3. Aufgabenblatt zu Funktionale Programmierung vom 27.10.2004, fällig: 03.11.2004

Themen: *Funktionen auf ganzen Zahlen, Zeichenreihen und Listen*

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe3.hs` ablegen. Sie sollen also wie auf dem ersten Aufgabenblatt ein "gewöhnliches" Haskell-Script schreiben. Versehen Sie wie auf den bisherigen Aufgabenblättern wieder alle Funktionen, die Sie zur Lösung brauchen, mit ihren Typdeklarationen und kommentieren Sie Ihre Programme aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Im einzelnen sollen Sie folgende Problemstellungen bearbeiten:

1. Wir betrachten wieder das Szenario aus den Teilaufgaben 2 und 3 von Aufgabenblatt 2. Diesmal wollen wir eine Funktion `computeDate` mit gleicher Signatur wie die der Funktion `ageInDays` bestimmen. Wie bei der Funktion `ageInDays` wird das erste Argument als Datum interpretiert, das zweite Argument als positive oder negative Anzahl von Tagen. Angewendet auf zwei Argumente d (für Datum) und t (für Tage) liefert die Funktion das Datum des t Tage in der Zukunft liegenden Tags, wenn t positiv ist, sonst das Datum des t Tage zurückliegenden Tags. Für $t = 1$ liefert `ageInDays` also das "morgige", also das auf d folgende Datum, für $t = -1$ das "gestrige", also das d vorausgehende Datum. Für $t = 0$ liefert `computeDate` das "heutige" Datum, also d selbst. Liegt das berechnete Datum vor dem 1. Januar 1 oder ist das Ausgangsdatum kein gültiges Datum nach dem 1. Januar 1 (einschließlich des 1. Januar 1 selbst) liefert die Funktion `computeDate` das Resultat 0. Denken Sie wie bei der Aufgabe zur Funktion `ageInDays` wieder in genau der gleichen Weise an Schaltjahre.

2. Schreiben Sie ein Prädikat `isSubString`, das als Argument ein Paar von Zeichenreihen erwartet und feststellt, ob die Zeichenreiche der ersten Komponente des Argumentpaars als Teilzeichenreihe in der Zeichenreihe der zweiten Komponente vorkommt. In diesem Fall gibt die Funktion `isSubString` den Wahrheitswert `True` aus, sonst `False`.

3. Schreiben Sie eine Haskell-Rechenvorschrift `allPermSubStrings`, die zwei Zeichenreihen über dem Alphabet `a,b,c,...z,A,B,C,...,Z` als

Argumente erwartet (anders als in der vorigen Aufgabe kein Paar) und eine Liste von Zeichenreihen als Ergebnis ausgibt. Die Ergebnisliste enthält dabei lexikographisch geordnet und ohne Wiederholungen all diejenigen Permutationen der ersten Zeichenreihe, die als Teilzeichenreihe im zweiten Argument vorkommen. Großbuchstaben werden lexikographisch nach Kleinbuchstaben einsortiert, d.h. "a" steht lexikographisch vor "A". Bei der Implementierung können Sie davon ausgehen, dass die Funktion nicht mit Zeichenreihen über einem anderen Alphabet als oben angegeben aufgerufen wird.

4. In Haskell gibt es den vordefinierten (polymorphen) Operator `!!` mit Signatur `!! :: [a] -> Int -> a` für den direkten Zugriff auf die Elemente in einer Liste. So liefert der Aufruf `ls!!n` grundsätzlich das n-te Element der Liste `ls`. Dabei werden die Elemente einer Liste beginnend mit 0 von links nach rechts durchgezählt. Ein Beispiel: Der Aufruf `[1,2,3,4,5]!!0` liefert das Element `1`, der Aufruf `[1,2,3,4,5]!!1` das Element `2` und der Aufruf `[1,2,3,4,5]!!4` das Element `5`. Ist das Argument `n`, auch *Index* genannt, negativ oder zu groß, also kein entsprechendes Element mehr in der Liste vorhanden, so endet der entsprechende Aufruf mit einem Laufzeitfehler. Schreiben Sie jetzt einen Haskell-Operator `!!!` mit gleicher Signatur wie `!!` und folgender Funktionalität. Angewendet auf eine nichtleere Liste `ls` und einen nichtnegativen Index `n` liefert `!!!` das zyklisch gezählte an der Stelle n stehende Listenelement zurück. D.h.: ist n nicht kleiner als 0 und nicht größer als die um 1 verringerte Anzahl der Elemente der Liste, so verhält sich die Operation `!!!` wie die Operation `!!`. Ist n größer als die um 1 verringerte Anzahl der Elemente der Liste, so wird (wieder mit dem ersten Element beginnend) zyklisch weitergezählt in der Liste bis hin zu n und das dort gefundene Element ausgegeben. In analoger Weise geschieht dies für negative Indizes. Dabei werden die Elemente der Liste allerdings von rechts nach links beginnend mit -1 durchgezählt. Ein Beispiel: Der Aufruf `[1,2,3,4,5]!!!2` liefert das Element `3`, der Aufruf `[1,2,3,4,5]!!!5` das Element `1` und der Aufruf `[1,2,3,4,5]!!!6` das Element `2`. Analog liefert der Aufruf `[1,2,3,4,5]!!!(-1)` das Element `5`, der Aufruf `[1,2,3,4,5]!!!(-5)` das Element `1` und der Aufruf `[1,2,3,4,5]!!!(-6)` das Element `5`. Angewendet auf eine leere Liste, wird kein bestimmtes Verhalten von `!!!` gefordert.

Benachrichtung über erreichte Punktezahlen

Am 20.10.2004 sind Benachrichtigungen über die von Ihnen bei der Erstabgabe zu Aufgabenblatt 1 erreichten Punktzahlen an Ihren Mail-Account `fpxxx@b1` versandt worden. Im Lauf des 27.10.2004 werden diese Mitteilungen entsprechend für die Nachabgabe zu Aufgabenblatt 1 und für die Erstabgabe zu Aufgabenblatt 2 verschickt. Sollten Sie diese automatisch versandten Benachrichtigungen nicht erhalten bzw. nicht erhalten haben, kontrollieren Sie bitte die Daten, die Sie beim ersten Einloggen auf Ihrem `fpxxx@b1`-account angegeben haben, insbesondere eine eventuell angegebene Adresse, an die Sie sich an Ihren `fpxxx@b1`-account geschickte Nachrichten weiterleiten lassen und überprüfen Sie diese Adresse auf ihre Richtigkeit. Können Sie auch dann keinen Grund feststellen, warum Sie diese Benachrichtigungen nicht erhalten haben, wenden Sie sich bitte an einen Tutor um Hilfe.