Exercise 6: Wet Vending Machine Controller

A coffee vending machine controller, Vend, accepts two coins for coffee; an ok is given after the first coin and then either a second coin (for coffee) or an abort (for refund) is accepted:

```
Vend = ?coin. ![ok,mutex]. (Coffee | Refund)
Coffee = ?[mutex,coin]. !coffee. (Coffee | Vend)
Refund = ?[mutex,abort]. !refund. (Refund | Vend)
```

Exercise: compile that to the Combinatorial Strand Algebra; if you do it by the U(P) algorithm you can then heavily hand-optimize it.

Each Vend iteration spawns two branches, Coffee and Refund, waiting for either coin or abort. The branch not taken in the mutual exclusion is left behind; this could skew the system towards one population of branches. Therefore, when the Coffee branch is chosen and the system is reset to Vend, we also spawn another Coffee branch to dynamically balance the Refund branch that was not chosen; conversely for Refund.

Standard questions can be asked: what happens if somebody inserts three coins very quickly? Or somebody presses refund twice? Etc.