

Programming Languages - Assignment 9

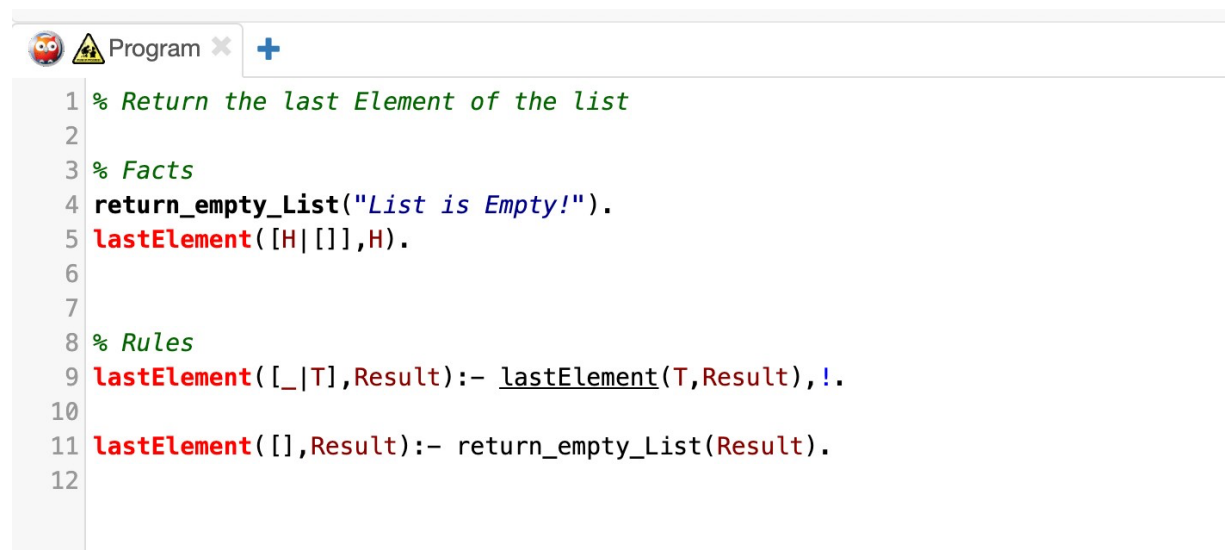
Compiler Used : <https://swish.swi-prolog.org/>

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

A screenshot of a web-based Prolog editor interface. The window title is "Program" with a close button and a plus sign. The code is written in Prolog and includes comments in green. The code defines a function to find the last element of a list. It starts with a comment "% Return the last Element of the list", followed by a blank line. Then a comment "% Facts" is shown, followed by a blank line. The code defines a predicate "return_empty_List" that takes a string and returns true. Then it defines "lastElement" with two clauses: one for a non-empty list [H|_] and one for an empty list []. The first clause calls "lastElement" recursively on the tail of the list. The second clause calls "return_empty_List". The code is numbered 1 through 12 on the left margin.

```
1 % Return the last Element of the list
2
3 % Facts
4 return_empty_List("List is Empty!").
5 lastElement([H|_],H).
6
7
8 % Rules
9 lastElement([_|T],Result):- lastElement(T,Result),!.
10
11 lastElement([],Result):- return_empty_List(Result).
12
```

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_cG4WMnJb89dzRGpF2YlXb_/view?usp=sharing

```
Program x +
1 % Find maximum in the given list
2
3 % Facts
4 emptyList("List is Empty!").
5 maxList([],A,A).
6 :- discontiguous maxList/3. % Add discontiguous directive here
7
8 % Rules
9 maxList([],Max) :- emptyList(Max).
10 maxList([H|T],Max) :- maxList([H|T],H,Max).
11 maxList([H|T],A,Max) :- H>A, maxList(T,H,Max),!; maxList(T,A,Max).
12
13
```

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).

