We want to find the smallest number in \( 0, \ldots, n \) with property \( p \). Linear search solves the problem. But evaluating \( p \) is expensive; let us say it takes time \( 1 \), and all else is free. The fastest solution is to evaluate \( p \) on all \( n \) numbers concurrently, and then find the smallest number that has the property. Write a program without concurrency for which the sequential to concurrent transformation gives the desired computation.

§ We introduce array \( A : [n*bin] \). We define the desired result \( R \), invariant \( I \), and helper specification \( P \) as follows.

\[
R = \neg (\exists j : 0, \ldots, h \cdot p j) \land (p h \lor h = n) \\
I = \forall j : 0, \ldots, i \cdot A j = p j \\
P = I n \land \neg (\exists j : 0, \ldots, h \cdot p j) \Rightarrow R
\]

Now the program is

\[
R \iff 10 \Rightarrow \text{for } i = 0, \ldots, n \text{ do } I i \Rightarrow I (i+1) \text{ od} \\
I i \Rightarrow I (i+1) \iff A i = p i \\
P \iff \text{if } h = n \text{ then ok else if } A h \text{ then ok else } h := h + 1 \text{. P fi fi}
\]

The \( n \) iterations of the \textbf{for}-loop can be executed concurrently.

We can express the result of the sequential to concurrent transformation at source as follows.

\[
R \iff 10 \Rightarrow \text{for } i = 0, \ldots, n \text{ do } I i \Rightarrow I (i+1) \text{ od} \\
I i \Rightarrow I (i+1) \iff \text{if } i = n \text{ then ok else if } A i := p i \parallel (i := i + 1. I i \Rightarrow I (i+1)) \text{ fi} \\
P \iff \text{if } h = n \text{ then ok else if } A h \text{ then ok else } h := h + 1 \text{. P fi fi}
\]

To understand the execution, it might help to unroll the recursion a little: in the refinement of \( I i \Rightarrow I (i+1) \), replace the recursive call \( I i \Rightarrow I (i+1) \) by what it calls:

\[
\text{if } i = n \text{ then ok else if } A i := p i \parallel (i := i + 1. I i \Rightarrow I (i+1)) \text{ fi}
\]

And maybe do the same once more.