UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2015

24 de Agosto de 2015

TPE - Fat Food

Aclaraciones

- Para aprobar la totalidad del TP es necesario tener aprobado cada uno de sus módulos.
- **No** está permitido el uso de **acum** para la resolución.

Una cadena de locales de comidas rápidas decide instalarse en nuestro país y su dueño, un payaso de amplia trayectoria, desea contratar nuestros servicios para que modelemos los problemas relacionados a la entrega de mercadería. Luego de una serie de reuniones, se ha podido recabar suficiente información acerca del problema, tomando las decisiones de diseño que se detallan a continuación. Nuestra misión, si decidimos aceptarla, es especificar todo lo que se pide a continuación.

1. Tipos

`tipo` Empleado = String;
`tipo` Energia = $\mathbb{Z}$;
`tipo` Cantidad = $\mathbb{Z}$;
`tipo` Bebida = Pesti Cola, Falsa Naranja, Se ve nada, Agua con Gags, Agua sin Gags;
`tipo` Hamburguesa = McGyver, CukiQueFresco (Cuarto de Kilo con Queso Fresco), McPato, Big Macabra;

2. Combo

El combo agrupa una bebida con un sandwich e indica la energía que requiere, por parte del empleado, para expenderlo.

```
tipo Combo {
       observador bebida (c: Combo) : Bebida;
       observador sandwich (c: Combo) : Hamburguesa;
       observador dificultad (c: Combo) : Energia;

       invariante dificultadHasta100 : energiaEnRango(dificultad(c));
}
```

1. `problema nuevoC` (b:Bebida, s:Hamburguesa, d:Energia) = `result` : Combo
   Crea un combo con bebida b, sandwich s y dificultad d.
2. `problema bebidaC` (c: Combo) = `result` : Bebida
   Devuelve la bebida correspondiente al combo c.
3. `problema sandwichC` (c: Combo) = `result` : Hamburguesa
   Devuelve el sandwich correspondiente al combo c.
4. `problema dificultadC` (c: Combo) = `result` : Energia
   Devuelve la dificultad correspondiente al combo c.

3. Pedido

El pedido es el encargo realizado por el cliente. Posee un número identificador, registra quién atendió dicho pedido y en qué consiste.

```
tipo Pedido {
       observador numero (p: Pedido) : Z;
       observador atendio (p: Pedido) : Empleado;
       observador combos (p: Pedido) : [Combo];

       invariante numeroPositivo : numero(p) > 0;
       invariante pideAlgo : |combos(p)| > 0;
}
```

5. `problema nuevoP` (n: $\mathbb{Z}$, e: Empleado, cs: [Combo]) = `result` : Pedido
   Crea un pedido con número n, empleado e y una lista de combos cs.

6. `problema numeroP` (p: Pedido) = `result` : $\mathbb{Z}$
   Devuelve el número correspondiente al pedido p.

7. `problema atendioP` (p: Pedido) = `result` : Empleado
   Devuelve el empleado asociado al pedido p.

8. `problema combosP` (p: Pedido) = `result` : [Combo]
   Devuelve la lista de combos correspondiente al pedido p.

9. `problema agregarComboP` (p: Pedido, c: Combo)
   Agrega el combo c al final de la lista de combos del pedido p

10. `problema anularComboP` (p: Pedido, i:$\mathbb{Z}$)
    Dado un pedido p, anula el combo ubicado en la posición i del observador $combos(p)$

11. `problema cambiarBebidaComboP` (p: Pedido, b: Bebida, i:$\mathbb{Z}$)
    Quién no ha ido con un amigo a un local de fat food y a último momento éste decide cambiar de opinión acerca de la bebida pedida ... que prefiere bajas calorías, que si es de esta línea de gaseosas prefiere agua, etc. Así, el local necesita de esta herramienta fundamental: Dado el pedido p, modifica el pedido para cambiar la bebida del combo ubicado en la posición i del observador $combos(p)$ por la bebida b.

