Concepts of Programming Languages, CSCI 305, Fall 2021
Lab 5, Prolog, Oct. 15

Use SWI-Prolog which is installed on the lab machines.

Write prolog code to perform the following functions. (Note that you have already written Scheme code for many of these.)

1. Define the *sum* function which takes two numbers and returns the sum of the numbers.

   Example:

   sum(4,5,X).   returns X = 9.

   ```
   sum(X,Y,Ans) :- Ans is X+Y.
   ```

2. Define the *sumList* function which takes a list of numbers and returns the sum of them. If the list is empty, return 0.

   ```
   sumList([],0).
   sumList([H|T], Ans) :-
           sumList(T, SumRest),
           Ans is H+SumRest.
   ```

3. Define the function *average* which takes two numbers and returns the average of the numbers.

   ```
   average(X, Y, Ans) :- Ans is (X+Y)/2.
   ```

4. Define the function *averageList* which takes a non-empty list of numbers and returns the average of the numbers in the list.

   ```
   averageList(X, Ans) :-
           sumList(X, Total),      % Place sum of the list in variable Total
           length(X, Length),      % Place length of the list in variable Length
           Ans is Total/Length.    % Place average in variable Ans
   ```

5. Define a function *convertFCList* which takes a list of Fahrenheit measurements an converts them to a list of Celsius measurements where:

   Celsius = (Fahrenheit – 32) 5/9

```
convertFC(X,Ans) :- Ans is (X-32)*5/9.      % Helper function

convertFCList([],[]).                       % Base case
convertFCList([H|T], Ans) :-
        convertFC(H, C_head),               % Place conversion of (car list) into C_head
        convertFCList(T, C_tail),           % Place the converted (cdr list) into C_tail
        Ans = [C_head|C_tail].              % Use pattern matching to build the answer.
                                            % In Scheme language, place
                                            %           (cons C_head  C_tail) into Ans
```

6. Define the function *myLast* which returns the last element of a list.

```
myLast([],X) :- X=error.                    % Want a non-empty list
myLast([X],X).                              % Base case
myLast([_|T],X) :- myLast(T,X).             % Recursively call with (cdr list) until base
                                            % case is reached
```

Using variable H instead of '_' works, however, the system warns that H is never used,

```
        Warning: c:/users/cschahczenski/documents/prolog/lab06.pl:19:
                Singleton variables: [H]
```

7. Define the function *myFilterList* which takes a low value, a high value and a list and produces a list of the values in the original list which fall within the low to high range (inclusively).

```
myFilterList(_, _, [], []).         % Base case
myFilterList(Low, High, [H|T], Ans) :-
        myFilterList(Low,High,T,SubAns),            % Handle tail of list
        ((Low=<H,High>=H)->Ans=[H|SubAns];Ans=SubAns).
                                                    % Include or don't include
```

Syntax of conditional statement:

```
<condition> -> (do_something) ; (do_something else)
```

Answer not using a conditional statement:

```
filterOne(Low, High, X, Ans) :-        % Helper function for included value
        Low=<X,
        X=<High,
        Ans = true.

filterOne(Low, High, X, Ans) :-        % Helper function for excluded value
        (X<Low; X>High),
        Ans = false.

myFilter(_ , _, [], []).      % Base case for empty list

myFilter(Low, High, [H|T], Ans):-        % Included element
                filterOne(Low, High, H, true),
                myFilter(Low, High, T, ListOfRest),
                Ans = [H|ListOfRest].

myFilter(Low, High, [H|T], Ans):-      %Excluded element
                filterOne(Low, High, H, false),
                myFilter(Low, High, T, ListOfRest),
                Ans = ListOfRest.
```

8. Define a function *eliminate_exp* (for eliminate expensive) which takes a maximum price and a list of prices. It returns the list of prices with those prices which exceed the maximum removed. (You can assume all prices are positive.)

```
eliminate_exp(Max, List, Ans):-
        myFilter(0, Max, List, Ans).
```