3. Übung zu ALGORITHMEN UND PROGRAMMIERUNG I

Abgabe bis Donnerstag, den 9. November

1. **Aufgabe** *4 Punkte*
   Ein frei beweglicher Körper mit der Masse m werde aus der Ruhelage eine Zeit t lang mit einer auf ihn einwirkenden Kraft k bewegt. Gesucht ist ein Algorithmus, der die Strecke s bestimmt, um die der Körper aus seiner ursprünglichen Lage fortbewegt wird. Programmieren Sie diesen Algorithmus in Haskell.
   Hinweise:

   (a) Die auf den Körper einwirkende Kraft erzeugt eine konstante Beschleunigung b mit

   $$b = \frac{k}{m}$$

   (b) Für den in der Zeit t zurückgelegten Weg s bei einer Bewegung mit konstanter Beschleunigung b gilt:

   $$s = \frac{1}{2} * b * t^2$$

2. **Aufgabe** *4 Punkte*
   Entwickeln Sie einen Algorithmus in Haskell, der zu einer Datumsangabe (t,m) (Tag und Monat), die Zahl n ermittelt, für die gilt:

   Der Tag mit dem Datum (t,m) ist der n-te Tag in einem (Nicht-Schalt-)Jahr.

   Beispiel:
   Der 14. Februar ist der 45. Tag eines Jahres.

3. **Aufgabe** *6 Punkte*
   Stellen Sie sich vor, die logischen Grundfunktionen wären in Haskell nicht vorhanden. Der primitive Datentyp **bool** wäre nur durch die Konstanten **True** und **False** gegeben.

   (a) Definieren Sie in Haskell die logische Funktion **oder** ,

      i. indem Sie die Möglichkeit ausnutzen, Konstante auf die linke Gleichungsseite zu schreiben,

      ii. indem Sie die in der Vorlesung definierten Funktionen **kod** und **dekod** sowie bekannte arithmetische Funktionen verwenden.

   (b) Beweisen Sie die Korrektheit Ihrer Funktionen.

The Haskell operators are listed below in decreasing order of binding power:
see Section 2.4 for a discussion of associativity and binding power.

	Left associative	Non-associative	Right-associative
9	!,!!,//		.
8			**,^,^^
7	*, /, 'div', 'mod', 'rem','quot'		
6	+,-	:+	
5		\\	:, ++
4		/=, <, <=, ==, >, >=, 'elem', 'notelem'	
3			&&
2			||
1		:=	
0			$

Also defined in this text are the operators

	Left associative	Non-associative	Right-associative
9	> . >		
5			> * >
1		>>=	

The restrictions on names of operators, which are formed using the characters

!, #, $, %, &, *, +, ., /, <, =, >, ?, @, \, ^, |, :, -, ~

are that operators must not start with a colon; this character starts an infix constructor. Operators can contain - and ~ , but only as their first character.
Finally, certain combinations of symbols are reserved, and cannot be used:

::, =>, =, @, \, |, ^, − >, < −.

To change the associativity or binding power of an operator, &&& say, we make a declaration like

infixl 7 &&&

which states that &&& has binding power 7, and is a left associative operator. We can also declare operators as non-associative (infix) and right associative (infixr). Omitting the binding power gives a default of 9. These declarations can be used for back-quoted function names, as in

infix 0 'poodle'