Homework 7: Message Passing

1. [Eager Stack] The lazy stack presented in class can be energy efficient, but it can be very slow. After popping \(k\) elements off of the stack, the request to pop another element has to propagate through \(k\) empty stack elements, and the result has to propagate back through these elements as well. Modify the behavior of the lazy stack element so that all push or pop operations take constant time.

2. [Distributed Computation]

Let \(\otimes\) be an associative operation, i.e., an operation that satisfies \(a \otimes (b \otimes c) = (a \otimes b) \otimes c\) for all \(a, b, c\). Given a vector \(x = (x_0, x_1, \ldots, x_{N-1})\), we wish to compute the quantities \(y_i = x_0 \otimes x_1 \otimes \cdots \otimes x_i\), for \(0 \leq i < N\).

The values \(x_0, \ldots, x_{N-1}\) can be received on channels \(X_0, \ldots, X_{N-1}\) respectively. The result values \(y_0, \ldots, y_{N-1}\) are to be produced on channels \(Y_0, \ldots, Y_{N-1}\). This could be implemented as a single process:

\[
\begin{align*}
  &\ast [ X_0 ? x_0; \ X_1 ? x_1; \ \cdots \ X_{N-1} ? x_{N-1}; \\
  &y_0 := x_0; \ y_1 := y_0 \otimes x_1; \ y_2 := y_1 \otimes x_2; \ \cdots \ y_{N-1} := y_{N-2} \otimes x_{N-1} \\
  &Y_0! y_0; \ Y_1! y_1; \ \cdots \ Y_{N-1}! y_{N-1};
\end{align*}
\]

However, this implementation requires us to perform \(O(N)\) operations in parallel, and it requires a process with \(O(N)\) memory. Design a collection of CHP processes with \(O(1)\) memory and channels that will perform this computation in at most \(O(\log N)\) operations performed sequentially.