se închide (adică: nr porții existente în curte + nr porții oferite = nr porții consumate + nr porții rămase în curte).

La începutul aplicației în curte există **H** porții hrană.

Scrieți un program prin care simulați acțiunile actorilor implicați știind că aceștia acționează concurent.

Indicații:

Animalele, Îngrijitorii și Administratorul sunt modelați de obiecte ale unor clase de tip Thread. Curtea este un obiect "partajat" de thread-urile de tip Producător și de cele de tip Consumator. Clasa Curtea are cel puțin 2 metode: Ofera() și Consuma().

Caz de testare:

- C = 10 animale;
- I = 3 îngrijitori;
- X = 10 ms; Y = 30 ms;
- Ct = 7 ms;
- N = 5 porții;

- T = 6 ms;
- Pt = 20 ms;
- At = 12 ms;
- H = 0 porții.
- Fiecare Îngrijitor efectuează 100 de depuneri de porții de hrană;
- Animalele își termină execuția doar atunci când și-au încheiat execuția toți Îngrijitorii.
- Administratorul își încheie execuția după ce Animalele și-au încheiat execuția.

Observatie: pentru așteptare condiționată este necesar sa se folosească un mecanism de tip “wait-notify” (busy-waiting nu este acceptabil).