

Lösung Übungsblatt Nr. 8 / ALP 1 zur Abgabe 22.01.2004**Aufgabe 1:****Programmcode**

```

interleave :: Int -> [Int] -> [[Int]]
interleave x [] = [[x]]
interleave x (y:ys) = [x:y:ys] ++ (map (y:) (interleave x ys))

perms :: [Int] -> [[Int]]
perms [] = [[]]
perms (x:xs) = [zs | ys <- perms xs, zs <- (interleave x ys)]

is_opt :: [Int] -> Int -> Int -> Int -> Bool
is_opt [] m n max = False
is_opt (x:xs) m n max | ((x>max) && (m>1)) = is_opt xs (m-1) n x
                      | ((x>max) && (m<=1)) = (x==n)
                      | otherwise = is_opt xs (m-1) n max
is_opt (x:_) m n max = (x==n)

geschenk :: [[Int]] -> Int -> Int -> Int -> Int
geschenk [[]] m n hit = hit
geschenk [] m n hit = hit
geschenk (x:xs) m n hit | ((is_opt x m n 0)==True) = geschenk xs m n hit+1
                        | otherwise = geschenk xs m n hit

testgeschenk :: Int -> Int -> Float
testgeschenk m n = (fromInt (geschenk permus m n 0) / fromInt (length permus)) * 100
where permus = perms [1..n]

```

Testlauf

```

Main> testgeschenk 5 7
35.2381
Main> testgeschenk 4 7
40.7143
Main> testgeschenk 3 7
41.4286
Main> testgeschenk 2 7
35.0

```

Beschreibung

Testgeschenk erhält 2 Variablen (m und n , wie in der Aufgabenstellung gefordert) und gibt dann den Prozentsatz der der Testläufe mit optimalem Ausgang als `Floating Point` aus. Kommentierung zur Prozentrechnung spar ich mir mal. Das Programm `geschenk` bekommt eine Liste mit allen Permutationen der Liste der Geschenke-Wertverteilung, m und n und den maximalen Wert der bereits gesehenen Geschenke (max). (im Startzustand = 0) Dann prüft es für jede Permutation in der Liste, ob das optimale Geschenk gefunden worden ist oder nicht. (`True/False`) Die Prüfung dessen übernimmt das Programm `is_opt` wie folgt. Wenn aktuelles Geschenk wertvoller als alle bereits gesehenen UND zu verwerfende Geschenke noch nicht alle verworfen, Maximalwert des Geschenks merken und trotzdem verwerfen. Wenn aktuelles Geschenk wertvoller als alle bereits gesehenen UND zu verwerfende Geschenke alle verworfen, nehmen und angucken (`True/False`) Wenn die Liste nur noch ein Element hat und dieses gleich dem maximal zu erreichenden Wertes hat -> `True` ansonsten `False`.

Um die Liste der Elemente einer Permutation durcharbeiten guckt sich das programm immer nur den Kopf der Liste an und ruft sich danach mit dem Rest nochmal auf, es sei denn es wurde vorher das "gedachte optimale Geschenk" gefunden.

Die Permutationen wurden in der Vorlesung vorgegeben und werden daher nicht kommentiert :)

Lösung Übungsblatt Nr. 8 / ALP 1 zur Abgabe 22.01.2004**Aufgabe 2:****Programmcode**

```
perfect_shuffle :: [a] -> [a]
perfect_shuffle (x:xs) = shuffle (take ((length (x:xs))`div`2) (x:xs)) (drop ((length (x:xs))`div`2)
(x:xs))

shuffle :: [a] -> [a] -> [a]
shuffle [] [] = []
shuffle [] [x] = [x]
shuffle [x] [] = [x]
shuffle (a1:b1) (a2:b2) = [a1] ++ [a2] ++ (shuffle b1 b2)
```

Testlauf

```
Main> perfect_shuffle [1..10]
[1,6,2,7,3,8,4,9,5,10]
Main> perfect_shuffle [20..35]
[20,28,21,29,22,30,23,31,24,32,25,33,26,34,27,35]
Main> perfect_shuffle [0..15]
[0,8,1,9,2,10,3,11,4,12,5,13,6,14,7,15]
Main>
```

Beschreibung

Das Programm bekommt eine Liste von Integern als Eingabewert und teilt sie in der Mitte. Dann fügt es die beiden halben Listen im Reißverschlussverfahren wieder zusammen zu einer Liste.

Aufgabe 3

leider ungelöst.