DISCLAIMER

Il materiale contenuto nel drive è stato raccolto e richiesto tramite autorizzazione ai ragazzi frequentanti il corso di studi di Informatica dell'Università degli Studi di Salerno. Gli appunti e gli esercizi nascono da un uso e consumo degli autori che li hanno creati e risistemati per tanto non ci assumiamo la responsabilità di eventuali mancanze o difetti all'interno del materiale pubblicato.

Il materiale sarà modificato aggiungendo il logo dell'associazione, in tal caso questo possa recare problemi ad alcuni autori di materiale pubblicato, tale persona può contattarci in privato ed elimineremo o modificheremo il materiale in base alle sue preferenze.

Ringraziamo eventuali segnalazioni di errori così da poter modificare e fornire il miglior materiale possibile a supporto degli studenti.

CoScienze
Associazione

DESCRIZIONE CHAINCODE GO

Descrivere il seguente chaincode go:

```
package main
import (
    "fmt"
    "sync"
)

func main() {
    slice := []string{"a", "b", "c", "d", "e"}
    sliceLength := len(slice)
    c := make(chan string, 5)
    for i := 0; i < sliceLeght; i++ {
        c <- slice[i]
        }
    var wg sync.WaitGroup
    wg.Add(sliceLength)

    fmt.Println("Running for loop...")

    for i := 0; i < sliceLength; i++ {
        go func(i int, c chan string) {
            defer wg.Done()
            val := <-c
            fmt.Printf("i: %v, val: %v\n", i, val)
        }(i,c)
    }

    fmt.Println("Doing other stuff")

    wg.Wait()

    fmt.Println("Finished for loop")
}
```


RISPOSTA

Package main indica che siamo nel pacchetto principale main.
Import indica l'importazione delle librerie necessarie per la compilazione del codice. Vengono importate la libreria standard "fmt" e la libreria che permette la sincronizzazione dei thread "sync".
Poi abbiamo la funzione main, dove viene creato un'array di stringhe e viene utilizzata una variabile ausiliaria per la lunghezza dell'array. Con make viene creato il canale c che accetta in input valori di tipo stringa ed ha un limite di bufferizzazione massimo di 5, ovvero può bufferizzare massimo 5 elementi. Poi abbiamo un ciclo dove carica gli elementi dell'array nel canale. Successivamente viene dichiaro WaitGroup che è un oggetto di sincronizzazione dove ogni istanza ha un contatore interno che viene incremento con add e decrementato con done. In particolare viene fatto l'incremento del numero di elementi dell'array. Dopo di chè viene fatta una stampa "running for loop" . A questo punto abbiamo un ciclo che chiama una funzione anonima che prende in input un valore intero e il canale di stringhe. Questa funzione con defear rimanda l'esecuzione dell'istruzione ovvero "done" fin quando il contesto chiamante non termina. Poi viene passato il contenuto del canale c all'interno della variabile val e viene fatta una stampa del valore alla posizione iesima. I parametri finali (i, c) servono per la visibilità di queste ultime. Viene stampato poi il messaggio "Doing other stuff" e le goroutine, ovvero i thread attendono in wait(). Quindi viene stampato un messaggio "Finished for loop"

Descrivere il seguente chaincode go:

```
package main
```

```
import (
 "fmt"
 "time"
)
func main() {
 c1 := make(chan string)
 c2 := make(chan string)

 go func() {
 time.Sleep(1 * time.Second)
 c1 <- "one"
 }()

 go func() {
 time.Sleep(2 * time.Second)
 c2 <- "two"
 }()

 fmt.Println("received", <-c2)
 fmt.Println("received", <-c1)
}
```

RISPOSTA:

Package main indica che siamo nel pacchetto principale main

Nell'import abbiamo l'importazione della libreria standard "fmt" e quella del tempo "time"

Nella funzione principale main abbiamo la creazione dei canali c1 e c2 , di tipo stringa, senza limite di buffer e sono bloccanti, quindi bisogna esplicitare chi riceve su quel canale altrimenti c'è errore.

La prima funzione anonima attende 1 secondo e poi carica 'one' nel canale c1.

La seconda funzione anonima attende 2 secondi e poi carica 'two' nel canale c2.

Infine il risultato finale sarà la stampa di c2 e dopo circa 2 secondi e subito dopo quella di c1 dato che avrebbe stampato dopo 1 secondo ma c'era prima la stampa di c2.

Queste due stampe rappresentano i ricevitori dei due canali non bufferizzati, senza le quali il programma avrebbe dato errore.

Descrivere il seguente chaincode go:

```
package main
import (
```

```
    " fmt "
    " time "
)
func numbers (c1 chan int , done chan bool ) {
  for i := 1; i <= 5; i++ {
      time.Sleep (20 * time.Millisecond )
      c1 <- i
  }
  done <- true
}
func alphabets (c2 chan string , done chan bool ) {
   for i := 'a'; i <= 'e'; i++ {
      time . Sleep (40 * time . Millisecond )
      c2 <- string (i)
   }
   done <- true
}
func main () {
   c1 := make ( chan int , 2)
   c2 := make ( chan string , 2)
   done := make ( chan bool )
   go numbers (c1 , done )
   go alphabets (c2 , done )
   go func () {
       for i := 1; i <= 5; i++ {
             fmt . Println (<-c1)
       }
   }()
   go func () {
       for i := 1; i <= 5; i++ {
             fmt . Println (<-c2)
       }
   }()
   for i := 0; i < 2; i++ {
       <-done
   }
   fmt . Println (" main terminated ")
}
```

RISPOSTA:

package main indica che siamo all'interno del pacchetto principale main.
Con import vengono importate le librerie necessarie alla compilazione del del codice: qella standard "fmt" e quella per il tempo "time".
Abbiamo la funzione numbers che prende in input due canali, uno intero "c1" e uno booleano "done". Questa funzione caricherà ad ogni iterazione, con un'attesa di 20 millisecondi, un valore numerico nel canale c1. Al termine del ciclo caricherà true all'interno del canale done.
Successivamente abbiamo la funzione alphabets che prende in input due canali, uno intero "c2" e uno booleano "done". Questa funzione caricherà ad ogni iterazione, con un'attesa di 40 millisecondi", una stringa nel canale c2. Al termine del ciclo caricherà true all'interno del canale done.
Infine c'è la funzione main, dove viene creato il canale c1 con input un intero è un limite massimo di bufferizzazione di 2, ovvero il canale può bufferizzare massimo 2 elementi per volta. Poi viene creato il canale c2 con input uno string è un limite massimo di bufferizzazione di 2. Poi abbiamo la creazione del canale done con in input un booleano che servirà come canale di attesa del termine delle go routine dei primi due canali. Successivamente vengono chiamate le go routine numbers e alphabet. Vengono poi chiamate le suddette funzioni più due funzioni anonime. La prima funzione anonima stampa i valori del canale c1 e la seconda funzione anonima stampa i valori del canale c2.
Per ultimo abbiamo un ciclo che attende la terminazione delle goroutine precedenti e al termine del ciclo stampa di un messaggio di terminazione.

Un **chanel** è una «sottorete» di comunicazione privata tra due o più membri della rete, che serve a scambiarsi transazioni in maniera privata.