

Esercizio 2 [punti 18] – Monitor



Si consideri un piccolo ponte che collega le due rive di un fiume. Il ponte, situato all'interno di un parco naturalistico, è accessibile sia da autoveicoli che da biciclette e pedoni.

Gli utilizzatori del ponte possono quindi appartenere a 3 categorie diverse:

- Auto
- Biciclette
- Pedoni.

Ogni utente, una volta entrato, impiega un tempo arbitrario per l'attraversamento, e successivamente esce.

Per le sue caratteristiche il ponte può essere usato in due modi diversi:

- Modalità *ecologica*, nella quale il ponte può essere attraversato soltanto da pedoni e biciclette.
- Modalità *autoveicoli*, nella quale il ponte può essere percorso soltanto da automobili.

Quindi, un'auto che vuole entrare sul ponte quando è utilizzato da bici e/o pedoni (cioè, in modalità ecologica) è costretta ad aspettare; analogamente una bici o un pedone che vuole accedere al ponte quando esso è percorso da auto (cioè, è in modalità autoveicoli) deve aspettare.

Non vi sono vincoli legati al verso di percorrenza.

Il ponte ha una capacità massima N , che esprime il numero massimo di persone che esso può accogliere contemporaneamente (a questo scopo, si supponga che ogni automobile equivalga a 10 persone).

Si rappresentino gli utenti (Pedoni, Bici e Auto) mediante thread concorrenti.

Si realizzi ² una politica di sincronizzazione tra i thread basata sul concetto di monitor che tenga conto dei vincoli dati, e che **nell'accesso al Ponte tenga conto delle priorità**, secondo il seguente ordine:

1. Auto
2. Pedoni
3. Bici

² Riguardo al linguaggio di programmazione e agli strumenti di sincronizzazione, il candidato può scegliere tra il linguaggio C/libreria `pthread`, o il linguaggio Java.