12. `problema elMezcladitoP` (p: Pedido) Modifica el pedido por un pedido de igual tamaño en el que no haya combos repetidos, respetando la mayor cantidad de combos posibles. Si fuera necesario modificar algún combo, deben utilizarse bebidas y sandwiches que se encuentren en el pedido original. **Importante:** tener en cuenta que esto no siempre es posible (requiere ...)

4. Local

El local expende algunas de las bebidas y sandwiches existentes en el mercado. Eso se puede consultar a través de los observadores *bebidasDelLocal* y *sandwichesDelLocal*, respectivamente. Al dueño de la cadena le interesa llevar el stock de bebidas y hamburguesas del local. Para ello se introducen los observadores *stockBebidas* y *stockSandwiches*. La lista de empleados y ex-empleados se puede consultar a través de los observadores *empleados* y *desempleados*. En el caso de los empleados, se puede consultar la energía con la que cuentan para atender un pedido, a través del observador *energiaEmpleado*. La ventas realizadas por el local, se pueden consultar a través del observador *ventas*. Así, el siguiente Tipo representa el local.

`tipo Local` {
  `observador stockBebidas` (l: Local, b: Bebida) : Cantidad ;
    `requiere` $b \in bebidasDelLocal(l)$;
  `observador stockSandwiches` (l: Local, h: Hamburguesa) : Cantidad ;
    `requiere` $h \in sandwichesDelLocal(l)$;
  `observador bebidasDelLocal` (l:Local) : [Bebida] ;
  `observador sandwichesDelLocal` (l:Local) : [Hamburguesa] ;
  `observador empleados` (l: Local) : [Empleado] ;
  `observador desempleados` (l: Local) : [Empleado] ;
  `observador energiaEmpleado` (l: Local, e: Empleado) : Energia ;
    `requiere` $e \in empleados(l)$;
  `observador ventas` (l: Local) : [Pedido] ;

  `invariante hayBebidasySonDistintas` : $|bebidasDelLocal(l)| > 0 \land distintos(bebidasDelLocal(l))$;
  `invariante haySandwichesySonDistintos` : $|sandwichesDelLocal(l)| > 0 \land distintos(sandwichesDelLocal(l))$;
  `invariante stockBebidasPositivo` : $(\forall b \leftarrow bebidasDelLocal(l))stockBebidas(l, b) \geq 0$;
  `invariante stockSandwichesPositivo` : $(\forall h \leftarrow sandwichesDelLocal(l))stockSandwiches(l, h) \geq 0$;
  `invariante empleadosDistintos` : $distintos(empleados(l) + +desempleados(l))$;
  `invariante energiaHasta100` : $(\forall e \leftarrow empleados(l))energiaEnRango(energiaEmpleado(l, e))$;
  `invariante empleadosQAtendieronDelLocal` : $(\forall v \leftarrow ventas(l))atendio(v) \in empleados(l) + +desempleados(l)$;
  `invariante ventasCorrelativas` : ... ;
  `invariante combosDeLocal` : ... ;
}

13. `invariante` ventasCorrelativas
    Los números de pedidos correspondiente a las ventas son correlativos. Es decir, no hay repetidos y no faltan números en el medio. Ejemplo: 4,5,6,7 son correlativos. 4,5,7 no lo son.

14. `invariante` combosDeLocal
    Los combos correspondiente a las ventas, son combos que se expenden en el local.

15. `problema stockBebidasL` (l: Local, b:Bebida) = `result` : Cantidad
    Devuelve el stock de bebida b en el local l.

16. `problema stockSandwichesL` (l: Local, h:Hamburguesa) = `result` : Cantidad
    Devuelve el stock de sandwich h en el local l.

17. `problema bebidasDelLocalL` (l: Local) = `result` : [Bebida]
    Devuelve la lista de bebidas que se expenden en el local l.

