Apellidos: **Nombre:** **Matrícula:**

(3 puntos) 7. A continuación mostramos una especificación formal de un recurso para jugar al popular juego de mesa *Los nómadas que cantan*. Los desarrolladores han ideado un recurso compartido para representar las materias primas que tiene el jugador, siendo estas materias primas cereal, agua y madera.

C-TAD MateriasPrimas
OPERACIONES
ACCIÓN cargarCereal:
ACCIÓN cargarAgua:
ACCIÓN cargarMadera:
ACCIÓN avanzar:
ACCIÓN reparar:
SEMÁNTICA
DOMINIO:
TIPO: $\mathit{MateriasPrimas} = (\mathit{cereal} : \mathbb{N} \times \mathit{agua} : \mathbb{N} \times \mathit{madera} : \mathbb{N})$
INICIAL: $\text{self} = (0, 0, 0)$
INVARIANTE: $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} < 10$

CPRE: $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} + 1 < 10$
cargarCereal
POST: $\text{self}^{pre} = (c, a, m) \land \text{self} = (c + 1, a, m)$

CPRE: $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} + 1 < 10$
cargarAgua
POST: $\text{self}^{pre} = (c, a, m) \land \text{self} = (c, a + 1, m)$

CPRE: $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} + 1 < 10$
cargarMadera
POST: $\text{self}^{pre} = (c, a, m) \land \text{self} = (c, a, m + 1)$

CPRE: $\text{self}.\mathit{cereal} > 0 \land \text{self}.\mathit{agua} > 0$
avanzar
POST: $\text{self}^{pre} = (c, a, m) \land \text{self} = (c - 1, a - 1, m)$

CPRE: $\text{self}.\mathit{agua} > 0 \land \text{self}.\mathit{madera} > 0$
reparar
POST: $\text{self}^{pre} = (c, a, m) \land \text{self} = (c, a - 1, m - 1)$

Se pide: implementar este recurso compartido usando monitores siguiendo la metodología vista en clase. En cuanto al código de desbloqueos os recomendamos implementar en el método `desbloqueoSimple()` un código genérico para todas las operaciones.

```
import es.upm.babel.cclib.Monitor;

public class MateriasPrimas {
  private Monitor mutex;
  private Monitor.Cond condCargar;
  private Monitor.Cond condAvanzar;
  private Monitor.Cond condReparar;
  private int cereal = 0;
  private int agua = 0;
  private int madera = 0;

  public MateriasPrimas(){
    cereal = 0;
    agua = 0;
    madera = 0;
    mutex = new Monitor();
    condCargar = mutex.newCond();
    condAvanzar = mutex.newCond();
    condReparar = mutex.newCond();
  }

  public void cargarCereal() {
    mutex.enter();
    if (cereal + agua + madera == 10) {
      condCargar.await();
    }
    cereal = cereal + 1;

    desbloqueoSimple();
    mutex.leave();
  }

  public void cargarAgua() {
    mutex.enter();
    if (cereal + agua + madera == 9) {
      condCargar.await();
    }
    agua = agua + 1;

    desbloqueoSimple();
    mutex.leave();
  }

  public void cargarMadera() {
    mutex.enter();
    if (cereal + agua + madera == 10) {
      condCargar.await();
    }
    madera = madera + 1;

    desbloqueoSimple();
    mutex.leave();
  }

  public void avanzar() {
    mutex.enter();
    if (cereal == 0 || agua == 0) {
      condAvanzar.await();
    }
    cereal = cereal - 1;
    agua = agua - 1;

    desbloqueoSimple();
    mutex.leave();
```

```
    }

    public void reparar() {
      mutex.enter();
      if (agua == 0 || madera == 0) {
        condReparar.await();
      }
      agua = agua - 1;
      madera = madera - 1;

      desbloqueoSimple();
      mutex.leave();
    }

    private void desbloqueoSimple() {
      if (cereal > 0 && agua > 0 && condAvanzar.waiting() > 0) {
        condAvanzar.signal();
      }
      else if (agua > 0 && madera > 0 && condReparar.waiting() > 0) {
        condReparar.signal();
      }
      else if (cereal + agua + madera < 10) {
        condCargar.signal();
      }
    }
  }
```