**1. Virtual CPU and tail recursion**

* What is an instruction?
  + A set of binary numbers that are giving to the cpu
  + Some of the instructions can be things like push and pop
* What is a CPU?
  + It processes instructions
  + It has four primary functions
    - Fetch
      * It needs to get the program (set of instructions)
    - Decode
      * Java compiler -> assembly language -> binary code
    - Execute
      * Based on the instructions the cpu will do it
    - Writeback
      * The cpu writes “something” back to the memory
* What is tail recursion?
  + Tail recursion is when the recursive call is at the end of the function
  + Tail recursion updates the stack, instead of increasing it with new data all the time
* When can tail recursion be applied?
  + When the recursive call is in the end of the function
* Why is tail recursion necessary in FP?
  + Its possible to run out of stack space, therefor FP needs tail recursion, since it updates the stack instead of increasing it all the time

**2. Map and flatmap**

* What is the difference between an array and a linked list?
  + In an array you can access whatever index you want at any time, in a linked list you have to go through every element until you get to the one you wanted
* What is recursion?
  + When you call the same function
* What is a higher-order function?
  + It takes other functions as arguments or returns one as a result
* What does mapping mean?
  + You go through an array and apply a function to every element, and returns a new array with the new elements
* What does flattening a list mean?
  + If you have a list which is having two lists inside it, flattening will return one big list with all the elements inside.

**3. Elm**

* What are some of the differences between OOP and FP?
  + No side effects in FP (Runtime handles side effect)
  + FP fails in compile instead of runtime
* Why do we need the Elm runtime?
  + It handles side effects for us
  + It notifies us when a user pressed a button on a website, and gives us data to handle in our update functions
* What are some of the benefits of using Elm?
  + Errors are notified in compile time
  + You are “forced” to handles possible erros (Result type)
  + (Type strong)
  + (Immutable data)
* What is the Maybe type in Elm? When is it used?
  + Just or Nothing
  + Type Maybe a (Generic) = Just a eller nothing
* What is the Result type in Elm? When is it used?
  + Ok or Err
* Can you reverse a string in Elm?
  + reverse String
* Can you write a call to a HTTP endpoint?
  + Http.send “data” (Http.get “url” “decoder”) (ElmHttpExercise)

**4. Haskell**

* What is a side-effect?
  + A side effect is when you do something else than manipulation with the data you get as input
* How do Haskell handle side-effects? What does 'IO ()' mean?
  + ‘IO ()’ is a Monad that updates the user interface and returns something “empty” back to you
* What does it mean that functions are first-class citizens?
  + It means they can be used and passed around as arguments and assigned to variables
  + In java for example classes are a first-class citizen where functions are “hidden” inside of classes.
* What is recursion?
  + When you call the same function
* What are some of the benefits of using Haskell?
  + Errors in compile time