UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación– Algoritmos y Estructura de Datos I
Parcial de Especificación– 17/04/2013

LU:
Apellidos:
Nombres:

Aclaraciones: El parcial NO es a libro abierto. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos. **Entregar cada ejercicio en hoja separada.**
Importante: Para la resolución del parcial **NO** es necesario ni está permitido el uso de **acum**.

El pueblito de Rastis, es un pequeño pueblito rectangular cuya característica principal es que todas sus calles, o bien se orientan de Norte a Sur, o bien de Este a Oeste. Todas las calles que van de Norte a Sur son paralelas entre sí y perpendiculares a las que van de Este a Oeste. El Dr. Rastri, Intendente de dicho pueblo, decidió implementar un sistema revolucionario de seguridad: instalar cámaras en varios puntos del pueblo. Las cámaras instaladas realmente no son de muy buena calidad y suelen descomponerse periódicamente. Es por ello que se decidió registrar en qué intervalos de tiempo se encuentran activas. Para modelar dicho escenario, se cuenta con la siguiente representación:

tipo Calle=String;
tipo Cruce=(Calle, Calle);
tipo Intervalo=($\mathbb{Z}$, $\mathbb{Z}$);

```
tipo Camara {
  observador filmacion (c :Camara) : [Intervalo];
  invariante positivos :
    (∀n ← filmacion(c))0 ≤ primero(n) < segundo(n);
  invariante noSeSolapan : ...;
  invariante intervalosOrdenados : ...;
}

tipo Pueblo {
  observador calles (p: Pueblo) : [Calle];
  observador sonParalelas (p :Pueblo, c1:Calle, c2:Calle) : Bool;
      requiere c1 ∈ calles(p) ∧ c2 ∈ calles(p);
  observador tieneCamara (p :Pueblo, c:Cruce) : Bool;
  observador camara (p :Pueblo, c:Cruce) : Camara;
      requiere tieneCamara(p, c);
  invariante sinCallesRepetidas : sinRepetidos(calles(p));
  invariante reflexiva : (∀c ← calles(p))sonParalelas(p, c, c);
  invariante simetrica : (∀c1, c2 ← calles(p))
    sonParalelas(p, c1, c2) → sonParalelas(p, c2, c1);
  invariante transitiva : (∀c1, c2, c3 ← calles(p))
    sonParalelas(p, c1, c2) ∧ sonParalelas(p, c2, c3) →
    sonParalelas(p, c1, c3);
  invariante alMenosUnCruce :
    (∃c1, c2 ← calles(p))sonPerpendiculares(p, c1, c2);
  invariante noHay3oMasPerpendicularesEntreSi :
    ¬(∃c1, c2, c3 ← calles(p))sonPerpendiculares(p, c1, c2) ∧
    sonPerpendiculares(p, c2, c3) ∧ sonPerpendiculares(p, c1, c3);
  invariante paraCamaraTieneQueSerCruce : (∀c1, c2 ← calles(p))
    sonParalelas(p, c1, c2) → ¬tieneCamara(p, (c1, c2));
}
```

aux sinRepetidos (l : [T]) : Bool = $(\forall i \leftarrow [0..|l|))\, l_i \notin l(i..|l|)$;
aux sonPerpendiculares (p:Pueblo, c1:Calle, c2:Calle) : Bool = $\neg sonParalelas(p, c1, c2)$;

Aclaración: Los intervalos son cerrados. Es decir, si filmacion(c) = (1,3), en los momentos 1, 2 y 3, la camara está filmando.

Ejercicio 1. [30 puntos]

a) **[5 p.]** Completar el invariante *intervalosOrdenados* del tipo Camara, que garantiza que la listas de intervalos están ordenados de menor a mayor, utilizando como referencia de ordenamiento el primer componente de la tupla.

b) **[5 p.]** Completar el invariante *noSeSolapan* del tipo Camara, que garantiza que no hay dos intervalos de tiempo que se solapen entre sí. Ejemplo: [(1,3),(2,4)] se solapan. [(1,2),(2,4)] vamos a tomar como que también se solapan. [(1,2),(5,6)] no se solapan.

c) **[15 p.]** Especificar el `aux momentosMediaticos` (p: Pueblo) : $[\mathbb{Z}]$ = ;
Este *aux* devuelve, de menor a mayor, los momentos en que más cámaras estuvieron filmando a la vez.
Es decir, si el pueblo cuenta con 3 cámaras (c1,c2,c3) y los intervalos de filmación de las cámaras son los siguientes:
c1: (1,3) y (6,8)
c2: (3,5) y (7,10)
c3: (3,8)
entonces momentosMediaticos(p) debería devolver [3,7,8] (en ese orden) ya que en esos momentos hubieron 3 cámaras filmando a la vez.
Aclaración: Puede suponer que al menos hay una cámara en el pueblo con al menos un intervalo de filmación.

d) **[5 p.]** Especificar el `aux cantidadDeCruces` (p: Pueblo) : $\mathbb{Z}$ = ;
Este *aux* calcula la cantidad total de cruces entre calles que hay en el Pueblo.
Es decir, si el pueblo cuenta con 3 calles (A,B,C) y A $\parallel$ B, A $\perp$ C y por lo tanto B $\perp$ C , entonces el pueblo cuenta con 2 cruces.

Ejercicio 2. [20 puntos] Especificar el `problema secuenciaDeEncendido` (p: Pueblo) = `result` : [Camara] . Este problema, dado un Pueblo *p*, devuelve la secuencia (de menor a mayor) en que se fueron encendiendo las cámaras. El momento de encendido de una cámara es el que corresponde a la primer componente de su primer intervalo. Es decir, dada la cámara c1 cuyos intervalos de filmación son (3,6) y (11,20), el momento de encendido de dicha cámara es 3, por ser el primer componente de su primer intervalo.

Ejercicio 3. [20 puntos]
Especificar el `problema silencioDeCamara` (p: Pueblo, n:Intervalo) = `result` : [Intervalo] . Este problema, dado un Pueblo *p*, devuelve la lista de intervalos de tiempo en que NINGUNA cámara estuvo filmando a la vez. Tener en cuenta que dichos intervalos deben estar contenidos en el intervalo pasado por parámetro. Pueden haber intervalos consecutivos en la solución, como por ejemplo: (1,2),(3,4), pero los intervalos no pueden estar solapados.

Ejercicio 4. [30 puntos] Especificar el `problema asfaltando` (p: Pueblo, cn: Calle, cp: Calle, cs: [Camara]) . Por orden del intendente (y antes de las elecciones ☺), se debe modificar el pueblo *p* agregando la calle nueva *cn* paralela a la calle *cp*. Las nuevas cámaras *cs* deben estar sobre la calle que se acaba de agregar (el Intendente no aclaró sobre qué cruces).