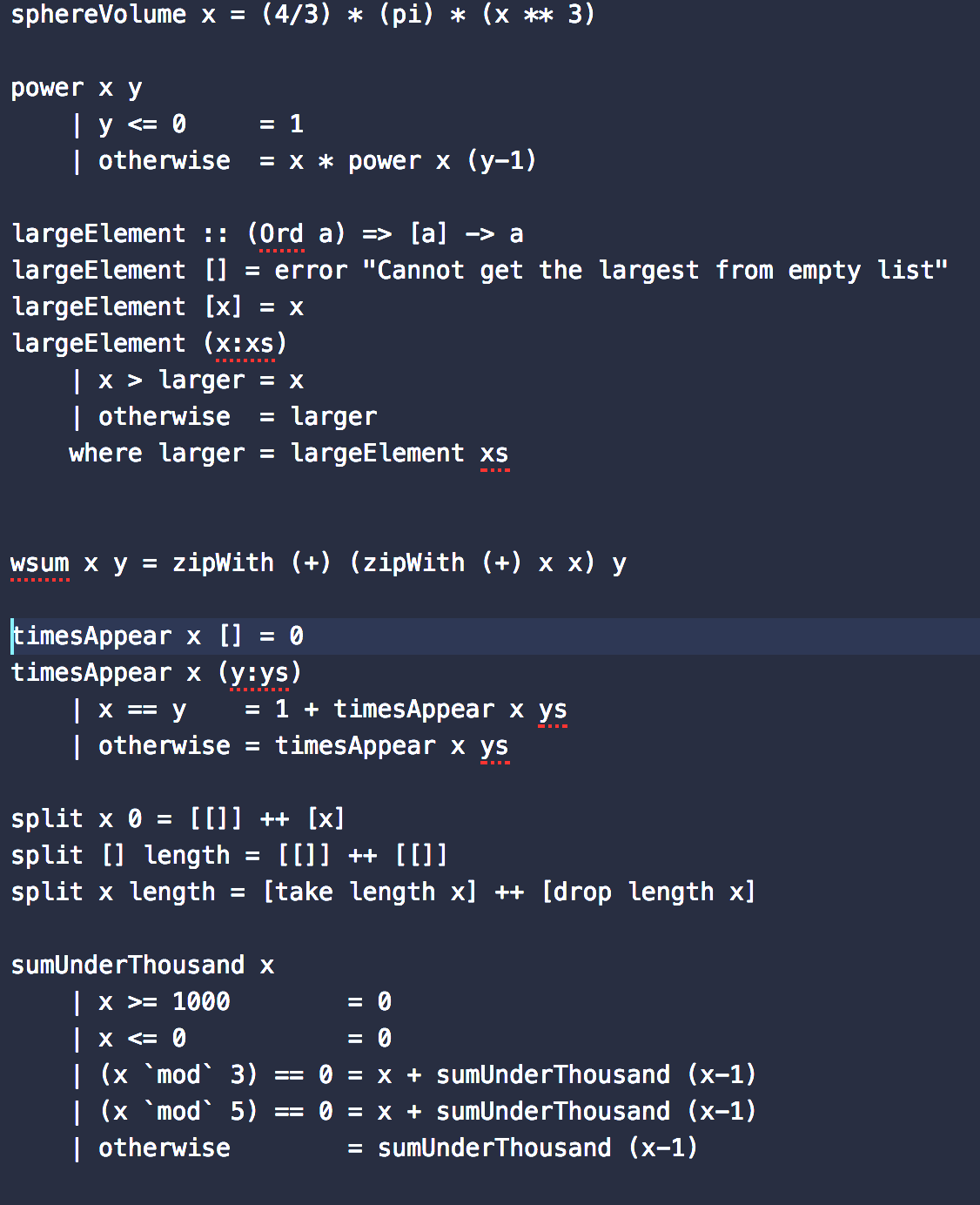
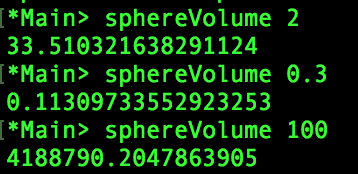
Haskell Assignment 3