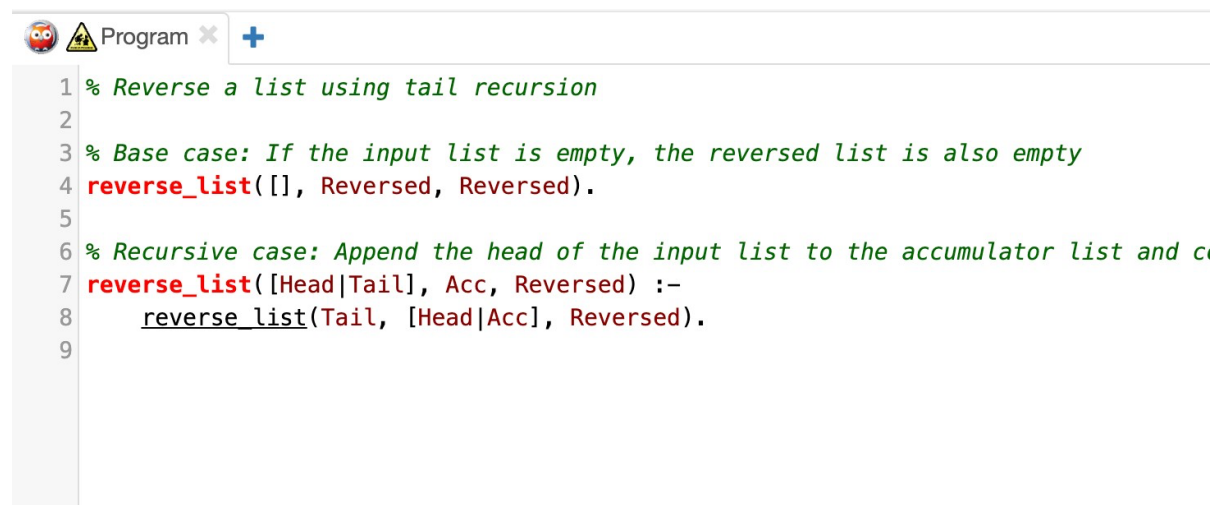
maxList([H|T],Max) :- maxList([H|T],H,Max).

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>

A screenshot of a text editor window titled "Program" with a yellow warning icon. The editor contains Prolog code for reversing a list using tail recursion. The code is as follows:

```
1 % Reverse a list using tail recursion
2
3 % Base case: If the input list is empty, the reversed list is also empty
4 reverse_list([], Reversed, Reversed).
5
6 % Recursive case: Append the head of the input list to the accumulator list and call reverse_list on the tail
7 reverse_list([Head|Tail], Acc, Reversed) :-
8     reverse_list(Tail, [Head|Acc], Reversed).
9
```

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_qqZphw1wMlevjNK1RiG_2q-/view?usp=drive_link

```
1 :- discontinuous unionList/3.
2 :- discontinuous member/2.
3
4 % Facts
5 unionList([],Z,Z).
6 member(X,[X|_]).
7
8 % Rules
9 unionList([H|T],L2,Result):- member(H,L2),!,unionList(T,L2,Result).
10 unionList([H|T],L2,[H|Result]):- \+member(H,L2),!,unionList(T,L2,Result).
11 member(X,[_|T]):- member(X,T).
12
```

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 doesn't 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 :

:- disjoint unionList/3.

:- disjoint 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_cG4WMnJb89dzRGpF2YlxXb_/view?usp=sharing

https://drive.google.com/file/d/13GJBlsMZ_hqqZphw1wMlevjNK1RiG2q-/view?usp=drive_link

