Erlang

After learning Erlang language for a week, I think “Erlang is the concurrency language that makes hard things easy and easy things hard.” That is, Erlang provides what many other languages can’t: scalable concurrency and reliability. In addition to that, it tries to make processes as lightweight as possible. Like Scala’s actor represents an object, in Erlang, an actor represents a lightweight process. In addition to that, I was fascinated to mix lists and tuples. In the same way, I would match up the data structures, assign variables to the values in the tuples. I assume, this is really helpful. I was intrigued to notice bit-matching through Erlang where I was able to pack several pieces of data into one byte quite easily. With these operations, I believe that Erlang is surprisingly powerful for low-level tasks. Since Erlang is dynamically typed, I didn’t have to worry too much about assigning types to data elements. I believe Erlang is particularly useful for tail recursion similar to that of Prolog. Unlike other programming languages, I discovered that it was fast and easy to calculate Factorial and Fibonacci of large numbers in Erlang. Because of this I was interested to determine the maximum integer size in Erlang. Although Erlang was more confusing to understand, I was able to express bigger ideas with less code. Using Erlang, it was unusual to see list comprehensions, an elegant and powerful abstraction that can quickly transform lists with generators and filters. Although concurrency was inherently difficult to achieve, it seemed like a piece of cake through Erlang. In the same way, I was intrigued to know that I could easily link two processes together.

Erlang's syntax lacks the beauty and simplicity comparing to that of Ruby. Variable starting with an uppercase letter and being in an immutable state was eerie to me comparing to other OOP/functional languages. Along with that, the syntax was pretty compelling and hard to grasp. In addition to that, the syntax of control structures was harder to understand, and make it work which I would do it clearly in other languages. That is, despite the fact Erlang solves complex problems easily with functions, it was difficult to solve simple problems using higher order functions. To me, the weirdest thing of Erlang was the use of unnecessary punctuations (",", ";" and ".") as syntax. And other oddities like the conditional presentation of an array of numbers as strings.

In, sum up, I strongly believe that I would use Erlang for its assets such as dynamic and reliable, lightweight (share-nothing processes), OTP (the enterprise libraries) and let it crash the procedure.