Fred Morrison
CSC-366

List Processing Users' Guide

Table of Contents:

1.) writelist(list)

a.) Parameters

i.) list → a prolog list

b.) Overview

i.) Prints out each element in the given list, starting at the first index, with a new line in between.

c.) Example

```
?- writelist([minnesota, vikings, green_bay, packers]).
minnesota
vikings
green_bay
packers
true.
```

2.) member(potential, list)

d.) Parameters
   - i.) potential → the element to search for in the list
   - ii.) list → a prolog list

e.) Overview
   - i.) Returns true or false depending on if the potential element exists in the list.

f.) Example

?- member(5, [1,2,3,4]).
false.

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

3.) size(list, Size)

g.) Parameters
   - i.) list → the list to find the size of
   - ii.) Size → the Variable to bind the result to

h.) Overview
   - i.) Calculates the number of elements in the given list, and binds the result to the variable Size.

i.) Example

?- size([0,1,2,3,4,5,6,7,8],S).
S = 9.

4.) item(index, list, Item)

j.) Parameters
   - i.) index → the index of the list to grab the item from
   - ii.) list → the list to find the item in
   - iii.) Item → the Variable to bind the result to

k.) Overview
   - i.) Given a list and an index (integer), find the element of the list at the given index and bind it to the Item variable.

l.) Example

?- item(3, [vikings,packers,lions,bears], I).
I = bears .

5.) append(list_one, list_two, NewList)

m.) Parameters
   - i.) list_one → the original list
   - ii.) list_two → the list which is being appended
   - iii.) NewList → the Variable to bind the result to

n.) Overview

        i.) Given 2 lists, append the second one to the end of the first, and bind the result to a variable.

    o.) Example

        ?- append([happy, grim], [angry, excited], R).
        R = [happy, grim, angry, excited].

        ?- append([happy],[grim],[angry],[excited],R).
        R = [happy, grim, angry, excited].

6.) last(list, Last)

    p.) Parameters

        i.) list→ the list to find the last element of
        ii.) Last → the variable in which the last element will be bound to

    q.) Overview

        i.) Given a list, find the last element in it and bind the value to the Last variable.

    r.) Example

        ?- last([first, second, third, fourth], L).
        L = fourth .

7.) remove(element, list, ResultingList)

    s.) Parameters

        i.) element→ the element to remove from the list
        ii.) list → the list to remove the element from
        iii.) ResultingList → the variable to bind the resulting list to

    t.) Overview

        i.) Given a list, and an element, attempt to remove the element from the list and bind the remaining list to the ResultingList variable.

    u.) Example

        ?- remove(love, [love, hate, relationship], R).
        R = [hate, relationship] .

8.) replace(index, element, list, ResultingList)

    v.) Parameters

        i.) index→ the index of the list to replace
        ii.) element → the new element to replace the old one with
        iii.) list → the list which contains the element to replace
        iv.) ResultingList → the resulting list after the replacement

    w.) Overview

        i.) Given a list, and index, and an element, replace the value at the index in the list with the element, and bind the resulting list to the ResultingList variable.

    x.) Example

?- replace(0,steak,[seafood,pizza,vegetables,fruit],R).
R = [steak, pizza, vegetables, fruit] .

9.) makelist(number, element, ResultingList)

y.) Parameters
- i.) number→ the number of elements to add to the list
- ii.) element → the element to fill the list with
- iii.) list → the created list

z.) Overview
- i.) Given a number (integer) and an element, create a new list filled with the specified number of elements, all set to the value of the provided element.

aa.) Example

?- makelist(10,vikings,L).
L = [vikings, vikings, vikings, vikings, vikings, vikings, vikings, vikings, vikings|...] .

10.) reverse(list, ResultingList)

bb.) Parameters
- i.) list→ the list to reverse
- ii.) ResultingList → the variable to bind the reversed list to

cc.)Overview
- i.) Given a list, reverse it, and bind the resulting list to the ResultingList variable.

dd.) Example

?- reverse([r,a,c,e,c,a,r],R).
R = [r, a, c, e, c, a, r] .

11.) lastput(element, list, ResultingList)

ee.) Parameters
- i.) element → the element to append to the list
- ii.) list→ the list to add the element to
- iii.) ResultingList → the variable to bind the resulting list to

ff.) Overview
- i.) Given a list, and an element, add the element to the end of the list.

gg.) Example

?- lastput(best, [franks, redhot, is, the], L).
L = [franks, redhot, is, the, best] .

