Zeichenketten:

1.)
Schreiben sie eine Funktion die testet ob eine Zeichenkette korrekt geklammert ist.
Hierbei bedeutet korrekt geklammert, dass in jedem Präfix der Zeichenkette mindestens so viele öffnende wie schließende Klammern vorkommen.
Bsp:
`korrekt ()()() = True`
`korrekt ((()))()( = False`

2.)
Implementieren Sie die Listenfunktion `countSymbols :: String -> [(Char,Int)]`
mit der für alle in der Liste auftretenden Symbole deren Vielfachheit bestimmt wird,
z.B. soll für die Eingabe `"abbacxxxax"` die Liste `[('a',3),('b',2),('c',1),('x',4)]`
ausgegeben werden.

3)
`onlySingles` soll nur die Zeichen in den Ausgabestring setzen, die im Eingabestring
genau einmal vorkommen, also z.B. für die Eingabe `"abbacxxxax"` den String "c" ausgeben.

Funktionen höherer Ordnung:
4.)
Definieren Sie eine Funktion `verdoppelAlles`, die angewendet auf eine Liste von Zahlen, alle Elemente verdoppelt

a) mit Hilfe von map
b) mit Hilfe von foldl
c) mit Hilfe von unfold

Bäume:
5.)
gegeben sei folgende Definition von Bäumen:
```
data Tree = Leaf Int | Node Int Tree Tree
```

a) implementieren sie eine Funktion `countEven`, die zählt wie viele ungerade Zahlen im Baum gespeichert sind

b) implementieren sie die `map` Funktion für diese Bäume

c) `countEvenPaths` soll die Anzahl der Wege von der Wurzel bis zu einem Blatt ausgeben, auf denen die Summe aller Markierungen (einschl. Wurzel und Blatt) gerade ist.

Typklassen:
6)
gegeben sei folgender Datentyp:
```
data Eq a => Menge  a = M [a]
```

a) Machen sie Menge zu einer Instanz der Klasse Eq. Hierbei sollen zwei Mengen a,b als gleich gelten, falls jedes Element aus a auch in b vorkommt.

b) implementieren sie die Methode `einfuegen ::Eq a => a → Menge a → Menge a`, die ein Element in die Menge einfügt, falls dieses noch nicht vorhanden ist.

Listcomprehension:

Lösen sie folgende Aufgaben mithilfe von Listcomprehension:

7) Schreiben sie eine Funktion die:

a) die Liste aller Zahlen zwischen 0 und 1000, in deren Binärdarstellung die vorletzte Stelle 1 ist erzeugt.

b) die Liste aller Paare (a,b) von ganzen Zahlen zwischen 1 und 50, deren Differenz einen Betrag < 5 hat erzeugt.

c) die eine Liste `x::[String]` bekommt und die Liste aller Palindrome aus x erzeugtdie in x vorhanden sind erzeugt.

Codierung:

8.)

Welcher der folgenden Codierung sind eindeutig decodierbar ? Begründungen dürfen niemals fehlen !

$C_1 = \{01, 10, 0100, 1101\}$
$C_2 = \{00, 0010, 011, 100, 1111\}$
$C_3 = \{010, 101, 0101\}$

9.)

Für welche der folgenden Codewortlängen lässt sich ein Präfixcode erzeugen ? Warum ?

a) n1 = n2 = 2, n3 = n4 = 3
b) n1 = n2 = n3 = 2, n4 = n5 = 6
c) n1 = n2 = n3 = n4 = n5 = n6 = 4