

Worth: 10% (together with part I of this assignment)

Due: By 1pm on Wednesday 8 April

For all of your answers, you are expected to use good Prolog style (in particular, make appropriate use of unification, helper predicates, and cut) including good external and internal comments (to explain the purpose of your code and your decisions on how to implement it), as well as good visual layout (formatting, indenting, blank lines) to make your code easy to read and understand.

For each question below, predicates will be described using the notation introduced in class, *i.e.*, parameters preceded by a “+” must be instantiated, parameters preceded by a “-” must not be instantiated, and parameters preceded by a “?” may or may not be instantiated, in order for the computation to produce correct results.

1. Write a predicate `permute(?L0,?L1)` that succeeds iff list `L1` is a permutation of list `L0`.

With backtracking, your predicate should generate every permutation of a list exactly once. For example,

```
?- permute([1,2,3], P).
P = [1, 2, 3] ;
P = [2, 1, 3] ;
P = [2, 3, 1] ;
P = [1, 3, 2] ;
P = [3, 1, 2] ;
P = [3, 2, 1] ;
No
```

(your code does not have to generate permutations in the same order as this example—it can be correct as long as every permutation is generated exactly once).

Submit your code for this question in file “`permute.pl`”.

2. Write a predicate `simple(+E,?S)` that succeeds iff `S` is a simplified form of the arithmetic expression `E`. For this question, we consider only expressions made up of numbers, variables (Prolog atoms), and basic arithmetic operators (+, -, *, /, ^, where ^ represents exponentiation). Your predicate should perform the following simplifications:
 - If `E` contains only numbers, then the query `simple(E,S)` should bind `S` to the numerical value of `E`.
 - The expressions “`0+E`”, “`E+0`”, “`E-0`”, “`1*E`”, “`E*1`”, “`E/1`”, and “`E^1`” should be simplified in the same way as `E` itself.
 - The expressions “`0-E`”, “`-1*E`”, “`E*-1`”, and “`E/-1`” should be simplified in the same way as `-E`.
 - The expression “`E+F`” should simplify to 0 if the simplified forms of `E` and `F` are the same, but with opposite signs.
 - The expressions “`-E+F`” and “`E-F`” should simplify to 0 if the simplified forms of `E` and `F` are the same.
 - The expressions “`0*E`”, “`E*0`”, and “`0/E`” should simplify to 0.
 - The expressions “`E/F`”, “`E*1/F`”, and “`1/E*F`” should simplify to 1 if the simplified forms of `E` and `F` are the same (and non-zero).
 - The expression “`E^0`” should simplify to 1.

For example,

```
?- simple(2^5+3*7, S).  
S = 53 ;  
No
```

```
?- simple(3*2*x^2 - 1*x^0 + (3-2)*x, S).  
S = 6*x^2-1+x ;  
No
```

Recall that in Prolog, the predicate `atom(X)` succeeds iff `X` is an “atom” (*i.e.*, an identifier that starts with a lowercase letter, like `x` or `artichoke`).

Submit your code for this question in file “`simple.pl`”.

Important Note: There are certain simplifications that your code will not handle, *e.g.*,

```
?- simple(2+x-x, S).  
S = 2+x-x ;  
No % because of associativity, this is parsed as (2+x)-x
```

```
?- simple(2*x-3*x, S).  
S = 2*x-3*x ;  
No % the simplifier doesn't have to handle distributivity
```

This is fine! You can get full marks as long as you have implemented all the basic simplifications above.