List Processing Manual

Author: Joshua Harkness

Introduction

The lp.pro file covers some basic list processing concepts in prolog. The purpose for the creation of this file was to familiarize future prolog writers with the list processing power of prolog.

This document will provide insight on how each rule works and will instruct the reader on how to use the program.

Commands with Examples

The commands available in the program are as follows:

writelist([]) – given a list, prolog will print out the list it was given.

```
?- writelist([1,2,3]).
1
2
3
true.
```

member(N,[]) – checks if the given element N is in the given list. This is a Prolog primitive.

```
?- member(2,[1,2,3]).
true .
```

length([]) – returns the length of the given list. This is a Prolog primitive.

```
?- length([1,2,3],L).
L = 3.
```

item(N, [],E) – returns the Nth item of the list. The Prolog primitive equivalent is nth0.

```
?- item(2,[1,2,3],Element).
Element = 3 .
```

append([],[],L) – concatenates the first list with the second, and returns the appended list. This is a Prolog primitive.

```
?- append([1,2,3],[4,5,6],NewList).

NewList = [1, 2, 3, 4, 5, 6].
```

last([],L) – returns the last value in the given list. This is a Prolog primitive.

```
?- last([1,2,3],Last).

Last = 3 .
```

remove(N,[],List) – removes element N from the list, then returns the list without that element. The Prolog primitive equivalent is delete.

```
?- remove(2,[1,2,3],NewList).

NewList = [1, 3] .
```

replace(N,X,[],L) – replaces the value N in the given list with the value X, and returns this new list. The Prolog primitive equivalent is select.

```
?- replace(1,one,[1,2,3],NewList).

NewList = [1, one, 3] .
```

makelist(N,X,L) – makes a list of N length with each value in the list being value X and returns this list.

```
?- makelist(3,7,NewList).

NewList = [7, 7, 7] .
```

reverse([],L) – takes a list and reverses the order of the elements then returns the new list. This is a Prolog primitive.

```
?- reverse([3,2,1],NewList).

NewList = [1, 2, 3] .
```

lastput(N,[],L) – places the value N last in the given list, and returns this new list.

?- lastput(4,[1,2,3],NewList).

NewList = [1, 2, 3, 4] .

pick([],V) – selects a random element in the list and returns it.

?- pick([1,2,3],Item).

Item = 2 .

take([],V,L) – removes a random value from the given list and returns both the removed value and the new list.

?- take([1,2,3,4,5,6],Item,NewList).

Item = 2,

NewList = [1, 3, 4, 5, 6] .

iota(N,L) – creates a list from 1 through N and returns this list.

?- iota(3, List).

List = [1, 2, 3] .

sum([],Sum) – adds each value of the list together and returns the sum. The Prolog primitive equivalent is sumlist.

?- sum([1,2,3],Sum).

Sum = 6.

min([],M) – returns the smallest value in the given list. The Prolog primitive equivalent is min_list.

?- min([4,3,5,8,1,2],Min).

Min = 1 .

max([],M) – returns the largest value in a given list. The Prolog primitive equivalent is max_list.

```
?- max([4,3,5,8,1,2],Max).

Max = 8 .
```

sort_inc([],L) – orders the values in a given list in increasing order and returns this new ordered list.

```
?- sort_inc([4,3,5,8,1,2],List).

List = [1, 2, 3, 4, 5, 8] .
```

sort_dec([],L) – orders the values in a given list in decreasing order and returns this new ordered list.

```
?- sort_dec([4,3,5,8,1,2],List).

List = [8, 5, 4, 3, 2, 1] .
```

alist([],[],AL) – generates a new list where the values for the first list are the keys to the same indexed value of the second list. The key/value pairs are each placed into a pair(). The new list consists of pair(key, value).

```
?- alist([2,3,4],[two,three,four],Alist).

Alist = [pair(2, two), pair(3, three), pair(4, four)].
```

assoc([], K,V) – given an associative array, returns the key and value for each key value pair in the list.

```
44 ?- assoc([pair(2,two),pair(3,three),pair(4,four)],K,V).

K = 2,

V = two ;

K = 3,

V = three ;

K = 4,
V = four ;
false.
```

flatten([A,[],C], L) – takes a list of elements, where one or more of the elements is a list, and removes the elements from the inner list(s) and places them into the outer most list. The Prolog primitive equivalent is flatList.

