Functional Programming - Exercise 4 - 25 Points

We are at the end of the lecture. One last chance to reflect about the topics we learned so far.

- **Exercise 1: Understanding persistent datastructures (9 Points)** Recently, immutability is hyped in many programming languages. Using immutable variables (e.g. const, final) or just preventing mutable variable assignment is part of a functional programming style. Real power of immutability however becomes really appearent, when using immutable data-structures. In the lecture we looked at an immutable set implementation and talked a lot about advantages of immutable data structures.
    - Given a persistent (immutable) single-linked-list implementation (we implemented a list in the lecture). What is the runtime complexity of the following operations and compare it with `System.Collection.Generic.List<>` in `C#` or `Java.Util.ArrayList`. Use asymptotic runtime in Big O Notation!
        - add element in the front of the list
        - get a copy of the list
        - add an element at the end of the list
- **Wrapup and reflection (16 Points)** FP is everywhere. Hopefully this course also helps you in making concise, robust, efficient and testable code. When looking back and putting the lecture into perspective of your experience answer the following questions:
    - What are the core concepts of functional programming in your opinion (name at least 3, please just one or two sentences each).
    - What are your main takeaways of the lecture (1-3 sentences).

All the best, and till next semester in parallel programming!

- **Sumbission.** Submit your as condensed as possible - e.g. a single file with all the code/markdown. Please don't put it into a zip folder if possible.