ESERCIZI CON ECLIPSE

Ho un silos di grano che ha n (>0 e < 20) bidoni cilindrici che possono contenere fino a 12 quintali di grano (usiamo un **intero** per indicare la quantità di grano).

Una **gru** può versare del grano (da 1 fino a 5 quintali) in un singolo cilindro a scelta dell'operatore, mentre un **nastro** trasportatore può prelevare 2 quintali di grano da tutti i contenitori (se un cilindro ha meno di due quintali, si svuota – cioè ad esempio se voglio prelevare 2 da tutti i cilindri ma uno di essi ha solo 1 quantale, esso viene semplicemente svuotato).

La gru e il nastro non funzionano mai tutte e due insieme, però almeno uno dei due funziona sempre.

Il sistema per sicurezza non permette mai di avere più di 10 quintali in ogni bidone (anche se ce ne starebbero 12) (la politica per garantire questo quando la gru riempie i cilindri è a tua scelta, ma cerca di essere permissivo).

Inizialmente i bidoni sono tutti con 1 quintale di grano.

5. Asmeta: progetto COGNOME_SILOS_ASMETA

ASSUMI CHE n = 3 ➔ Ho 3 cilindri solamente

Scrivi la specifica asmeta (usa più costrutti possibili, ad esempio derivate, macro rule etc). Non usare la **regola** forall, usa il par.

Scrivi le seguenti proprietà e dimostrale (o prova che sono false)

Proprietà P1:

Tutti i cilindri non hanno mai più di 10 quintali di grano

Proprietà P2:

Se attivo il nastro, nello stato successivo tutti i cilindri avranno al massimo 8 quintali

Proprietà P3:

Se il primo cilindro è vuoto, finché non si attiva la gru, esso rimane vuoto. (usa weak until aw)

Scrivi inoltre almeno una proprietà che è giustamente falsa e controlla che il model checker trovi il contro esempio atteso (puoi partire anche dalle proprietà sopra).

Scrivi almeno un'altra proprietà di safety e una liveness che siano vere.

Scrivi uno scenario avalla di un caso significativo (ad esempio riempio tutti i cilindri).

ATTENZIONE: **commenta** ogni dominio, ogni funzione e ogni proprietà con almeno una riga di commento prima delle sua dichiarazione o definizione

6. JML progetto COGNOME_SILOS_JML

Scrivi il codice in Java con in contratti opportuni (la classe Silos ha uno **costruttore**, il metodo **gru** mette il grano (data la posizione del cilindro e il grano da mettere) e **nastro** che toglie 2 quintali di grano a tutti i silos.

Cerca di scrivere sia le precondizioni, che le postcondizioni dei metodi.

Cerca di scrivere anche invarianti.

Testa i contratti JML con una classe main in cui chiami i diversi metodi.

Prova anche a modificare il codice e controlla che i contratti siano violati. Documenta bene le violazioni e le loro cause in commenti.

7. KEY progetto COGNOME_SILOS_KEY

Copia il progetto precedente, togli tutte le cose statiche (anche il main) non usare gli enumerativi. Dimostra con key che i contratti siano rispettati.

Type	Target	Contract	Proof Re...	Proof Re...
Silos	Silos(int)	JML operation contract 0	New Pro...	Closed
Silos	nastro()	JML operation contract 0	New Pro...	Open
Silos	gru(int, i...	JML operation contract 0	New Pro...	Closed

Io ne ho chiusi 2 su 3. Ho usato diverges true e anche invarianti oltre a loop invariant.

8. MODEL BASED TESTING

Traduci lo scenario avalla nel caso di test JUNIT per il codice java che hai scritto nel progetto COGNOME_SILOS_JML

Prova ad introdurre un errore nel codice del Silos e controlla che il caso di test lo catturi. Documenta il tutto con commenti nel codice e nel codice.