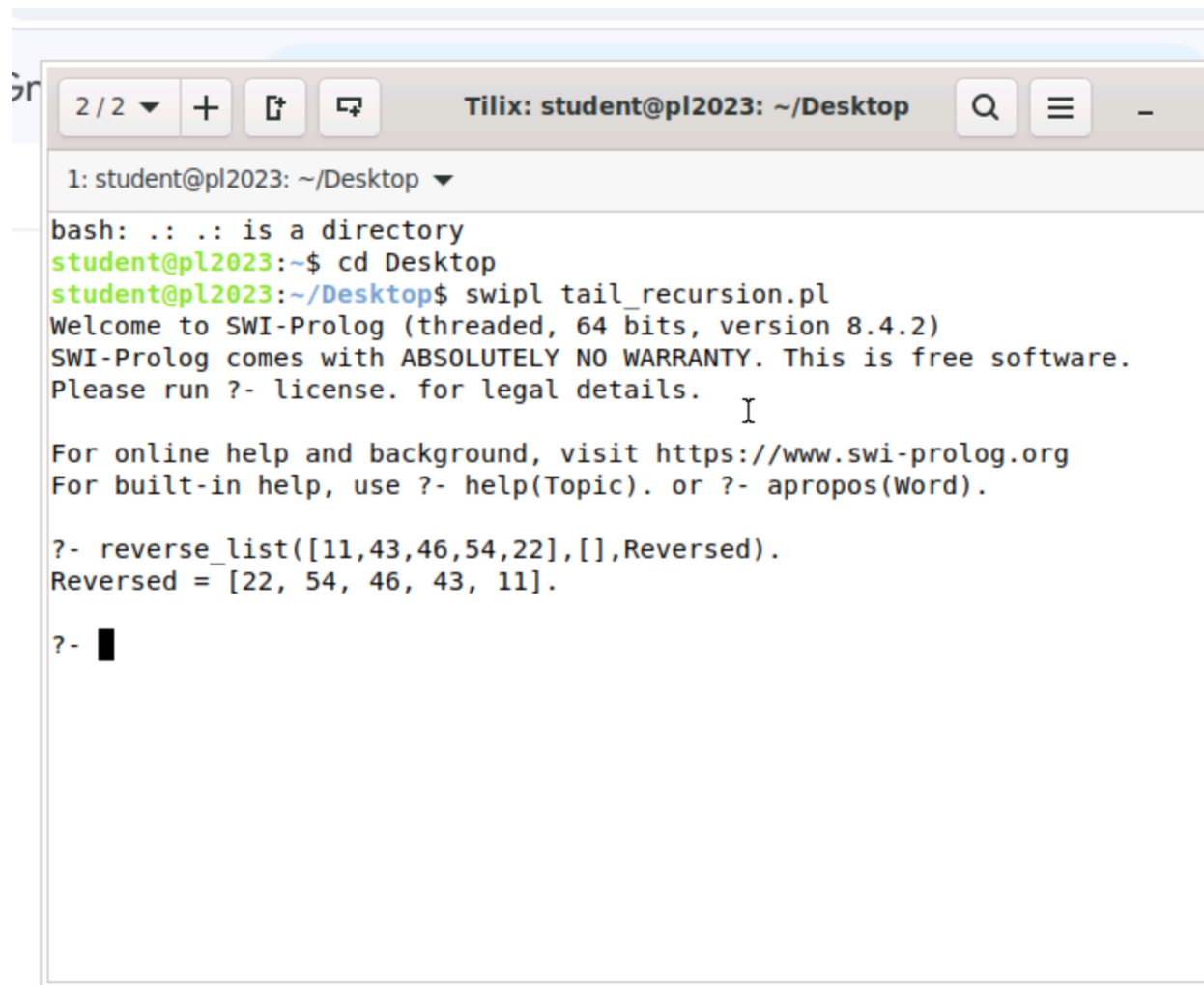
https://drive.google.com/file/d/1MHTGH0ywk3uLoR4cL08CxZsRKtrM1hGE/view?usp=drive_link

https://drive.google.com/file/d/1B_aPYrXh_cG4WMnJb89dzRGpF2YlxXb_/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)



```
2 / 2 ▾ + [?] [?] Tilix: student@pl2023: ~/Desktop Q ≡ -  
1: student@pl2023: ~/Desktop ▾  
bash: .: .: is a directory  
student@pl2023:~$ cd Desktop  
student@pl2023:~/Desktop$ swipl tail_recursion.pl  
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
I  
For online help and background, visit https://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
?- reverse_list([11,43,46,54,22],[],Reversed).  
Reversed = [22, 54, 46, 43, 11].  
?- █
```