FreeRTOS Exercises in mutexes and queues – solving deadlock

a) dining philosophers

Program the dining philosopher problem. Use 4 philosopher threads. The philosophers sit at a table with 4 plates and 4 forks: F1, F2, F3 and F4 - each fork is modelled as a mutex. A philosopher repeats this pattern: take right fork, take left fork, eat, put left fork, put right fork, think. Let each philosopher output its actions on the little stm32F429I screen. Let's use the Chinese version of the problem (chopsticks are much easier to draw instead of forks ;-). We create 4 philosophers – the chopstics will be placed either vertically or horizontally. Depict the philosophers as small circles – one in each corner of a square on the screen. Draw a big circle for the table. On the table draw a circle for each plate. A chopstick can be drawn at one of 3 possible places according to its usage: on the table between two plates or at the philosopher to the left or at the philosopher to the right. You can in addition also let the philosophers output text messages in the bottom of the screen such as "Ph1 takes Ch1" - in this way you achieve redundant output, so you can ease your debugging of your graphical output.

Some ways to deal with the lurking deadlock:

a) take chopstics in ranking order

b) voluntary release of chopstics when deadlock occurs

c) take two chopstics at a time or no one

d) Queue of right-fork tickets for avoiding the lurking deadlock.
Solve the problem with the lurking deadlock by introducing a queue of right-fork tickets. Before philosophers can take their right fork they need to have a right-fork ticket, which must be returned after the usage of the right-fork. (The implementation of freeRTOS that we have does not implement semaphores – thus we use a queue instead in this exercise).