9. Übung **Informatik A** **WS 13/14**

Klaus Kriegel **Abgabe: 06.01.2014, 12:00 Uhr**

Zum Weihnachtsfest ist dieser reguläre Übungszettel etwas kleiner als üblich. Alle Aufgaben werden gewertet. Darüber hinaus wird es (am 16.12.) eine Zusatzübung geben, deren Abgabe freiwillig ist. Alle darauf erreichten Punkte werden als Bonuspunkte eingetragen. Die Zusatzübung kann bis zum 08.01.2014 um 10:00 Uhr abgegeben werden.

Aufgabe 1: **Listenrekursion** (2 Punkte)

Entwerfen Sie analog zur primitiv rekursiven Implementierung der Fibonacci-Zahlen mit Hilfe von Paaren (in der VL) eine primitiv rekursive Berechnung dieser Zahlen mit Hilfe von Listen. Neben der Hauptfunktion `fastFib :: Int -> Int` sollte Ihre Implementierung eine Funktion `fibList :: Int -> [Int]` verwenden, welche bei Eingabe n die Liste Fibonacci-Zahlen von f_n bis f_0 enthält.

Aufgabe 2: **Selection Sort** (4 Punkte)

Beim Sortieren durch Einfügen wird in jedem Schritt ein beliebiges Element der Eingabefolge - man nimmt jeweils das nächste - an die richtige Stelle in der Ausgabefolge eingefügt (kein Aufwand für die Auswahl, aber für das Einfügen). Wir wollen die duale Idee implementieren, nämlich in jedem Schritt jeweils das maximale Element aus der Eingabefolge streichen (aufwändig) und als dann als Anfangselement in die aktuelle Ausgabefolge einsetzen (einfach). Der maximale Wert in einer `[Int]`-Liste kann mit der vordefinierten Funktion `maximum` bestimmt werden. Ihre Implementierung sollte mindestens die folgenden Funktionen realisieren:
`delMax :: [Int] -> [Int]` soll aus der Eingabeliste ein maximales Element (wenn es mehrfach vorkommt, dann das zuerst gefundene) aus der Liste entfernen.
`selStep :: ([Int],[Int]) -> ([Int],[Int])` soll einen Schritt des Verfahrens ausführen, wobei an der ersten Stelle im Paar der aktuelle Rest der Eingabeliste und an der zweiten Stelle die aktuelle Ausgabeliste stehen soll.
`selSort :: [Int] -> [Int]` soll das Sortierverfahren implementieren. Wahrscheinlich ist es günstig, eine weitere Hilfsfunktion zum rekusiven Aufruf von `selStep` zu definieren.

Aufgabe 3: **Listenfunktionen** (3 Punkte)

a) Die Funktion `maxList :: ([Int],[Int]) -> [Int]` soll für ein Paar von Listen der Längen l_1 und l_2 die Liste der Maxima an den Stellen von 0 bis $min(l_1, l_2) - 1$ bestimmen und dahinter die restlichen Elemente der längeren Liste anhängen. So soll der Aufruf `maxList ([2,3,5],[4,4,4,4,1])` die Liste `[4,4,5,4,1]` erzeugen.
b) Die Funktion `maxListOfLists :: [[Int]],[[Int]] -> [[Int]]` soll das analoge für Listen von Listen leisten.
So soll `maxListOfLists [[1,4],[],[3,1,4]] [[2,3,1],[6,7],[5],[],[4,4]` die Liste `[[2,4,1],[6,7],[5,1,4],[],[4,4]` erzeugen. Versuchen Sie, das Problem mit Hilfe der Funktionen `map` und `zip` auf den Fall a) zurückzuführen.