

17 dicembre 2015 – SOM

ESERCIZIO ADA/GO

Si consideri il pronto soccorso di un ospedale.

Il pronto soccorso è organizzato come segue.

Ogni paziente accede inizialmente a un servizio di “**TRIAGE**”, attraverso il quale vengono valutate le condizioni di gravità del paziente, al quale il servizio assegna un **codice di urgenza** (ROSSO, GIALLO, VERDE o BIANCO).

Una volta ottenuto il codice di urgenza, il paziente viene indirizzato verso l’ambulatorio nel quale sarà visitato.

In particolare, i pazienti si classificano in due categorie: pazienti **adulti e bambini**.

Pertanto:

- ogni paziente adulto si presenterà presso l’ambulatorio “**ADULTI**” con il proprio codice di urgenza;
- ogni bambino si presenterà presso l’ambulatorio “**PEDIATRIA**” con il proprio codice di urgenza.

Presso ogni ambulatorio sono disponibili **N medici**, ognuno dei quali visita un paziente alla volta, contemporaneamente agli altri medici.

La politica di gestione di ogni ambulatorio prevede che le richieste di visita vengano servite in ordine di codice di priorità, secondo l’ordine seguente:

1. codice rosso
2. codice giallo
3. codice verde
4. codice bianco

Realizzare un’applicazione a scelta nel linguaggio ADA o nel linguaggio GO nella quale **Pazienti**, **Triage**, **Ambulatorio Pediatrico** e **Ambulatorio Adulti** siano rappresentati da ***processi*** (TASK o goroutine) **distinti**. La sincronizzazione tra i processi dovrà tenere conto dei vincoli dati.

20 dicembre 2013 – SOM

ESERCIZIO SEMAFORI

Un grande albergo è situato in una località turistica particolarmente arida, nella quale non è presente un acquedotto.

Pertanto l'albergo è attrezzato con una grande **cisterna** che viene rifornita da un'autocisterna **quando è vuota**.

La cisterna ha **capacità pari a Max** litri.

La cisterna alimenta gli **N rubinetti** collocati all'interno dell'albergo.

A questo proposito si assumano le seguenti ipotesi semplificative:

1. ogni rubinetto, una volta azionato, prelevi una quantità di acqua costante pari a 1 litro. Si supponga che ogni prelievo richieda una quantità di tempo trascurabile.
2. La cisterna viene riempita quando è completamente vuota.
3. Durante il riempimento della cisterna i rubinetti non possono essere azionati.

Si realizzi un'applicazione concorrente in **java** che rappresenti l'**autocisterna** (che rifornisce la cisterna dell'albergo) e i **rubinetti** mediante thread concorrenti e che rispetti i vincoli dati mediante un'opportuna politica di sincronizzazione realizzata tramite semafori.