Exercise 123. In Dante's Inferno, he describes a visit to Hell. In a very recently discovered chapter, he encounters five people sitting at a table with a pot of stew in the middle. Although each one holds a spoon that reaches the pot, each spoon's handle ismuch longer than each person's arm, so no one can feed him- or herself. They are famished and desperate. Dante then suggests "why do not you feed one another?"
The rest of the chapter is lost.

1. Write an algorithm to allow these unfortunates to feed one another. Two or more people may not feed the same person at the same time. Your algorithm must be, well, starvation-free.

Nosso algoritmo é, pois o "unfortunate" alimenta o "unfortunate" imediatamente a direita e quem foi alimentado só será alimentado novamente depois que o "unfortunate" que o alimentou também for alimentado.

Teorema: Não é possível um infeliz B ser alimentado 2x seguidas sem que o vizinho da esquerda, infeliz A, tenha sido alimentado 1x;

OBS: -> [A] -> [B] -> [C] -> [D] -> [E] ->

>> Suponha que é possível o infeliz B ser alimentado 2x seguidas sem que o infeliz A seja alimentado 1x;

Prova por contradição:
- Supunha que não. [Supomos que a negação do teorema seja verdadeira.]
- Suponha que exista um infeliz B que foi alimentado 2x seguidas antes de A ser alimentado 1x. [Deve-se deduzir uma contradição.]
- Se o infeliz B foi alimentado 1x, temos o seguinte caso:
- A seguinte condição [*if (this.getSopa().getInfelizes()[this.posicao()].isAlimentado() && !this.isAlimentado())*] garante que o infeliz A não alimente o B antes de A ser alimentado.
- Logo, A só irá alimentar B pela 2a vez depois de ter sido alimentado pelo infeliz E. [Essa contradição mostra que a suposição é falsa e, desta forma, o teorema é verdadeiro.]

Temos também a tabela verdade que apresenta os estados das threads em cada situação:

ESTADO ATUAL		**ESTADO FUTURO**	
Infeliz A	**Infeliz B**	**Infeliz A**	**Infeliz B**
Alimentado	**Alimentado**	**Descansa**	**Descansa**

Alimentado	¬ Alimentado	Alimenta B	É alimentado
¬ Alimentado	Alimentado	É alimentado	Alimenta A
¬ Alimentado	¬ Alimentado	Alimenta B	É alimentado

2. Discuss the advantages and disadvantages of your algorithm. Is it centralized, decentralized, high or low in contention, deterministic or randomized?

Possui contenção pois a thread só prossegue a execução depois que todas as demais threads foram alimentadas. O nosso algoritmo é determinístico pois a thread só pode alimentar o vizinho imediatamente a direita, e esta característica cria uma baixa contenção pois o máximo de espera será um ciclo completo em que todos os infelizes tenham sidos alimentados. Nosso algoritmo é descentralizado pois não existe um controle das threads em um único ponto, e sim cada thread fica livre para executar suas ações respeitando as condições estabelecidas.

Exercise 215. Extend the LockObject class to support concurrent readers.

215 - O LockObject já possui os readers de forma concorrente, assim não precisa ser alterado.

Exercise 216. InTinyTM, theLockObjectclass'sonCommit() handler first checks whether the object is locked by another transaction, and then whether its stamp is less than or equal to the transaction's read stamp.

1. Give an example showing why it is necessary to check whether the object is locked.
2. Is it possible that the object could be locked by the committing transaction?
3. Give an example showing why it is necessary to check whether the object is lockedbeforechecking the version number.