Concurrency and occam-π

occam-π Exercises (extra)

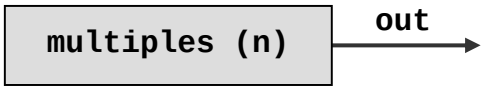
[For all these exercises, starter files are given in your **`exercises`** folder. The file for this is **`e1.occ`**.]

Exercise e1:

A *strictly ascending* sequence of numbers is one in which there are no repeats and each number is the sequence is bigger than the one before it.

Write a process **`ascending.merge.2`** that connects two strictly ascending input streams of **`INT`**s to an output stream. The output stream must be the strictly ascending *merge* of the input streams – i.e. it must contain all the numbers from its input streams in ascending order, eliminating repeats:

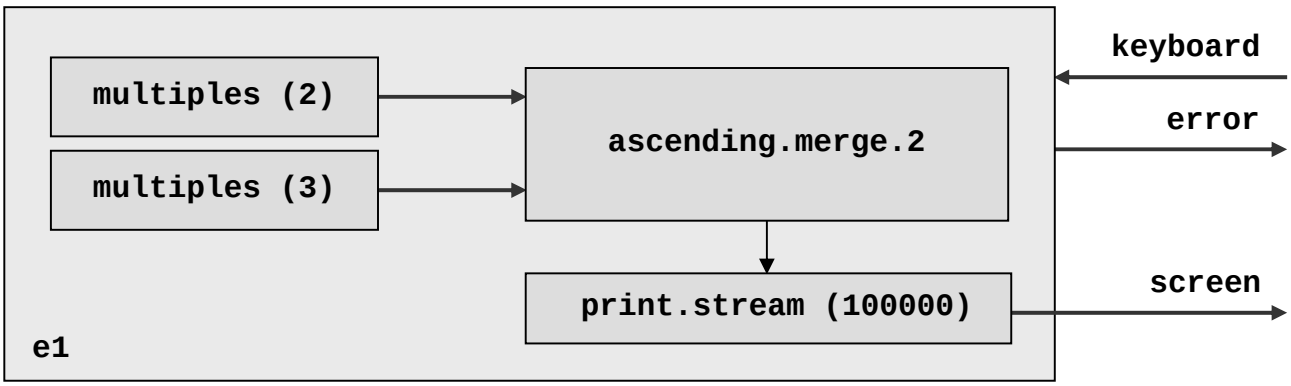
To test this, use the following process:



where **`multiples(n)`** is a process that outputs the stream of (**`INT`**) numbers:

```
0*n, 1*n, 2*n, 3*n, 4*n, 5*n, 6*n, ...
```

which, so long as n > 0, is strictly ascending. Test using the following circuit:

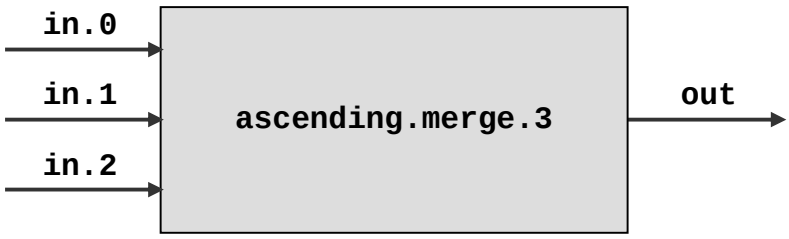


which should produce the ascending sequence of multiples of 2 and 3, with no repeats:

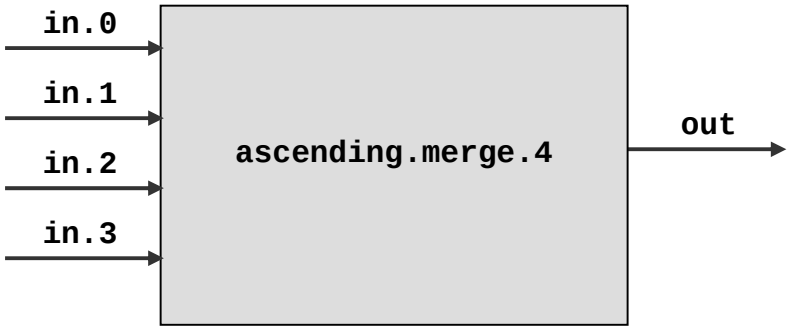
0, 2, 3, 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 26, 27, 28, …

/continued

Now, write a three-way merge process:



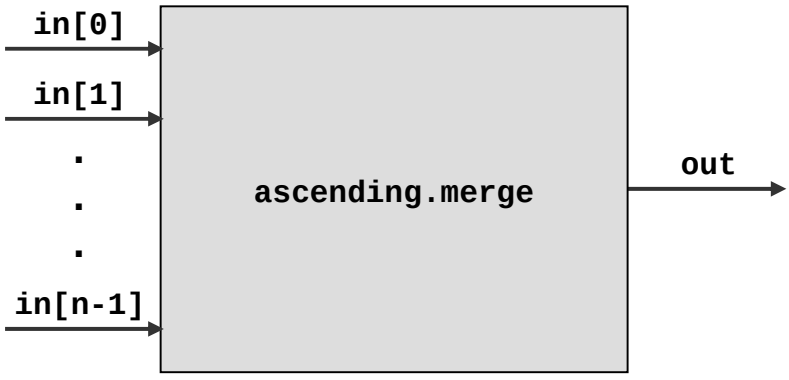
and a four-way:



Hint: build these with a network of **`ascending.merge.2`** processes.

Modify your **`e1`** process to demonstrate that these work.

Challenge: write a merge process for an *array* of strictly ascending input streams:



Hint: build this as a network of **`ascending.merge`** and **`ascending.merge.2`** processes (i.e. with recursion and concurrency). ☺ ☺ ☺

Finally, modify **`e1`** to demonstrate **`ascending.merge`**.