LP User Manual

Introduction

This program is called the List Processing (LP) program. It contains functions that have useful list processing predicates. This User Manual contains the various functions that were used in this program coupled with useful explanations,examples to help contextualize the code, and a handy Table of Contents to help with the navigation of this document.

Table of Contents

---

rest

```
rest(List, RestOfList)
```

Description: Creates a new list minus the first element of the original list.

Parameters:

List - list of elements.
RestOfList- List minus the first element of the original list.

Returns:

A list minus the first element of the original list.

Example:

```
?- rest([1,2,3], T).
T = [2, 3].
```

last_element

```
last_element(List, Last)
```

Description: Returns the last element of a list.

Parameters:

List - list of elements.
Last - last element of a list.

Returns:

The last element of a list.

Example:

```
?- last_element([1,2,3], T).
T = 3.
```

write_list

```
write_list(List)
```

Description: Returns the elements of a list.

Parameters:

List - list of elements.

Returns:

Elements of a list.

Example:

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

write_list_reversed

```
write_list_reversed(List)
```

Description: Returns the elements of a list in reverse order.

Parameters:
List - list of elements.

Returns:
Elements of a list in reverse order

Example:

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

size

```
size(List,L)
```

Description: Returns the size of a list.

Parameters:
List - list of elements.
Last - size of a list.

Returns:
The size of a list.

Example:

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

count

```
count(Element,List,Count)
```

Description: Returns the count of the specified element of a list.

Parameters:
Element - element to be searched in a list.
List - list of elements.
Count- count of element in a list.

Returns:
The size of a list.

Example:

```
?- count(it,[it,bit,fit,it,lit,pit],L).
L = 2.
```

element_of

```
element_of(Element,List)
```

Description: Returns a boolean indicating whether or not the specified element is in the list.

Parameters:
Element - element to be searched in a list.
List - list of elements.

Returns:
Boolean value indicating whether or not the specified element is in the list.

Example:

```
?- element_of(X,[it,bit,fit]).
X = it.
```

contains

```
contains(List,Element)
```

Description: Returns a boolean indicating whether or not the specified element is in the list.

Parameters:
List - list of elements.
Element - element to be searched in a list.

Returns:
Boolean value indicating whether or not the specified element is in the list.

Example:

```
?- contains([it,bit,pit]it).
false.
```

nth

```
nth(Index,List,Element)
```

Description: Returns the element of a list specified by the index.

Parameters:

Index - Position of the index that will be returned
List - list of elements.
Element - element to be searched in a list.

Returns:

The element of a list specified by the index.

Example:

```
?- nth(2,[how,are,you],Element).
Element = you.
```

pick

```
pick(List,Element)
```

Description: Returns a random element of a list.

Parameters:

List - list of elements.
Element - random element of a list.

Returns:

The element of a list specified by the index.

Example:

```
?- pick([it,bit,slit], Ele).
Ele = bit.
```

sum

```
sum(List,Element)
```

Description: Returns the sum of the elements of a list.

Parameters:

List - list of elements.
Element - sum of the elements of a list.

Returns:

The sum of the elements of a list.

Example:

```
?- sum([1,2,3], L).
L = 6.
```

make_list

```
make_list(Length,Element,List)
```

Description: Returns a list of length Length which contains the specified Element.

Parameters:

Length - length of the list.
Element - Element needed to construct the list
List- list of of the Element of length N

Returns:

A list of length Length which contains the specified Element.

Example:

```
?- make_list(3, Alikeju, List).
List = [Alikeju, Alikeju, Alikeju].
```

iota

```
iota(Length, List)
```

Description: Returns a list starting from 1 to Length.

Parameters:

Length - length of the list.
List - List starting from 1 to Length

Returns:

A list starting from 1 to Length.

Example:

```
?- iota(3, Iota).
Iota = [1,2,3].
```

add_first

```
add_first(Element,OldList,NewList)
```

Description: Returns a list beginning with the specified Element and ending with the original ending.

