CI4251 - Programación Funcional Avanzada
Tarea 3

Ernesto Hernández-Novich
86-17791
<emhn@usb.ve>

Mayo 26, 2015

Eat all the empanadas!

Rafita prepara empanadas en una recóndita playa del oriente del país. La calidad de sus empanadas de cazón es legendaria, tanto que la contratan para servir los pasapalos de las fiestas playeras organizadas por la banda usual de desadaptados que allí pulula en cualquier puente o feriado largo.

Para esos eventos, Rafita se presenta con su gran paila, que tiene capacidad para freír m empanadas. Una vez fritas, echa *todas* las empanadas, simultáneamente, en un gran colador plástico de color indeterminado, raído por el tiempo, los elementos y el manoseo. Hecho esto, y consecuencia del calor por la paila y el inclemente sol, Rafita se sienta a tomar cerveza.

Cuando un parroquiano está hambriento, se acerca a buscar empanadas. Los parroquianos de la fiesta no hacen cola, sino que meten mano y se sirven del colador, tomando y comiendo **una** empanada a la vez, siempre y cuando haya empanadas disponibles. Cuando el parroquiano come, vuelve a tomar cerveza por un rato, hasta que tenga hambre nuevamente.

Si un parroquiano tiene hambre, pero no hay empanadas, le avisa a Rafita para que prepare más.

Se desea que Ud. construya una simulación de este comportamiento usando Haskell. En este sentido, tome en cuenta lo siguiente:

- Rafita es la única que prepara empanadas y siempre prepara *exactamente* m empanadas en lote. Naturalmente, Rafita será modelada con un hilo de ejecución.
- Rafita tarda un tiempo al azar en preparar las empanadas. A efectos de esta simulación, considere un tiempo al azar entre 3 y 5 segundos. La simulación debe indicar claramente `Rafita está cocinando` y pasado el intervalo `Rafita sirvió las empanadas`.
- En la fiesta puede haber un número arbitrario de parroquianos. Su simulación debe estar parametrizada para $n > 0$ parroquianos, y cada parroquiano debe estar representado por un hilo independiente.
- Los hábitos de bebida y comida de los parroquianos son muy variables, así que debe considerar que transcurre un tiempo al azar entre 1 y 7 segundos desde el momento en que empuña su empanada, la consume,

vuelve a su bebida, y tiene hambre de nuevo. La simulación debe indicar claramente `Parroquiano N come empanada` cuando comienza a comer y `Parroquiano N tiene hambre` cuando vuelve a buscar una empanada.

- Rafita tiene ingredientes infinitos para preparar las empanadas, y los parroquianos no tienen nada más productivo que hacer, de manera que una vez que comienza la fiesta, sólo termina cuando se interrumpe la simulación con `Ctrl-C`.

Presente **dos** soluciones para este problema: una empleando técnicas clásicas de concurrencia (sincronización con `MVar`) y otra empleando Memoria Transaccional (`STM`).

```
import System.Random

randomSeed :: Int
randomSeed = 42

classic :: Int -> Int -> IO ()
classic m n = undefined

transactional :: Int -> Int -> IO ()
transactional m n = undefined
```

En ambos casos, cuando se interrumpa la simulación, presente un resultado sumario indicando:

```
Rafita preparó R empanadas.

Parroquiano 1:    P1
Parroquiano 2:    P2
...
Parroquiano N:    PN
Total:            T
```

Donde `R` es el total de empanadas, necesariamente múltiplo de m; `P1` hasta `PN` corresponden a la cantidad de empanadas que comió cada parroquiano, respectivamente, y `T` resulta de la suma de esos consumos.

Finalmente, para poder comprobar la fidelidad de su simulación es necesario que use números pseudo-aleatorios, como los que se proveen en `System.Random` *fuera* del monad `IO`, usando `randomSeed` como semilla.