

Softwaretechnik und Programmiersprachen I - Sommersemester 2011

Übungsblatt 5

Besprechung der Aufgaben in 25.12.02.55 am Mittwoch 14:30-16:00 Uhr oder Freitag 10:30-12:00 Uhr
Bei Fragen wenden Sie sich bitte an John Witulski: John.Witulski@uni-duesseldorf.de.

Aufgabe 5.1 (Umgang mit Listen)

Implementieren Sie folgende Prädikate zum Umgang mit Listen, ohne dabei auf bestehende Bibliotheksprädikate zurückzugreifen:

a. $app(L1, L2, R)$: R ist die Liste, die entsteht, wenn die Listen $L1$ und $L2$ geheftet werden.

b. $duplicate(L, R)$: R ist die Liste, die durch Duplizieren jedes Elements der Liste L entsteht.

?-duplicate([1,2,3],R).

R = [1,1,2,2,3,3]

c. $range(N, M, L)$: L ist die Liste, die die Elemente im Intervall $[N, M]$ enthält, wobei N und M natürliche Zahlen sind.

?-range(1,10,L).

L=[1,2,3,4,5,6,7,8,9,10]

d. $zip(A, B, Z)$: A , B und Z sind gleich lange Listen. X ist das i . Element von A und Y das i . Element von B , genau dann wenn $pair(X, Y)$ das i . Element von Z ist. Z.B.

?- zip([a,b,c],[1,2,3],Z).

Z = [pair(a,1),pair(b,2),pair(c,3)]

e. $rev(L, R)$: R ist die Liste, die entsteht, wenn die Elemente von L in ihrer Reihenfolge umgedreht werden.

f. $element_at(E, L, P)$: P ist die Position, an der sich das Element E in der Liste L befindet.

?-element_at(E,[11,12,13,14,15],3).

E=13

g. $len(L, N)$: N die Länge der Liste L ist.

h. $palindrom(P)$: Die Liste P ist ein Palindrom.

Aufgabe 5.2 (Listen)

Implementieren Sie folgende Prädikate:

- a. *islist(L)*: *islist/1* überprüft, ob das angegebene Argument L eine Liste ist.
- b. *flatt(L, R)*: L ist eine Liste, die als Elemente weitere Listen enthalten kann, und R ist die resultierende Liste, die durch Ersetzen jeder Liste mit ihren Elementen entsteht.

```
?-flatt([a,[b,[c,d],e],f],R).  
R=[a,b,c,d,e,f]
```

Hinweis: Sie können die Prädikaten *islist/1* und *app/3* (siehe Aufgabe 5.1) benutzen.

- c. *compress(L, R)*: *compress/2* ersetzt aufeinanderfolgende gleiche Elemente in der Liste L durch einzelne Kopien von diesen. R ist die resultierende Liste.

```
?-compress([a,a,a,a,b,c,c,a,a,d,e,e,e,e],R).  
R = [a,b,c,a,d,e]
```

- d. *split(L, N, L1, L2)*: N ist die Position in der Liste L, an der die Liste L in zwei Teillisten L1 und L2 aufgeteilt wird, wobei das N-te Element in L das letzte in L1 ist.

```
?-split([1,2,3,4,5,6,7,8],3,L1,L2).  
L1 = [1,2,3]  
L2 = [4,5,6,7,8]
```