Parameters:

Element - element to add to the front of a list.
OldList - Original list specified by the user.
NewList - List beginning with the specified Element.

Returns:

A list beginning with the specified Element and ending with the original ending.

Example:

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

add_last

```
add_last(Element,OldList,NewList)
```

Description: Returns a list ending with the specified Element and beginning with the original ending.

Parameters:

Element - element to add to the end of a list.
OldList - Original list specified by the user.
NewList - List ending with the specified Element.

Returns:

A list ending with the specified Element and beginning with the original ending.

Example:

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

esrever

```
esrever(List,NewList)
```

Description: Returns a list in the reversed order of the specified list.

Parameters:

List - Original list specified by the user.
NewList - List n the reversed order of the specified list.

Returns:

A list in the reversed order of the specified list.

Example:

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

join_lists

Case 1

```
join_lists(List1,List2,NewList)
```

Description: Returns a list that is the joined list of List1 and List2.

Parameters:

List1 - First list specified by the user.
List2 - Second list specified by the user.
NewList - List that is the joined list of List1 and List2.

Returns:

A list that is the joined list of List1 and List2.

Example:

```
?- join_lists([hi,how],[are,you], L).
L = [hi, how, are, you].
```

Case 2

```
join_lists(List1,List2,List3,NewList)
```

Description: Returns a list that is the joined list of List1, List2, and List3.

Parameters:

List1 - First list specified by the user.
List2 - Second list specified by the user.
List3 - Third list specified by the user.
NewList - List that is the joined list of List1,List2, and List3.

Returns:

A list that is the joined list of List, List2 and List3.

Example:

```
?- join_lists([hi,how],[are,you], [today,bob], L).
L = [hi, how, are, you, today, bob].
```

Case 3

```
join_lists(List1,List2,List3,List4,NewList)
```

Description: Returns a list that is the joined list of List1, List2, List3, and List4.

Parameters:

List1 - First list specified by the user.
List2 - Second list specified by the user.
List3 - Third list specified by the user.
List4 - Fourth list specified by the user.
NewList - List that is the joined list of List1,List2, List3, and List4.

Returns:

A list that is the joined list of List, List2,List3, and List4.

Example:

```
?- join_lists([hi,how],[are,you], [today,bob], [you,look,great], L).
```

```
L = [hi, how, are, you, today, bob, you, look, great].
```

product

```
product(List,Product).
```

Description: Returns the product of a list.

Parameters:

List - List of numerical elements.
product - Product of a list.

Returns:

The product of a list.

Example:

```
?- product([1,3,3],L).
L = 9.
```

factorial

```
factorial(Element,Factorial).
```

Description: Returns the factorial of an element.

Parameters:

Element - Numerical element.
Factorial - Factorial of an element.

Returns:

The factorial of an element.

Example:

```
?- factorial(7, F).
F = 5040.
```

make_set

```
make_set(List,SetList).
```

Description: Removes the duplicates of the given list to create a set out of the given list and returns the set.

Parameters:

List - List of elements
SetList - Set of the given list.

Returns:

The factorial of an element.

Example:

```
?- make_set([b,b,c,c,d,d,e,e], F).
F = [b,c,d,e].
```

replace

```
replace(Index,Element,List,NewList).
```

Description: Returns the new list with the replaced element of index N.

Parameters:

Index - Index of the desired element to be replaced.
Element - Element that will be inserted into the new list.
List - Original list specified by the user.
NewList - New list with the element placed at the specified index.

Returns:

The new list with the replaced element of index N.

Example:

```
?- replace(2,stressed,[i,am,happy],L).
L - [i,am,stressed].
```

remove

```
remove(Element,List,NewList)
```

Description: Returns the new list with the removed element of index Element.

Parameters:

Element - Element that will be removed.
List - Original list specified by the user.
NewList - New list minus the element that was removed.

Returns:

The new list with the removed element of index Element.

Example:

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

take

```
take(List,Element,NewList)
```

Description: Returns the new list with the removed element of index Element.

