CSC 366 List Processing User Manual

Part 1 Introduction

Lists are a very important part of programming and can help in algorithms. Most lists usually have a head and tail. As per their namesake the head is usually what comes first and the tail is what follows the head. For example if we had a list comprised of elements (1,2,3,4,5,6). If we used one of the below functions just as a simple one of finding the head. It would say the head would be equal to 1. Lists work differently in Prolog and have their own syntax and semantics. Below are 33 list functions and their descriptions and an example demo for each one.

Part 2 Functions

1. Syntax: first(List,H)
Semantic: Gives the first element of the list.
Demo:
```
?- first([0|1],H).
H = 0.
```
2. Syntax: rest(List,T).
Semantic: Gives the rest element of this list.
Demo:
```
?- rest([4|5],T).
T = 5.
```
3. Syntax: last_element(List,Result).
Semantic: Gives the last element in the list.
Demo:
```
?- last_element([a,b,c],H).
H = c .
```
4. Syntax: write_list(List).
Semantic: Display each element of the list in a new line
Demo:
```
?- write_list([a|[b,c,d]]).
a
b
c
d
true.
```
5. Syntax: write_list_reversed(List).

Semantic: Display each element of the list in reverse order

Demo:

```
?- write_list_reversed([a|[b,c,d,e]]).
e
d
c
b
a
true.
```

6. Syntax: size(List,Length).
Semantic: Outputs the size of the list.
Demo:

```
?- size([a,b,c,d],L).
L = 4.
```

7. Syntax: count(Element,List,Count).
Semantic: Gives the number of occurrences of a specific element in the list
Demo:

```
?- count(a,[a,v,b,a,c],C).
C = 2 ;
C = 1 ;
C = 1 ;
C = 0 ;
false.
```

8. Syntax: element_of(Element,List).
Semantic: Checks whether or not a certain element is in the list
Demo:

```
?- element_of(a,[a,b,c]).
true .
```

9. Syntax: contains(List,Element).
Semantic: Checks if the list contains the specific element
Demo:

```
?- contains([1,3,5,7,9],1).
true .
```

10. Syntax: nth(N,List,Element).
Semantic: Gives the element in the nth position in the list
Demo:

?- nth(0,[1,2,3],X).
X = 1 .
11. Syntax: pick(List,Item).
Semantic: Picks a random element from the list
Demo:
?- pick([a,b,c,d,e],X).
X = e .
12. Syntax: sum(List,Sum).
Semantic: Gives the sum of all the element of the list
Demo:
?- sum([1,2,3,4,5],S).
S = 15.
13. Syntax: make_list(Length, Element, List).
Semantic: Combines the two given lists into a single new list
Demo:
?- make_list(1,[[a,b,c,d],[1,2,3,4]],List).
List = [[[a, b, c, d], [1, 2, 3, 4]]] .
14. Syntax: iota(N,IotaN).
Semantic: Makes list from a specific beginning number to the size provided by the user
Demo:
?- iota(5,S).
S = [1, 2, 3, 4, 5] .
15. Syntax: add_first(Element,List,First).
Semantic: Adds an element at the beginning of the list
Demo:
?- add_first(I,[like,eating,sushi],Name).
Name = [I, like, eating, Sushi].
16. Syntax: add_last(Element,List,Last).
Semantic: Adds an element at the end of the list
Demo:
?- add_last(sushi,[I,like,eating],Last).
Last = [I, like, eating, sushi].
17. Syntax: esrever(InputList,Reverse).
Semantic: Makes all the elements in the list reverse
Demo:
?- esrever([f,a,k,e],Re).
Re = [e, k, a, f] .
18. Syntax: join_lists(List1,List2,Result).
Semantic: Joins two lists together

Demo:
?- join_lists([1,2,3,4],[5],Result).
Result = [1, 2, 3, 4, 5].
19. Syntax: product(List,Product).
Semantic: Gives the product of all elements in the list
Demo:
?- product([3,7],Product).
Product = 21.
20. Syntax: factorial(N,Factorial).
Semantic: Computes the factorial of elements in the list
Demo:
?- factorial(7,Factorial).
Factorial = 5040 .
21. Syntax: make_set(List,NewSet).
Semantic: Makes the list with only unique elements
Demo:
?- make_set([1,1,1,1,3,3,3,5,5,5,6],NewSet).
NewSet = [1, 3, 5, 6] .
22. Syntax: replace(N,Element,List,NewList).
Semantic: Replaces the common element from the list with new value
Demo:
?- replace(1,pen,[a,b,c,d],List).
List = [a, pen, c, d] .
23. Syntax: remove(Element,List,Removed).
Semantic: Removes specific element from the list
Demo:
?- remove(a,[a,b,c,d,e],List).
List = [b, c, d, e] .
24. Syntax: take(List,Pick,Rest).
Semantic: Remove a random element from a list and display it as two separate lists
Demo:
?- take([1,2,3,4,5,6],Dice,Rest).
Dice = 3,
Rest = [1, 2, 4, 5, 6] .
25. Syntax: split(List,First,Second).
Semantic: Splits two lists by matching the order of the elements
Demo:
?- split([[AA,BB]],First,Second).
First = [AA],

Second = [BB].
26. Syntax: min_pair(N1,N2,Min).
Semantic: Gives the minimum number amongst the pair
Demo:
?- min_pair(11,33,Min).
Min = 11 .
27. Syntax: max_pair(N1,N2,Max).
Semantic: Gives the minimum number amongst the pair
Demo:
?- max_pair(-22,23,Max).
Max = 23.
28. Syntax: min(List,Min).
Semantic: Gets a smallest number in the list
Demo:
?- min([3,6,0,-12,22],Min).
Min = -12 .
29. Syntax: max(List,Max).
Semantic: Gets a largest number in the list
Demo:
?- max([3,6,0,-12,22],Max).
Max = 22 .
30. Syntax: sort_inc(List, SortedList).
Semantic: Sorts the list in increasing order
Demo:
?- sort_inc([6,5,45,7,88,99,100,34],List).
List = [5, 6, 7, 34, 45, 88, 99, 100] .
31. Syntax: sort_dec(List, SortedList).
Semantic: Sorts the list in decreasing order
Demo:
?- sort_dec([-1,-2,-3,-55,-89,2],List).
List = [2, -1, -2, -3, -55, -89] .
32. Syntax: a_list(ListF, ListS, Alist).
Semantic: Create an association list
Demo:
?- a_list([1,2,3,4,5],[a,b,c,d,e],List).
List = [pair(1, a), pair(2, b), pair(3, c), pair(4, d), pair(5, e)].
33. Syntax: assoc(Alist,Key,Value).
Semantic: Find the value of a key in an association list
Demo:

```
?- assoc([pair(a, b), pair(c, d), pair(e, f)],cheese,Value).
Value = b .
```