12.) pick(list, RandomElement)

hh.) Parameters
- i.) list → the list to pick from
- ii.) RandomElement → the variable to bind the random element to

ii.) Overview

i.) Given a list, pick a random element from it, and bind it to the RandomElement variable.

jj.) Example

?- pick([1,10,100,1000,10000,100000,1000000],Random).
Random = 100000 .

13.) take(list, Element, ResultingList)

kk.) Parameters

i.) list → the list to take from
ii.) Element → the variable to bind the random element to
iii.) ResultingList → the variable to bind the resulting list to

ll.) Overview

i.) Given a list, pick and remove a random element from it. Bind the random element to the Element variable, and bind the remaining list to the ResultingList variable.

mm.) Example

?- take([banana,apple,orange,mango],E,L).
E = mango,
L = [banana, apple, orange] .

14.) iota(number, List)

nn.) Parameters

i.) number → the number of elements to add to the list
ii.) List → the variable to bind the resulting list to

oo.) Overview

i.) Given a number (integer), create a list with ascending elements (starting at 1, up to the given number), and bind the resulting list to the List variable.

pp.) Example

?- iota(5,L).
L = [1, 2, 3, 4, 5] .

15.) sum(list, Sum)

qq.) Parameters

i.) list → the list whose elements we are summing
ii.) Sum → the variable to bind the sum of the elements to

rr.) Overview

i.) Given a list, sum all of its elements and bind the result to the Sum variable.

ss.) Example

?- sum([10,20,40,30],S).
S = 100.

16.) min(list, Min)

- tt.) Parameters
    - i.) list → the list whose elements we're finding the minimum of
    - ii.) Min→ the variable to bind the minimum element to
- uu.) Overview
    - i.) Given a list, find the minimum element, and bind it to the Min variable.
- vv.) Example

  ?- min([10,20,40,60,5,80,100],M).
  M = 5.

17.) max(list, Max)

- ww.) Parameters
    - i.) list → the list whose elements we're finding the maximum of
    - ii.) Max→ the variable to bind the maximum element to
- xx.) Overview
    - i.) Given a list, find the maximum element, and bind it to the Max variable.
- yy.) Example

  ?- max([10,20,40,60,5,80,100],Max).
  Max = 100 .

18.) sort_inc(list, Ordered)

- zz.) Parameters
    - i.) list → the list whose elements we're sorting incrementally
    - ii.) Ordered→ the variable to bind the resulting sorted list to
- aaa.) Overview
    - i.) Given a list, sort it incrementally, and bind the resulting list to the Ordered variable.
- bbb.) Example

  ?- sort_inc([10,20,40,60,5,80,100],Ordered).
  Ordered = [5, 10, 20, 40, 60, 80, 100] .

19.) sort_dec(list, Ordered)

- ccc.) Parameters
    - i.) list → the list whose elements we're sorting decrementally
    - ii.) Ordered→ the variable to bind the resulting sorted list to
- ddd.) Overview
    - i.) Given a list, sort it decrementally, and bind the resulting list to the Ordered variable.
- eee.) Example

  ?- sort_dec([10,20,40,60,5,80,100],Ordered).
  Ordered = [100, 80, 60, 40, 20, 10, 5] .

20.) alist(first_list, second_list, AssociationList)

fff.) Parameters

i.) first_list → the first list to create the association from

ii.) second_list → the second list to create the assocation from

iii.) AssociationList → the variable to bind the associative list to

ggg.) Overview

i.) Create an association list from two lists of equal length, and bind it to the AssociationList variable

hhh.) Example

?- alist([minnesota, green_bay, chicago, detroit],[vikings, packers, bears, lions], Assoc_List).
Assoc_List = [pair(minnesota, vikings), pair(green_bay, packers), pair(chicago, bears), pair(detroit, lions)].

21.) assoc(alist, key, Value)

iii.) Parameters

i.) alist → the association list to grab the value from

ii.) key → the key to search the association list for

iii.) Value → the variable to bind the value for the key to

jjj.) Overview

i.) find the Value in the second slot corresponding to the key in the first slot of some alist pair.

kkk.) Example

?-assoc([pair(minnesota,vikings),pair(green_bay,packers),pair(detroit,lions),pair(chicago,bears)],minnesota,V).
V = vikings .

22.) flatten(list, FlattenedList)

lll.) Parameters

i.) list →a list of lists to flatten

ii.) FlattenedList → the variable to bind the flattened list to

mmm.) Overview

i.) Turn a list of lists into a one-dimensional list, and bind it to FlattenedList variable.

nnn.) Example

?- flatten([[java,scala],[python,ruby], [c], [html,javascript]], L).
L = [java, scala, python, ruby, c, html, javascript] .