Parameters:

List - Original list specified by the user.
Element - Element that will be removed.
NewList - New list minus the element that was removed.

Returns:

The new list with the replaced element of index N.

Example:

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

split

```
split(List1,List2,List3)
```

Description: Returns two new lists where the first list contains the first element of the first old list and the second list contains the second element of the second old list.

Parameters:

List1 - First list specified by the user.
List2 - Second list specified by the user.
List3 - Third list specified by the user.

Returns:

Two new lists where the first list contains the first element of the first old list and the second list contains the second element of the second old list.

Example:

```
?- split([[one,un],[two,deaux],[three,trois]],A,B).
A = [one,two,three],
B = [un, deaux, trois.
```

min_pair

```
min_pair(N1,N2,N1)
```

Description: Returns the smallest element of a pair.

Parameters:

N1 - First numeric element specified by the user.
N2 - Second numeric element specified by the user.

Returns:

The smallest element of a pair.

Example:

```
?- min_pair(2,4,L).
L = 2.
```

max_pair

```
max_pair(N1,N2,N1)
```

Description: Returns the largest element of a pair.

Parameters:

N1 - First numeric element specified by the user.
N2 - Second numeric element specified by the user.

Returns:

The largest element of a pair.

Example:

```
?- max_pair(2,4,L).
L = 4.
```

min

```
min(NumericList,MinNumber)
```

Description: Returns the smallest element of a list.

Parameters:

NumericList - numeric list specified by the user.
MinNumber - smallest number in a list.

Returns:

The smallest element of a list.

Example:

```
?- min([1,2,4],L).
L = 1.
```

max

```
max(NumericList,MaxNumber)
```

Description: Returns the largest element of a list.

Parameters:

NumericList - numeric list specified by the user.
MaxNumber - largest number in a list.

Returns:

The largest element of a list.

Example:

```
?- max([1,2,4],L).
L = 1.
```

sort_inc

```
sort_inc(UnorderedNumericList,OrderedNumericList)
```

Description: Returns a sorted list in increasing order.

Parameters:

UnorderedNumericList - numeric list specified by the user.
OrderedNumericList - list in increasing order

Returns:

A sorted list in increasing order.

Example:

```
?- sort_inc([2,8,1,4],L).
L = [1,2,4,8].
```

sort_dec

```
sort_dec(UnorderedNumericList,OrderedNumericList)
```

Description: Returns a sorted list in decreasing order.

Parameters:

UnorderedNumericList - numeric list specified by the user.
OrderedNumericList - list in increasing order

Returns:

A sorted list in decreasing order.

Example:

```
?- sort_dec([2,8,1,4],L).
L = [1,2,4,8].
```

a_list

```
a_list(List1,List2,AssociationList)
```

Description: Creates and returns an association list from the two lists of equal length, the first considered to be keys, the second considered to be values, and the third the resulting association list representing represented as pairs encapsulated into terms with the name ‘pair’.

Parameters:

List1 - List specified by the user.
List2 - List specified by the user.
AssociationList - Association list from the two lists of equal length.

Returns:

An association list from the two lists of equal length, the first considered to be keys, the second considered to be values, and the third the resulting association list representing represented as pairs encapsulated into terms with the name ‘pair’.

Example:

```
?- a_list([a,b,c,d],[1,2,3,4],AssociationList).
AssociationList-[pair(a,1),pair(c,3),pair(d,4),pair(b,2)].
```

assoc

```
assoc(AList,Key,Value)
```

Description: Finds the Value in the second slot corresponding to Key in the first slot of some pair of the association list and returns that Value and return that value.

Parameters:

AList - Association list specified by the user.
Key - The first slot of some pair.
Value - The second slot of some pair.

Returns:

The Value in the second slot corresponding to Key in the first slot of some pair of the association list and returns that Value.

Example:

```
?- assoc([pair(b,2),pair(a,1)],a,Value).
Value = 1.
```