Programming Languages - Assignment 9

Compiler Used : https://swish.swi-prolog.org/

Go to this URL -> Create a Program -> Enter the Code -> Run the Program by calling the function

Q1. Find the Last Element in the List

Filename : **last_element.pl**

https://drive.google.com/file/d/1OurUI9IW4KILW9tvO_FqSqDDzp3R08U8/view?usp=sharing

Program

```
% Return the last Element of the list

% Facts
return_empty_List("List is Empty!").
lastElement([H|[]],H).


% Rules
lastElement([_|T],Result):- lastElement(T,Result),!.

lastElement([],Result):- return_empty_List(Result).

```

Output : (just call the main function (lastElement) within and pass the list as shown below)

```
lastElement([1, 2, 3, 4], X).
X = 4
```

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

```
lastElement([15, 32,44,3, 94], X).
X = 94
```

```
?- lastElement([15, 32,44,3, 94], X).
```

Code :

```
% Return the last Element of the list
% Facts
return_empty_List("List is Empty!").
lastElement([H|[]],H).
% Rules
lastElement([_|T],Result):- lastElement(T,Result),!.
lastElement([],Result):- return_empty_List(Result).
```

Q2. Find the Max Element in the List

Filename : **max_element.pl**

https://drive.google.com/file/d/1B_aPYrXh_cG4WMnJb89dzRGpF2YIxXb_/view?usp=sharing

Program

```
% Find maximum in the given list

% Facts
emptyList("List is Empty!").
maxList([],A,A).
:- discontiguous maxList/3. % Add discontiguous directive here

% Rules
maxList([],Max) :- emptyList(Max).
maxList([H|T],Max) :- maxList([H|T],H,Max).
maxList([H|T],A,Max) :- H>A, maxList(T,H,Max),!; maxList(T,A,Max).
```

Output : (just call the main function (maxList) within and pass the list as shown below)

```
maxList([159, 32,44,3, 94,112], X).
```

X = 159

```
?- maxList([159, 32,44,3, 94,112], X).
```

```
maxList([159, 32,445,3, 94,112], X).
```

X = 445

```
?- maxList([159, 32,445,3, 94,112], X).
```

CODE:

```
% Find maximum in the given list


% Facts

emptyList("List is Empty!").

maxList([],A,A).

:- discontiguous maxList/3. % Add discontiguous directive here
```

% Rules

maxList([],Max) :- emptyList(Max).

maxList([H|T],Max) :- maxList([H|T],H,Max).

maxList([H|T],A,Max) :- H>A, maxList(T,H,Max),!; maxList(T,A,Max).

Q3. Reverse the List using Tail Recursion

Filename : **tail_reverse.pl**

https://drive.google.com/file/d/1MHTGH0ywk3uLoR4cL08CxZsRKtrM1hGE/view?usp=sharing

Program

```
% Reverse a list using tail recursion

% Base case: If the input list is empty, the reversed list is also empty
reverse_list([], Reversed, Reversed).

% Recursive case: Append the head of the input list to the accumulator list and c
reverse_list([Head|Tail], Acc, Reversed) :-
    reverse_list(Tail, [Head|Acc], Reversed).
```

Output : (just call the main function (reverse_list) as shown below and pass the list

```
reverse_list([1, 2, 3, 4], [], Reversed).
Reversed = [4, 3, 2, 1]
reverse_list([16, 72, 23, 44, 54, 84, 23], [], Reversed).
Reversed = [23, 84, 54, 44, 23, 72, 16]
?- reverse_list([16, 72, 23, 44, 54, 84, 23], [], Reversed).
```

Code :

% Reverse a list using tail recursion

% Base case: If the input list is empty, the reversed list is also empty

reverse_list([], Reversed, Reversed).

% Recursive case: Append the head of the input list to the accumulator list and continue with the tail

reverse_list([Head|Tail], Acc, Reversed) :-

  reverse_list(Tail, [Head|Acc], Reversed).

