Challenge3

Nume: Cotet Teodor Mihai 333CA
user: teodor_mihai.cotet

Am rezolvat și prima și a doua problema dintr-un foc.

Pentru a simula problema am creat o coada în care bag „oameni” cu o direcție random și un timp random de a parcurge poteca. Nu permit extragerea din coada simultana => sync pe Main.o1
Ca sa fac paralelizarea trecerilor am folosit 3 semafoare:
-> 2 pentru a nu permite mai multe de 5 persoane pe poteca într-o direcție
-> unul (binar) pentru a nu permite ca 2 oameni sa meargă în directii opuse simultan
Ideea consta în actulizarea semaforului ultim descris (opus) în funcție de celelalte 2. Încerc sa îl actulizez când intra prima persoana dintr-un anumit sens, sau pleacă ultima (le țin evidenta).

Când fac aceste planificari nu permit concurenta => sync pe Main.o1 la intrari și sync pe Main.o2 la iesiri din tunel. Eu dau aquire la semaforul opus (încerc sa schimb sensul de mers, trebuie să fie liber pe alt sens) atunci când tocmai a venit cineva pe direcția pe care vreau sa schimb și ca sa nu dau de mai multe ori acquire pe opus fac asta doar pentru prima persoana care vrea sa meargă în sensul respectiv (`s[p.direction].availablePermits() == 4`). Problema ar fi ca ca partea de ieșire și de intrare (de planificare) s-ar putea executa în paralel așa ca ca s-ar putea întâmpla ceva de genul: s-ar putea face un release, apoi imediat un acquire pe același sens, și atunci codul de după release care elibera semaforul opus nu-l va mai elibera (se eliberează când nu mai e nicio persoana pe drum), deși semaforul care a făcut acquire este unicul și deci primul care pe direcția respectiva, astfel va încerca sa dea acquire pe opus dar nu va reuși, așa ca am vrut ca în timp de ce se executa etapa de planificare de final sa nu se execute și cea de planificare de început => am folosit un sync pe o2. Cu toate astea dacă ajungem într-o stare în care se încerca `s[p.direction].acquire();` și nu se reușește pentru ca deja sunt 5 oameni pe drum, starea de release nu se va executa niciodată (pt ca nu se pot executa simultan, iar noi suntem blocați în cea de început) => deadlock. Așa ca am făcut doua while care au rolul de a face ceva de genul: „nu ai voie sa executi bucățile astea 2 în paralel, dar dacă ești blocat în etapa de planificare a intrarii și se etapa de planficare de final vrea sa modifice ceva, dă-i ei prioritate) => cele 2 while-uri

```
while(s[p.direction].availablePermits() == 0);
```

și `while(opus.availablePermits() == 0);`

Modul de planificare este următorul: primul care venit primul servit.