Sheet 3

Exercise 3.1

see `Tasks.hs`.

Exercise 3.2

see `InfLists.hs`.

Exercise 3.3

Could not solve this exercise, tried but failed to understand, even with research and the tips in the discord server.

Exercise 3.4

see `Time.hs`.

Exercise 3.5

see `Temperature.hs`.

Exercise 3.6

a) Currying is about turning a function that has multiple arguments into a series of functions that take a single argument until there is only a return value left. Yes, it is possible to create a curried version of any function. In fact, when executing a function, Haskell creates a curried version of it by default. The benefits of currying are: simplicity, modularity,, and re-usability. Currying makes it easier to partially apply functions and reuse logic of functions.

b) foo = \x -> (\y -> x * y) This function takes an x value, returns a function that takes a y value. That inner function which takes the y value then returns the result of x * y.

c)
```
foo :: Bool -> Int -> Int -> Int
foo = \x -> \y -> \z ->
    if x then y + z
    else y * z
```

Exercise 3.7

see `steganography.hs`.

c) It reminds us of lazy evaluation. The way that the professor only submits the grades when asked is like in lazy evaluation when the value is only computed when needed or asked for.

Exercise 3.8

see `mandelbrot.hs`.

AI Tools

ChatGPT (Open AI) was used throughout this exercise to gain a better understanding of the concepts of the programming languages. No lines of code were copy pasted.