Explanation of How it is Tail Recursive :

reverse_list([], Reversed, Reversed).

This is the base case, which simply unifies the accumulator (**Reversed**) with the result.

Prolog

reverse_list([Head|Tail], Acc, Reversed) :- reverse_list(Tail, [Head|Acc], Reversed).

In the recursive case, the recursive call to **reverse_list/3** is the last operation performed, after which the result is returned directly without any further computation. This recursive call is tail-recursive because there are no more operations to perform after it.

Furthermore, the accumulator (**Acc**) is used to accumulate the reversed list as the recursion progresses, which is a characteristic of tail-recursive algorithms.

Therefore, this code does indeed use tail recursion to reverse a list.

Q4. Given two list that represent sets of numbers (no duplicates in a set) calculate the union of the two sets which produces another set

Filename : union_list.pl

https://drive.google.com/file/d/13GJBlsMZh_qqZphw1wMlevjNK1RiG2q-/view?usp=drive_link

Program

```
:- discontiguous unionList/3.
:- discontiguous member/2.

% Facts
unionList([],Z,Z).
member(X,[X|_]).

% Rules
unionList([H|T],L2,Result):- member(H,L2),!,unionList(T,L2,Result).
unionList([H|T],L2,[H|Result]):- \+member(H,L2),!,unionList(T,L2,Result).
member(X,[_|T]):- member(X,T).

```

Output : Call the main function unionList and pass the two lists as shown below.

```
unionList([1, 2, 3], [3, 4, 5], Result).
Result = [1, 2, 3, 4, 5]
unionList([1, 2, 3, 4, 5 ,7 , 8, 9], [3, 4, 5, 22, 25, 28], Result).
Result = [1, 2, 7, 8, 9, 3, 4, 5, 22, 25, 28]
?- unionList([1, 2, 3, 4, 5 ,7 , 8, 9], [3, 4, 5, 22, 25, 28], Result).
```

Note : As told in the question, the input lists have to be set, i.e no duplicates. If you give duplicates , this doenst work.

```
unionList([1, 2, 3, 4, 4, 4, 4], [5, 5, 5, 6, 7, 8], Result).
Result = [1, 2, 3, 4, 4, 4, 4, 5, 5, 5, 6, 7, 8]
?- unionList([1, 2, 3, 4, 4, 4, 4], [5, 5, 5, 6, 7, 8], Result).
```

It works only if the two lists are sets, as asked in the question

```
unionList([1, 2, 3, 4], [5, 6, 7, 8], Result).
Result = [1, 2, 3, 4, 5, 6, 7, 8]
?- unionList([1, 2, 3, 4], [5, 6, 7, 8], Result).
```

Code :

```
:- discontiguous unionList/3.
:- discontiguous member/2.

% Facts
unionList([],Z,Z).
member(X,[X|_]).

% Rules
unionList([H|T],L2,Result):- member(H,L2),!,unionList(T,L2,Result).
unionList([H|T],L2,[H|Result]):- \+member(H,L2),!,unionList(T,L2,Result).
member(X,[_|T]):- member(X,T).
```

Thank you

Sai Nishanth Mettu

sm11326

GOOGLE DRIVE LINKS TO THE CODE FILES

https://drive.google.com/file/d/1B_aPYrXh_cG4WMnJb89dzRGpF2YIxXb_/view?usp=sharing

https://drive.google.com/file/d/13GJBlsMZh_qqZphw1wMlevjNK1RiG2q-/view?usp=drive_link

https://drive.google.com/file/d/1MHTGH0ywk3uLoR4cL08CxZsRKtrM1hGE/view?usp=drive_link

https://drive.google.com/file/d/1B_aPYrXh_cG4WMnJb89dzRGpF2YIxXb_/view?usp=drive_link

IN VITAL

Steps

1. Open Terminal
2. swipl filename.pl

3. Run the main function inside the file like…. MainFunction([1,2,3,4,5], [], Reversed).
#Example for Tail Recursion…similar to all. (below)