**Haskell Code:**

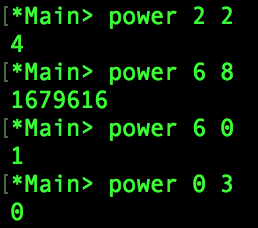


**Evidence of each function running:**

1. **Volume of a Sphere**

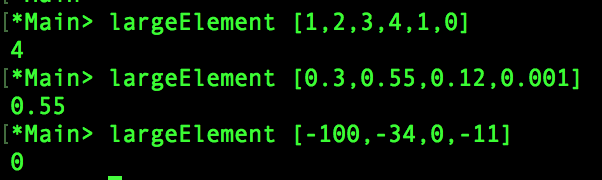
****

1. **Power**

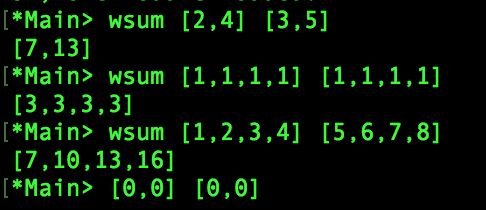
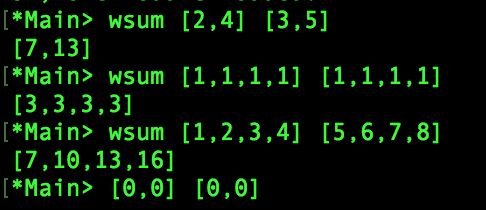
****

Note: Negative and decimals numbers are not accounted for

1. **Largest element in a list of Integers**

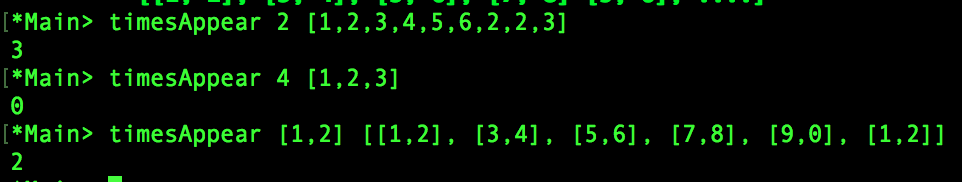
****

1. **wsum**

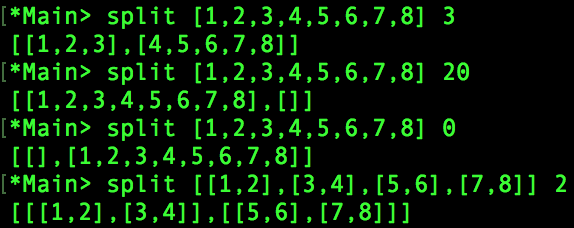
****

****

1. **Number of elements**

****

1. **Split the list into two lists**

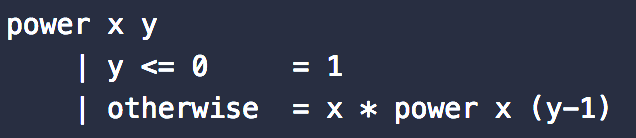
****

1. **Sum of all-natural numbers under one thousand**

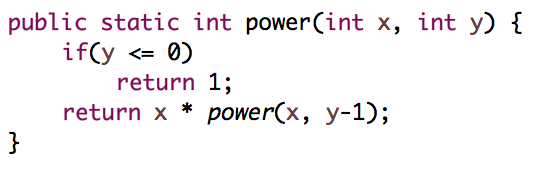
****

**9. Compare one of the above Haskell methods to one in Java**

Haskell:



Java:



In the two pictures above, I am comparing the power function implemented in both Java and Haskell. Both look relatively similar; however, Haskell is more readable in this function as it is easy to see the parameters and the conditionals. Java on the other hand is a little bit harder to read. In Java you have to define the type of every variable, meanwhile it is optional in Haskell. Not defining a type can lead to some unexpected results if the function is passed a variable that is not an integer. Another difference is that Java shows what is being returned via it’s “return” function. Haskell does not have this, making it a little ambiguous to what the function is returning to those who do not know Haskell.