Concurrent Programming
Practical 2: Dining Philosophers
originally by Gavin Lowe

It is recommended that you read the material from the section of the course notes on the Dining Philosophers before you attempt this practical. The code from the lectures is on the course website.

The aim of the practical is to investigate some variants of the Dining Philosophers, which aim to avoid deadlocks. You should implement and test two of the three variants below. **Optional:** implement all three.

Deadline: Practical sessions in Week 4.

Your report should be in the form of a well-commented program, together with a brief discussion of any design considerations and of your results.

Variant 1: a right-handed philosopher

In the standard version of the dining philosophers, all the philosophers are left-handed: they pick up their left fork first. Implement a variant where one of the philosophers is right-handed, i.e. she picks up her right fork first.

Variant 2: using a butler

Now consider a variant using an extra process, which represents a butler. The butler makes sure that no more than four philosophers are ever simultaneously seated.

Variant 3: using timeouts

Now consider a variant where, if a philosopher is unable to obtain her second fork, she puts down her first fork, and retries later. In the Dining Philosophers example from lectures, all delays are a few hundred milliseconds, so the delay here should be of a similar order.

In order to implement this idea, you will need to use the following function (in OutPort[A]).

```
def sendWithin(millis: Long)(x: A): Boolean
```

This tries to send x for up to millis milliseconds, but if the message isn't received within that time, then it times out and returns. The command returns a boolean, indicating whether x was successfully received.

If we reach a state where all philosophers are blocked, it's possible that they all put down their forks at the same time, and then all retry at the same time, leading to them being blocked again. How can we avoid this (with high probability)?