

# Concurrent Programming Practical

## 3: Hamming Numbers

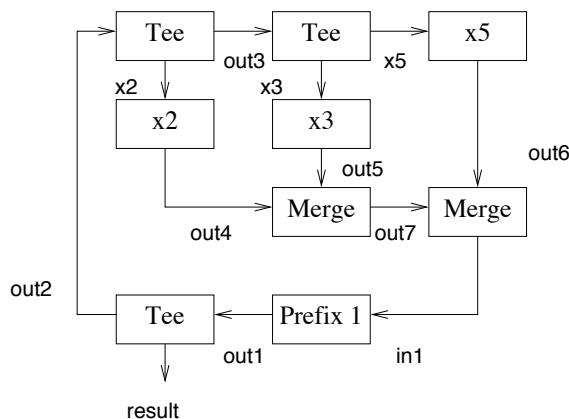
It is recommended that you read the material from the section of the course on *Message Passing* (using Channels) before you attempt this practical.

The aim of the practical is to implement a dataflow network to generate an ordered sequence of *Hamming Numbers*, with no duplicates. The (inductive) definition of the Hamming numbers is as follows:

- 1 is a Hamming number.
- If  $h$  is a Hamming number, then so are  $2h$ ,  $3h$  and  $5h$ .

Another way of thinking of the Hamming numbers is that they are those whose only prime factors are 2, 3 and 5.

The illustration below shows a network that solves the problem. If you have read the material suggested, it should be fairly obvious what the various components are expected to do. The component **Merge** receives two ordered streams, and merges them into a single ordered stream, without repetitions.



## Your Task

Implement the network.

You may make use of components from the package `io.threadcso.component` if you like (but note that `io.threadcso.component.merge` has a different behaviour from the `Merge` required here). In some cases you will need to implement your own components, especially to answer some of the questions below.

If you implement the network with **no buffering** on any of the channels, you will find that it deadlocks. Explain precisely how this deadlock arises: you should explain what state each component is in in the deadlocked state. You might need to adapt some of the components in order to understand this behaviour.

Re-implement the network with **sufficient buffering** to enable it to find the value of the 1000th Hamming number. The declaration

```
val c = OneOneBuf[T](size)
```

defines **c** to be an asynchronous buffered channel, passing data of type **T**, with capacity **size**. Think about which channels need to be buffered, and which can remain unbuffered.

**Optional:** Modify the components so that all the processes **terminate cleanly once the 1000th Hamming number has been output.**

**Just for fun:** Generalize the construction of the network so that it generates the sequence of “Hamming-like” numbers whose factors are given as a list of parameters to your program.

## Reporting

Your report should be in the form of a well-commented program, together with brief answers to the above points.

Bernard Sufrin  
15<sup>th</sup> January, 2016(4.1325)