

Esercizi Go

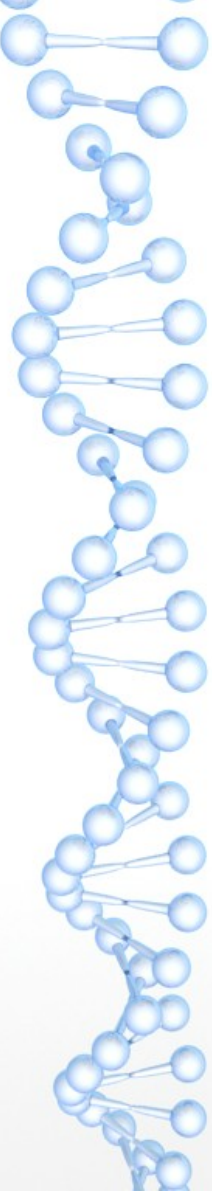


Regole generali

- Le consegne degli esercizi per casa valgono fino a 3 punti nell'esame finale.
- Le consegne sono 4, di difficoltà crescente.
- E' consentito che vi consultiate tra voi, ma ognuno dovrà scrivere la sua soluzione
- Per chiedere info mandate una mail a

Se chiedete consigli su come implementare il codice ossevate queste regole:

- Nella mail scrivete la logice del programma, almeno a parole o in pseudocode
- Scrivete bene l'ordine in cui avete capito che devono avvenire le istruzioni, cioè cosa deve fare prima il vostro programma e cosa può essere fatt contemporaneamente.
- Scrivete le domande su un file .doc (o equivalenti) che allegate alla mail. Ripercorrete quello che avete capito scrivendo in nero, nei punti in cui avete le domande, scrivetele in rosso. In questo modo il contesto della domanda è chiaro.



Consegna 4 (difficile)

Vi è dato un programma (che trovate su Moodle: tunnelBug.go) che simuli la seguente situazione:

- Ci sono due gruppi di palline G1 e G2 in due luoghi diversi L1 e L2 uniti da un tunnel. In L1 e in L2 ci sono due persone P1 e P2.
- La persona P1 vuole lanciare tutte le palline in G1 da L1 a L2, e viceversa P2 vuole lanciare lanciare le palline in G2 da L2 a L1.
- Il tunnel è stretto, ci può passare solo una pallina alla volta. Se due palline vengono lanciate nel tunnel contemporaneamente, tornano al punto di partenza (immediatamente). Una pallina attraversa il tunnel in un secondo (`time.Sleep(time.Second)`).
- Una persona non può lanciare una pallina finché quella che ha lanciato precedentemente non è arrivata a destinazione o non ha incontrato una pallina che andava in senso contrario.
- Ci sono due gruppi di palline e due routine che lanciano le palline da un capo all'altro. Le routine attendono un tempo casuale (`time.Sleep(time.Duration(rand.Intn(2))*time.Second)`) prima di lanciare una nuova palla. Le routine finiscono quando nel relativo gruppo le palline finiscono.



Che fare

- Debuggate i deadlock! Ci sono errori nel codice. Fate esperimenti, vedete cosa succede, ripercorrete la logica e rendete mutualmente esclusive la parti di codice che devono avvenire una alla volta. Possono esserci soluzioni diverse e non mi aspetto che raggiungete tutti la stessa.
- Per vedere il funzionamento del programma, inserite delle stampe che vi mostrino in che ordine girano i thread e aggiungete degli sleep per forzare un certo ordine dell'esecuzione.
- Non ci sono commenti nel code. Dovete capire voi cosa fa.



Hint

- Nel codice, se il tunnel è libero, aspetto un oggetto che mi viene inviato se l'altra routine trova il tunnel occupato. E se l'altra routine trova il tunnel occupato ma mi invia il messaggio quando l'altra routine è già andata oltre? Una modifica semplice evita un deadlock, ma crea altri problemi.
- Una possibile soluzione prevede tre modifiche principali al codice: un altro channel per essere sicuro che solo un thread stia lavorando su una certa parte di codice; la modifica di un channel esistente e una select.
- Siate creativi, non è detto che siano le uniche possibilità. Provate a decommentare gli sleep per vedere se trovate deadlock sistematici e provate a aggiungere sleep prima e dopo le aprti di codice che inserite per vedere se trovate deadlock sistematici.