Olimpiada

La olimpiada nationala de informatica participa **(n**) concurenți din **J** județe și trebuie rezolvate **N** probleme. Participantii pot folosi o platforma unde pentru fiecare problema se poate încărca soluția care se va evalua (executa) automat și în urma acestei acțiuni se va asocia un anumit punctaj (maxim 100 puncte) pentru participant. Aceste punctaje parțiale se salvează sub forma - (**id_concurent, id_problema, punctaj, t_incarcare**), unde **t_incarcare** este timpul la care a fost încărcată soluția. Pentru fiecare problemă există un timp de start (**t_start**) la care participanții primesc problema și un deadline de încărcare a soluției. Fiecare rezultat se trimite la serverul central unde se se adaugă într-un *'pool',* cu capacitate maximă egală cu ***PMAX***.

Se fac clasamente pe fiecare problemă în parte și un clasament global care ține cont de punctaje globale pe toate problemele. Clasamentele pe fiecare problemă se fac in functie de punctaj și de rapiditatea de rezolvare **(t_incarcare -t_start).**

Pentru a realiza clasamentul global trebuie sa se faca agregarea acestor rezultate parțiale. Pentru fiecare concurent se calculează punctajul total, și în cazul unor punctaje egale diferențierea se face pe baza vitezei de rezolvare, care se estimează prin suma diferențelor de timp **(deadline- t_incarcare**) pentru fiecare problemă. În cazul în care o problemă nu a fost rezolvată deloc (nu s-a încărcat soluția, sau timpul de rezolvare a fost depășit) se va considera un timp egal cu diferenta **deadline-t_start.**

Preluarea rezultatelor parțiale din *pool*-ul de rezultate și actualizarea listelor corespunzătoare clasamentelor per problemă și a listei corespunzătoare clasamentului final se va face concurent folosindu-se un număr **N** de thread-uri. Fiecare thread preia o înregistrare din pool, actualizează lista corespunzătoare problemei și lista globală.

Output: listele clasamentelor per problemă și global. Aceste liste se salveaza la final in fisiere.

Simplificare: (simulare clienți cu thread-uri)

Ținând cont de aspectul distribuit al problemei, ar trebui realizată o aplicație client-server în care clienții să trimită rezultatele parțiale către serverul central. Transmiterea rezultatelor s-ar face în blocuri de cate **max** rezultate parțiale. Pentru simplificare nu se va considera implementarea client-server pentru preluarea rezultatelor parțiale. Se va considera că aceste rezultate sunt deja salvate în memoria serverului central în fișiere corespunzătoare fiecărei județ - cu înregistrări de tipul (**id_concurent, id_problema, punctaj, t_incarcare**). Adăugarea în *pool*-ul de rezultate parțiale se face concurent cu ajutorul mai multor thread-uri – **r,** un thread adaugă în pool câte **max** înregistrări odata..

Testare

N=5-
t_start_1=0;
t_start_2= deadline_1= 30
t_start_3 = deadline_2= 30+45
t_start_4 = deadline_3= 30+45+60
t_start_5 = deadline_4= 30+45+60+20
deadline_5= 30+45+60+20+70

n=1000
J=10 (considerăm **100 de concurenți în fiecare județ)**
PMAX=50
max=10
r=4

Rezultatele parțiale se genereaza folosind valori random pentru punctaj (valori intre 10 si 100) și pentru timpul de încărcare (valori intre **(t_start_i, deadline_i+delta)** unde **delta =(deadline_N-t_start_1)/(N*10).**

Timpul de execuție trebuie măsurat și afișat.

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