Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Prof. Dr.-Ing. Sebastian Schlesinger
Fachbereich 2 - Duales Studium Wirtschaft & Technik

Übungsblatt 2

# Mathematik I - Theoretische Informatik

HWR Berlin, Wintersemester 2025

Prof. Dr.-Ing. Sebastian Schlesinger

**Aufgabe 1 (Schaltjahre)** (4 Punkte)

Schreiben Sie eine Funktion, die ein Jahr erhält und zurückgibt, ob es ein Schaltjahr ist.

Die Regeln für Schaltjahre sind:

Ein Jahr ist ein Schaltjahr, wenn

- es durch 4 teilbar ist, außer
  - wenn es durch 100 teilbar ist (dann ist es keines), es sei denn
    - es ist durch 400 teilbar (dann ist es doch wieder eines).

Zum Beispiel ist 1997 kein Schaltjahr, aber 1996 ist eines. 1900 ist keines, aber 2000 ist eines.

**Aufgabe 2 (Element in Liste finden)** (4 Punkte)

Schreiben Sie ein Programm, das für eine gegebene Liste von Integern prüft, ob eine gegebene Zahl in der Liste enthalten ist.

> Es gibt eine `Prelude`-Funktion `elem`, die genau das tut. Verwenden Sie die nicht, sondern programmieren Sie die Aufgabe selbst.

**Aufgabe 3 (Summe von Elementen einer Liste)** (4 Punkte)

Schreiben Sie ein Programm, das die Summe der Elemente einer gegebenen Liste aus Integern bildet.

> Es gibt eine `Prelude`-Funktion `sum`, die genau das tut. Verwenden Sie die nicht, sondern programmieren Sie die Aufgabe selbst.

**Aufgabe 4 (Take $n$)** (4 Punkte)

Schreiben Sie ein Programm, das aus einer gegebenen Liste von beliebigen Elementen eine gegebene Anzahl entnimmt.

Zum Beispiel soll wenn die Liste `[1,2,3,4,5,6,7,8]` gegeben ist und die Zahl 5, die Liste `[1,2,3,4,5]` zurückgegeben werden.

**Aufgabe 5 (Typen)** (3 Punkte)

Was sind die Typen der folgenden Datenstrukturen?

(i) `(True, 'c')`

(i) `[(True, 'c'),(False,'d')]`

(i) `([True,False],['c','d'])`

**Aufgabe 6 (Potenz)** (2 Punkte)
Definieren Sie eine Funktion `power :: Int -> Int -> Int`, die zwei Integer $x$ und $y$ erhält und $x^y$ zurückgibt.

**Aufgabe 7 (Rekursion auf Listen)** (2 Punkte)
Definieren Sie eine Funktion `ascending :: Ord a => [a] -> Bool`, die eine Liste von Elementen eines beliebigen geordneten Typs erhält und zurückgibt, ob die Elemente in aufsteigender Reihenfolge angeordnet sind.

**Aufgabe 8 (Higher-Order-Funktion)** (2 Punkte)
Definieren Sie eine Funktion `takeWhile' :: (a -> Bool) -> [a] -> [a]`, die eine Liste von Elementen eines beliebigen Typs und eine Prädikatsfunktion erhält und die Elemente der Liste solange zurückgibt, wie das Prädikat erfüllt ist.

**Aufgabe 9 (Binärbäume)** (5 Punkte)
Definieren Sie einen Datentyp für binäre Bäume mit Zahlen als Elemente in Haskell und implementieren Sie Funktionen zum Einfügen und Suchen eines Elementes sowie zum Summieren aller Elemente in einem Baum.

**Aufgabe 10 (Abstrakter Datentyp)** (4 Punkte)
Wir hatten in der Vorlesung den abstrakten Datentypen für natürliche Zahlen definiert. Implementieren Sie in Haskell die Funktionen `natLength :: [a] -> Nat` und `natDrop :: Nat -> [a] -> [a]`.

`natLength` soll die Länge einer Liste als natürliche Zahl zurückgeben.
`natDrop n xs` soll die Liste `xs` um die ersten `n` Elemente kürzen.

**Aufgabe 11 (Funktionale Patterns)** (2 Punkte)
In objektorientierten Sprachen gibt es Design Patterns wie Factory, Singleton, Decorator, Observer, Strategy, etc. Recherchieren Sie im Internet nach funktionalen Pattern und stellen Sie eines davon kurz vor.

**Aufgabe 12 (Closures)** (3 Punkte)
Ein anspruchsvolleres funktionales Pattern sind Closures. Recherchieren Sie das und erklären Sie es.

**Aufgabe 13 (Ordnung in Python)** (4 Punkte)
In objektorientierten Sprachen, z.B. Python, kann man Methoden überladen, um Gleichheit und Vergleiche zu definieren. Damit lassen sich auch Vergleiche wie $<$, $>$, $\leq$, $\geq$, == verwenden und die Objekte in Datenstrukturen, die eine Ordnung benötigen oder bevorzugen sortiert einfügen. Recherchieren Sie, wie das in Python funktioniert und machen Sie ein Beispiel, z.B. eine Klasse für Studierende mit der lexikographischen Ordnung nach Nachname, Vorname, Matrikelnummer.