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

Demo

```
1 ?- consult('lp.pro').
ERROR: c:/users/joshu/documents/csc366/assignments/assignment12/lp.pro:12:
    No permission to modify static procedure `length/2'
    Defined at c:/program files/swipl/boot/init.pl:2865
ERROR: c:/users/joshu/documents/csc366/assignments/assignment12/lp.pro:13:
    No permission to modify static procedure `length/2'
    Defined at c:/program files/swipl/boot/init.pl:2865
% lp.pro compiled 0.00 sec, 48 clauses
true.

2 ?- writelist([1,2,3]).
1
2
3
true.

3 ?- member(2,[1,2,3]).
true .
```

```
4 ?- member(4,[1,2,3]).

false.


5 ?- length([1,2,3],L).

L = 3.


6 ?- item(0,[1,2,3],Element).

Element = 1 .


7 ?- item(2,[1,2,3],Element).

Element = 3 .


8 ?- append([1,2,3],[4,5,6],NewList).

NewList = [1, 2, 3, 4, 5, 6].


9 ?- append([],[1,2,3],NewList).

NewList = [1, 2, 3].


10 ?- last([1,2,3],Last).

Last = 3 .


11 ?- last([1],Last).

Last = 1 .


12 ?- remove(2,[1,2,3],NewList).

NewList = [1, 3] .
```

```
13 ?- remove(2,[],NewList).

NewList = [] .


14 ?- remove(4,[1,2,3],NewList).

NewList = [1, 2, 3] .


15 ?- replace(1,one,[1,2,3],NewList).

NewList = [1, one, 3] .


16 ?- replace(0,one,[1,2,3],NewList).

NewList = [one, 2, 3] .


17 ?- makelist(3,7,NewList).

NewList = [7, 7, 7] .


18 ?- makelist(0,nothing,NewList).

NewList = [] .


19 ?- reverse([3,2,1],NewList).

NewList = [1, 2, 3] .


20 ?- reverse([3],NewList).

NewList = [3] .


21 ?- lastput(2,[],NewList).

NewList = [2] .
```

```
22 ?- lastput(4,[1,2,3],NewList).

NewList = [1, 2, 3, 4] .


23 ?- pick([1,2,3],Item).

Item = 3 .


24 ?- pick([1,2,3],Item).

Item = 3 .


25 ?- pick([1,2,3],Item).

Item = 2 .


26 ?- pick([1,2,3],Item).

Item = 2 .


27 ?- pick([1,2,3,4,5,6],Item).

Item = 5 .


28 ?- pick([1,2,3,4,5,6],Item).

Item = 5 .


29 ?- pick([1,2,3,4,5,6],Item).

Item = 3 .


30 ?- pick([1,2,3,4,5,6],Item).

Item = 3 .
```

```
31 ?- pick([1,2,3,4,5,6],Item).

Item = 5 .


32 ?- take([1,2,3,4,5,6],5,NewList).

false.


33 ?- take([1,2,3,4,5,6],Item,NewList).

Item = 2,

NewList = [1, 3, 4, 5, 6] .


34 ?- take([1,2,3,4,5,6],Item,NewList).

Item = 3,

NewList = [1, 2, 4, 5, 6] .


35 ?- iota(5,List).

List = [1, 2, 3, 4, 5] .


36 ?- iota(3, List).

List = [1, 2, 3] .


37 ?- sum([1,2,3],Sum).

Sum = 6.


38 ?- sum([1,1,1],Sum).

Sum = 3.
```

```
39 ?- min([4,3,5,8,1,2],Min).

Min = 1 .


40 ?- max([4,3,5,8,1,2],Max).

Max = 8 .


41 ?- sort_inc([4,3,5,8,1,2],List).

List = [1, 2, 3, 4, 5, 8] .


42 ?- sort_dec([4,3,5,8,1,2],List).

List = [8, 5, 4, 3, 2, 1] .


43 ?- alist([2,3,4],[two,three,four],Alist).

Alist = [pair(2, two), pair(3, three), pair(4, four)].


44 ?- assoc([pair(2,two),pair(3,three),pair(4,four)],K,V).

K = 2,

V = two ;

K = 3,

V = three ;

K = 4,

V = four ;

false.


45 ?- flatten([a, [b,c], [[d],[],[e]]], R).

R = [a, b, c, d, e] .
```