COMP 410
Fall 2024

More Recursion and Accumulators in Prolog

1.) Write a procedure named `sumAll` that will find the sum of a list of numbers. The sum of an empty list is zero. Do not write a helper procedure. An example query is below:

```
?- sumAll([4, 3, 2, 7], X).
X = 16.
```

2.) Write a procedure named sublist that will nondeterministically return all the lists which can be constructed from the elements in an input list. Do not write a helper procedure. An example query is below, where semicolon (`;`) was repeatedly pressed to get all answers:

```
?- sublist([1, 2, 3], X).
X = [1, 2, 3] ;
X = [1, 2] ;
X = [1, 3] ;
X = [1] ;
X = [2, 3] ;
X = [2] ;
X = [3] ;
X = [].
```

3.) Write **two** procedures named `sumAllAccum` which perform the same operation as `sumAll`, but they make use of an accumulator. The first `sumAllAccum` procedure should simply call the second `sumAllAccum` procedure with an initial accumulator. An example query is below:

```
?- sumAllAccum([3, 2, 8, 1], X).
X = 14.
```

4.) Write **two** procedures named `reverse` which will reverse the elements of a list. The second procedure should make use of an accumulator, and the first procedure should call the second with an initial accumulator. Your procedure should run in `O(n)`. An example query is below:

```
?- reverse([1, 2, 3], X).
X = [3, 2, 1].
```