18. `problema sandwichesDelLocalL` (l: Local) = `result` : [Hamburguesa]
    Devuelve la lista de sandwiches que se expenden en el local l.

19. `problema empleadosL` (l: Local) = `result` : [Empleado]
    Devuelve la lista de empleados que atienden en el local l.

20. `problema desempleadosL` (l: Local) = `result` : [Empleado]
    Devuelve la lista de ex-empleados del local l.

21. `problema energiaEmpleadoL` (l: Local, e:Empleado) = `result` : Energia
    Devuelve la energía del empleado e en el local l.

22. `problema ventasL` (l: Local) = `result` : [Pedido]
    Devuelve las ventas del local l.

23. `problema unaVentaCadaUno` (l:Local) = `result` : Bool
    Indica si las ventas que realizaron los empleados actuales del local l, fueron hechas rotando de manera estricta. Ejemplo: Si los empleados actuales del local l son A,B,C, y D es un ex-empleado y se realizaron las ventas **B**,D,**A**,D,**C**,**B**,D,D,**A**,**C**,**B**,**A** debería devolver verdadero(las ventas de D no deben considerarse ya que corresponde a un ex-empleado).

24. `problema venderL` (l: Local, p:Pedido)
    Agrega el pedido p a la lista de ventas del local l, en caso de ser posible. Por normas de la empresa, el pedido sólo puede ser atendido por un empleado del local. Si el empleado no cuenta con la energía suficiente como para atender el pedido, dicha persona es despedida y por lo tanto pasa a formar parte del staff de desempleados. Al realizar la venta, el stock debe ser actualizado. Lo mismo sucede con la energía del empleado. La energía del empleado termina siendo la energía que tenía previo a vender el pedido, menos la sumatoria de las dificultades de los combos del pedido.

25. `problema candidatosAEmpleadosDelMesL` (l: Local) = `result` : [Empleado]
    Devuelve la lista de empleados del local l que más ventas hicieron y dentro de este grupo, los que más combos vendieron.

26. `problema sancionL` (l: Local, e:Empleado, n:Energia)
    Modifica el local l, sacándole una cantidad n de energía al empleado e. Si el empleado queda con un nivel de energía negativa, el mismo es despedido del local.

27. `problema elVagonetaL` (l: Local) = `result` : Empleado
    Devuelve el empleado actual que más descanso se tomó entre pedido y pedido. Por ejemplo, si los empleados son A, B y C, y atendieron los pedidos en el siguiente orden: A-**B-C-B-C-B-C-B-C-B**-A, el descanso más largo es el de A (9 ventas), porque B sólo tiene descansos de una venta, y el descanso más lardo de C es de 2 ventas. En el caso de que un empleado no hubiese atendido pedidos, se toma que su descanso es la cantidad total de ventas que se hicieron en el local. En el caso de que un empleado hubiese atendido al menos un pedido, tener en cuenta que también se contabilizan como descansos a) la cantidad de pedidos entre el comienzo y el primer pedido que atendió, y b) entre el último pedido que atendió y el total de pedidos.

28. `problema anularPedidoL` (l: Local, n:$\mathbb{Z}$) Elimina el pedido número n de la lista de ventas. Para realizar esta operación el que atendió el pedido debe ser empleado del local. Se tiene que actualizar el stock y la energía del empleado.

29. `problema agregarComboAlPedidoL` (l: Local, c: Combo, n:$\mathbb{Z}$)
    Sólo en el caso de ser posible (ver stock), se agrega el combo c al pedido número n de ventas.
    Importante: Esta operación sólo se puede realizar en caso de que, quien haya atendido el pedido n, aún siga siendo empleado y tenga energía suficiente para mantenerse como empleado al entregar ese combo extra.

5. Auxiliares

`aux distintos` (ls:[T]) : Bool $= (\forall i, j \leftarrow [0..|ls|), i \neq j) ls_i \neq ls_j$;
`aux energiaEnRango` (e: Energia) : Bool $= 0 \leq e \leq 100$;